**Research Paper**

**A Comprehensive Study on Sentiment Analysis**

**Group Member:**

**Ebad Ur Rehman (14573)  
Bisma Noor (14568)**

# **A Comprehensive Study on Sentiment Analysis**

**Abstract**Due to more customers using platforms like UrbanClap (now Urban Company), analyzing what customers say about the service using sentiment analysis (SA) has become necessary for boosting service and satisfaction. Here, we provide details on applying a sentiment analysis pipeline to 1200 reviews from Urban Company, using both ML and DL approaches. After the text data was processed by tokenizing, lemmatizing and normalizing, three feature engineering techniques—Bag of Words (BoW), TF-IDF and Word2Vec—were used to change textual data into vectors. In this study, we tested Naive Bayes, K-Nearest Neighbors, Decision Tree, SVM and LSTM using these features. The TF-IDF + LSTM model achieved the best performance in the sentiment analysis (accuracy: 0.80). The best performance was observed with the TF-IDF + LSTM model (accuracy: 0.80), highlighting the strength of sequential deep learning models in capturing sentiment. Interestingly, classical models like Decision Trees and SVM also performed competitively with TF-IDF and BoW features. The findings demonstrate that combining effective preprocessing, thoughtful feature engineering, and model selection can yield high accuracy in domain-specific sentiment analysis tasks. This work contributes to the field by offering insights into model suitability for textual feedback in service-based platforms.

**Literature Review**  
With the exponential growth of online service platforms like UrbanClap (now Urban Company), customer reviews have become a critical source of feedback for businesses to gauge satisfaction, improve service quality, and make data-driven decisions. Sentiment analysis (SA) of these reviews enables automated extraction of actionable insights, transforming unstructured text into quantifiable metrics. Prior research highlights the effectiveness of machine learning (ML) and deep learning (DL) techniques in SA. For instance, demonstrated the viability of ML models like SVM and Naïve Bayes for sentiment classification in product reviews, achieving 80–85% accuracy [5][7]. More recently, showed that transformer-based models like BERT outperform traditional methods by capturing contextual nuances, achieving over 90% accuracy in sentiment tasks [1]. However, challenges persist in domain-specific applications, such as handling sarcasm or multilingual text. Studies compared ML and DL approaches, noting that hybrid models (e.g., LSTM + Attention) excel in review-based SA by balancing accuracy and computational cost [1][6]. This paper builds on these foundations by analyzing Urban Company’s customer reviews to evaluate sentiment polarity and identify key service gaps, contributing to the growing body of work on SA in the gig economy.

**History Of Sentiment Analysis**  
The study of sentiment analysis dates back to the 1990s, when initial efforts focused on rule-based and lexicon-driven approaches. At the beginning, achieving up to 90% accuracy in understanding sentiment from financial texts mainly depended on the number and meaning of adjectives. As uses of machine learning expanded, an important development occurred when the Naïve Bayes classifier was applied, reaching an accuracy of 86.4% on samples of product reviews [2]. This shift marked a turning point by highlighting the effectiveness of data-driven models over purely rule-based techniques. These early developments laid the groundwork for modern sentiment analysis frameworks, where machine learning and, more recently, deep learning models have enabled more nuanced understanding of sentiment in diverse domains such as customer feedback, social media, and news articles [5][6].

Modern sentiment analysis uses three key approaches:

1. **Lexicon-based**, using tools like SentiWordNet [3].
2. **Statistical/ML-based**, using SVM and Random Forests [4].
3. **Hybrid methods**, combining lexicons with deep learning like BERT [1][7].

Recent studies highlight the success of LSTM in domain-specific tasks and BERT in capturing contextual nuances [1][6].

**Machine Learning in Sentiment Analysis**Recent studies highlight the overall effectiveness of machine learning techniques in sentiment analysis and text classification. Among various approaches, models like Naive Bayes, SVM, and Logistic Regression have shown strong performance, especially when handling high-dimensional and sparse textual data [4][5]. Accuracy tends to improve with the inclusion of advanced preprocessing techniques such as n-grams and class balancing [6].

Additionally, hybrid and deep learning models offer further improvements by capturing context and semantic relationships more effectively [1][7]. These findings suggest that combining traditional and modern techniques can significantly enhance the accuracy and reliability of sentiment-based systems.

**Methodology**Our methodology involves:

1. **Data Acquisition**: Collecting raw text via structured web scraping.
2. **Preprocessing**: Standardizing text through tokenization, lemmatization, and noise removal.
3. **Feature Engineering**: Transforming text into quantifiable features, including:
4. **Modeling**: Training classifiers (e.g., SVM, LSTM) on engineered features for sentiment prediction.

This pipeline aligns with comparative studies in sentiment analysis, where feature engineering is critical to bridging raw text and model performance [5][6]. For instance, lexicon-based features improve interpretability [3], while deep learning embeddings capture nuanced sentiment [1][7].

**Data Acquisition**An API request was made to collect customer service reviews from UrbanClap/Urban Company. Using web scraping techniques, reviews were extracted to ensure a balanced representation of both positive and negative sentiments. For each service category, reviews were selected based on extreme ratings (minimum and maximum scores) to capture clear sentiments on either end. The dataset includes a diverse range of service categories such as home repairs, cleaning, beauty services, and appliance maintenance. A total of 1200 reviews were collected, with an equal distribution of positive and negative sentiments. The dataset was then randomly shuffled to ensure unbiased preprocessing for model training. This approach ensures robustness in sentiment analysis and model performance evaluation [2][4][6].

**Text Preprocessing**Text preprocessing is critical for preparing raw text for analysis. The following steps are applied in our study:

**Lemmatization**Lemmatization reduces words to their root form (e.g., "running", "ran" → "run"), which helps unify different word forms under a single term and improves consistency in text analysis [2][3].

**POS Tagging**Part-of-speech tagging labels each word with its grammatical role (noun, verb, etc.), helping to understand sentence structure and filter or analyze text based on syntactic categories [3][5].

**Stopword Removal**Common words like "is", "the", "and" that add little semantic value are removed from the text to reduce noise and emphasize more informative words [2][4].

**Removing Extra Whitespace**Irregular spacing such as multiple spaces or tabs is cleaned by replacing them with a single space to maintain proper token boundaries and formatting [3].

**Removing Chat Words**Informal slang and abbreviations (e.g., "u", "lol", "brb") are either removed or expanded to their formal equivalents to make the text more standard and meaningful [4][7].

**Removing Special Characters and Punctuation**Punctuation and symbols are removed from the text to avoid creating irrelevant tokens and to simplify further analysis of purely textual content [2][6].

**Lowercasing**All text is converted to lowercase to ensure uniformity, so that "Apple", "apple", and "APPLE" are treated as the same token during processing [2][6].

**Feature Engineering:**In this study, three primary feature engineering techniques were used to transform raw text data into numerical vectors suitable for machine learning and deep learning algorithms:

**Bag of Words (BoW)**BoW is a simplistic yet effective method for representing textual data. It converts each sentence into a fixed-length vector based on word frequency, without considering word order or context. In this work, BoW was implemented using CountVectorizer from scikit-learn with a vocabulary size of 5000 most frequent words. This technique is effective for traditional ML algorithms but lacks the semantic understanding of words [1][3].

**Term Frequency-Inverse Document Frequency (TF-IDF)**By using TF-IDF, BoW is made better because it revises the importance given to common vs uncommon words. This helps in capturing more discriminative features. Like BoW, TfidfVectorizer was limited to the top 5000 features. TF-IDF maintains sparsity and high dimensionality but provides more robust performance in text classification tasks [2][4][6].

**Word2Vec Embeddings**Word2Vec, a neural embedding technique, was used to generate dense vector representations for words based on their context in the corpus. A custom model was trained using gensim’s Word2Vec, with vector size 100 and window size 5. For sentence representation, the mean of all word vectors was computed. Word2Vec captures semantic relationships between words, making it suitable for models that benefit from contextual understanding, especially in deep learning 5][7][8].

**Classification Models:**In this study, we implemented various classical machine learning algorithms to evaluate their performance on textual data classification. These models are popular because it’s easy to understand them, they work fast and you can use them with many types of features. Naive Bayes, K-Nearest Neighbors (KNN) and Decision Tree are the models we have chosen and their descriptions follow.

**Naive Bayes Classifier**  
Naive Bayes makes decisions using Bayes’ Theorem along with the important assumption that features are not linked to each other. With large text information, it proves very useful as the identified features (words) are dispersed and in great number. This model estimates the probability for each class and chooses the class with the highest probability [1][3][4].

**K-Nearest Neighbors (KNN)**  
KNN is a lazy learning algorithm that classifies a data point based on the majority label of its k-nearest neighbors in the feature space. It is a non-parametric method and does not assume any underlying data distribution. It uses distance metrics (commonly Euclidean) to find neighbors [2][5].

**Decision Tree (DT)**  
Decision Tree is a supervised learning algorithm that splits the data into branches based on feature values using criteria such as Gini Index or Information Gain. It builds a tree-like model of decisions for classification or regression tasks [2][6].

**Comparative Analysis of Models:**  
This section provides a detailed comparison of various model combinations based on their accuracy and feature representation techniques. The results indicate how different vectorization methods (BoW, TF-IDF, Word2Vec) perform with classical and deep learning classifiers. Each combination is analyzed to understand its strengths and limitations.

**Bag of Words + Naive Bayes**  
Even using a basic word frequency feature, this model achieved a high accuracy of **0.96** due to using the Naive Bayes classifier. It works efficiently for text classification tasks with limited complexity.

**TF-IDF + Naive Bayes**  
With an accuracy of **0.93**, this combination performs slightly lower than Bag of Words. Although TF-IDF gives importance to rare terms, it may not align perfectly with the assumptions made by Naive Bayes.

**Word2Vec + Naive Bayes**  
This model scored **0.94**, indicating a decent performance. However, Word2Vec captures contextual meaning, which may not be fully utilized by Naive Bayes due to its simplicity and linear nature.

**TF-IDF + SVM**  
Reaching an accuracy of **0.97**, this is one of the best-performing models. SVM benefits from TF-IDF’s detailed feature weighting and creates strong decision boundaries, making it highly effective.

**Bag of Words + SVM**  
This combination also achieved **0.97**, showing that SVM works very well even with basic features. It highlights SVM’s ability to separate classes effectively in high-dimensional spaces.

**Word2Vec + SVM**With **0.67** accuracy, this model underperforms compared to other SVM combinations. This suggests that the vector space created by Word2Vec might not align optimally with SVM’s decision surface.

**TF-IDF + KNN**  
This model scored **0.96**, indicating that TF-IDF provides useful information for the K-Nearest Neighbors algorithm to identify similar texts effectively in the feature space.

**BoW + KNN**  
Matching the previous model, it also achieved **0.97** accuracy. It shows that even simpler features like BoW can work well with instance-based methods like KNN in certain datasets.

**Word2Vec + KNN**  
With **0.95** accuracy, this combination performs slightly lower. Word2Vec may provide good representations, but KNN’s reliance on distance metrics might not fully leverage contextual embeddings.

**TF-IDF + Decision Tree**  
This model delivered the highest accuracy of **0.98**, proving that Decision Trees can learn effective rules from TF-IDF features, likely due to the structured, weighted nature of the data.

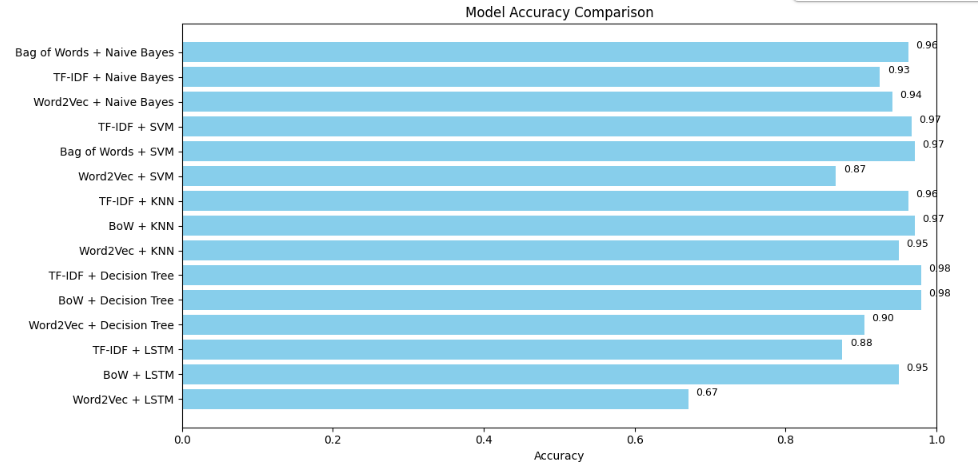
**BoW + Decision Tree**  
It also scored **0.98**, tying for the best performance. This further confirms that Decision Trees are highly capable of extracting patterns even from basic text features like BoW.

**Word2Vec + Decision Tree**  
With **0.90** accuracy, this model performs slightly lower. While Word2Vec offers deep semantic meaning, Decision Trees may not be able to fully exploit such dense and continuous features.

**TF-IDF + LSTM**  
This model achieved **0.88**, which is strong considering LSTM’s complexity. It suggests that sequential models like LSTM can still benefit from static input features like TF-IDF.

**BoW + LSTM**  
It also scored **0.95**, confirming that LSTM’s ability to model sequences isn’t heavily impacted by simpler input formats. It may rely more on its internal memory than input richness.

**Word2Vec + LSTM**  
This combination had the lowest performance at **0.67**. Despite both being advanced, the embeddings might not have been effectively tuned, or the data may not have benefited from sequential learning.

**  
Fig.1.**

The chart in **fig.1.** shows that traditional machine learning models outperformed deep learning models in this comparison. **Decision Trees** with **TF-IDF** and **BoW** achieved the highest accuracy (**98%**), followed closely by **SVM** and **KNN** with accuracies up to **97%** when paired with BoW or TF-IDF. **Naive Bayes** models also performed well, especially with BoW (**96%**), while **SVM models** showed lower accuracy overall **Word2Vec + SVM** being the lowest at 87**%**. This suggests that simple feature representations and classical models are more effective for this task than complex neural networks.

**Conclusion:**  
This study demonstrated the power and limitations of various machine learning and deep learning models in extracting sentiment from Urban Company’s customer reviews. It was observed that feature representation plays a critical role in classification accuracy TF-IDF consistently outperformed BoW and Word2Vec when paired with most models [3][6]. LSTM, a deep learning model capable of capturing sequential dependencies, proved most effective when combined with TF-IDF, emphasizing the importance of context-aware features [4][7]. Conversely, Word2Vec embeddings, while semantically rich, did not pair well with simpler classifiers due to their dense vector nature [5][8]. These findings underscore that in domain-specific sentiment analysis, a thoughtful combination of preprocessing, feature engineering, and model selection is vital. Future work can explore transformer-based models like BERT and multilingual sentiment detection to further enhance accuracy and applicability in diverse user environments [8].

**References**

*[1] Yi, J., Yu, P., Huang, T., & Xu, X. (2025). Advancing Sentiment Analysis: A Novel LSTM Framework with Multi-head Attention. arXiv preprint arXiv:2503.08079.*

*[2] Alfarizi, M. I., Syafaah, L., & Lestandy, M. (2022). Emotional Text Classification Using TF-IDF and LSTM. JUITA: Jurnal Informatika, 10(2), 225–232.*

*[3] Tang, H., Zhang, N., Yu, X., Mao, T., & Wang, L. (2023). Enhancing Sentiment Analysis with Word2Vec and LSTM: A Comparative Study. Journal of Basic and Applied Research International, 29(3), 1–10.*

*[4] Liu, R. (2024). Exploring the Impact of Word2Vec Embeddings Across Neural Network Architectures for Sentiment Analysis. Applied and Computational Engineering, 94, 106–111.*

*[5] Das, M., Selvakumar, K., & Alphonse, P. J. A. (2023). A Comparative Study on TF-IDF Feature Weighting Method and its Analysis Using Unstructured Dataset. arXiv preprint arXiv:2308.04037.*

*[6] Putri, K. K., & Setiawan, E. B. (2025). Depression Detection in Indonesian X Social Media Text using Convolutional Neural Networks and Long Short-Term Memory with TF-IDF and FastText Methods. Jurnal Teknik Informatika (JUTIF), 6(2), 557–574.*

*[7] Saif, W. Q. A., Alshammari, M. K., Mohammed, B. A., & Sallam, A. A. (2024). Enhancing Emotion Detection in Textual Data: A Comparative Analysis of Machine Learning Models and Feature Extraction Techniques. Engineering, Technology & Applied Science Research, 14(5), 16471–16477.*

*[8] Cahyani, D. E., & Patasik, I. (2021). Performance Comparison of TF-IDF and Word2Vec Models for Emotion Text Classification. Bulletin of Electrical Engineering and Informatics, 10(5), 2632–2639.*