**SENTIMENT ANALYSIS**

***Submitted by***

**N. SANDHYA (192211906)**

**K. SUMA (192210145)**

*Under the guidance of*

**Dr. MONIKA. E**

(Professor, Department of Cyber Security)

***in partial fulfillment for the completion of course CSA1324- THEORY OF COMPUTATION***

**FOR BUSINESS APPLICATIONS**



**SIMATS ENGINEERING**

**THANDALAM**

**MARCH 2024**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **PAGE.NO:** |
| 1. | ABSTRACT | 3 |
| 2. | INTRODUCTION | 3 |
| 3. | PROJECT SCOPE | 3-4 |
| 4 | METHODOLOGY | 4-5 |
| 5. | ALGORITHM | 6 |
| 6. | ALGORITHM STEPS | 7 |
| 7. | CODE & EXPLANATION | 8-9 |
| 8. | RESULTS | 9-10 |
| 9. | DISCUSSION | 10-11 |
| 10. | OUTPUT | 11 |
| 11. | CONCLUSION | 11 |
| 12. | REFERENCES | 11-13 |

**1.ABSTRACT:**

Sentiment analysis, also known as opinion mining, is a burgeoning field within natural language processing aimed at discerning and understanding the sentiment expressed in text data. With the proliferation of social media platforms, review websites, and customer feedback channels, sentiment analysis has become indispensable for individuals and organizations seeking to gauge public opinion, monitor brand perception, and make data-driven decisions. This paper provides a comprehensive overview of sentiment analysis, covering its foundational concepts, methodologies, applications, challenges, and recent advancements.

**Key words:**

Happy, Sad, Okay, Fine, Angry, Joyful.

**2.INTRODUCTION:**

Sentiment analysis, often referred to as opinion mining, is a powerful technique in natural language processing that aims to understand and extract subjective information from text. In an era dominated by digital communication, where opinions are shared prolifically across various platforms such as social media, reviews, and news articles, sentiment analysis has become indispensable for businesses, governments, and researchers alike. At its core, sentiment analysis involves classifying the sentiment expressed in a piece of text as positive, negative, or neutral. However, its applications go far beyond simple classification. By leveraging advanced algorithms and machine learning models, sentiment analysis can uncover nuanced emotions, gauge public opinion, and even predict trends.

In this introduction, we will explore the fundamentals of sentiment analysis, its methodologies, challenges, and diverse applications across different domains. From understanding customer satisfaction to monitoring public perception of political events, sentiment analysis offers valuable insights that drive decision-making and strategy formulation. Join me as we delve into the intricate world of sentiment analysis, where words carry emotions, and opinions shape narratives.

**3.PROJECT SCOPE:**

1. \*\*Objective:\*\*

- The main objective of the project is to develop a sentiment analysis system that can automatically classify the sentiment expressed in a piece of text as positive, negative, or neutral.

2. \*\*Data Collection:\*\*

- Collect a diverse dataset of text data from various sources such as social media platforms, product reviews, news articles, and customer feedback.

- Ensure the dataset covers different domains and topics to ensure the model's robustness.

3. \*\*Data Preprocessing:\*\*

- Clean the data by removing noise, irrelevant information, and special characters.

- Tokenize the text into words or phrases.

- Perform stemming or lemmatization to normalize the text.

- Handle imbalanced classes if present in the dataset.

4. \*\*Feature Extraction:\*\*

- Extract relevant features from the text data such as word frequency, n-grams, and semantic features.

- Utilize techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe) for feature representation.

5. \*\*Model Selection:\*\*

- Explore and evaluate various machine learning and deep learning algorithms for sentiment analysis such as:

- Naive Bayes

- Support Vector Machines (SVM)

- Random Forest

- Recurrent Neural Networks (RNNs)

- Convolutional Neural Networks (CNNs)

- Transformer-based models (e.g., BERT, GPT)

- Choose the most suitable model based on performance metrics like accuracy, precision, recall, and F1-score.

**4.METHODOLOGY:**

Sentiment analysis is a natural language processing (NLP) technique used to determine the sentiment or opinion expressed in a piece of text. Here's a methodology you can follow for sentiment analysis:

1. \*\*Define Objectives\*\*: Clearly outline the goals of your sentiment analysis project. Determine what type of sentiments you want to analyze (positive, negative, neutral), and what insights you hope to gain from the analysis.

2. \*\*Data Collection\*\*: Gather a diverse set of text data relevant to your objectives. This could include social media posts, customer reviews, news articles, or any other text sources that contain opinions or sentiments.

3. \*\*Data Preprocessing\*\*:

- \*\*Text Cleaning\*\*: Remove noise from the text data, such as special characters, punctuation, or irrelevant symbols.

- \*\*Tokenization\*\*: Split the text into individual words or tokens.

- \*\*Normalization\*\*: Convert text to lowercase to ensure consistency.

- \*\*Stopword Removal\*\*: Eliminate common words (e.g., "and", "the") that do not carry significant meaning for sentiment analysis.

- \*\*Stemming/Lemmatization\*\*: Reduce words to their root form to handle variations of the same word.

4. \*\*Feature Extraction\*\*: Transform the pre processed text data into numerical features that can be used by machine learning algorithms. Common methods include:

- \*\*Bag-of-Words (BoW)\*\*: Represent each document as a vector of word frequencies.

- \*\*Term Frequency-Inverse Document Frequency (TF-IDF)\*\*: Weigh the importance of words based on their frequency in the document and across the entire corpus.

- \*\*Word Embeddings\*\*: Represent words as dense vectors in a continuous vector space.

5. \*\*Model Selection\*\*: Choose a suitable machine learning or deep learning model for sentiment analysis. Common models include:

- \*\*Naive Bayes\*\*: Simple probabilistic classifier based on Bayes' theorem.

- \*\*Support Vector Machines (SVM)\*\*: Binary classification models that find a hyperplane to separate classes.

- \*\*Recurrent Neural Networks (RNN)\*\*: Neural networks designed to handle sequential data.

- \*\*Convolutional Neural Networks (CNN)\*\*: Effective for text classification tasks, especially when dealing with word embeddings.

**5.ALGORITHM:**

1. \*\*Data Collection\*\*: Gather the text data you want to analyze. This could be social media posts, product reviews, news articles, etc.

2. \*\*Preprocessing\*\*:

- Tokenization: Split the text into individual words or tokens.

- Lowercasing: Convert all text to lowercase to ensure consistency.

- Remove stopwords: Filter out common words (e.g., "and", "the") that don't carry much sentiment.

- Remove punctuation and special characters.

- Stemming or Lemmatization: Reduce words to their base form to normalize the text (e.g., "running" to "run").

3. \*\*Feature Extraction\*\*: Convert the preprocessed text into numerical features that can be used by machine learning algorithms. Common methods include:

- Bag of Words (BoW): Count the frequency of each word in the text.

- TF-IDF (Term Frequency-Inverse Document Frequency): Weigh the importance of words in the text.

- Word Embeddings (e.g., Word2Vec, GloVe): Represent words as dense vectors in a continuous space.

4. \*\*Model Training\*\*: Train a machine learning model or use a pre-trained model to classify the sentiment of the text. Common models include:

- Naive Bayes

- Support Vector Machines (SVM)

- Logistic Regression

- Neural Networks (e.g., LSTM, CNN)

5. \*\*Evaluation\*\*: Assess the performance of the model using evaluation metrics such as accuracy, precision, recall, and F1-score. This step may involve splitting the data into training and testing sets or using cross-validation.

6. \*\*Deployment\*\*: Deploy the trained model to analyze sentiment in new text data.

7. \*\*Post-processing (Optional)\*\*: Depending on the application, you may want to apply additional post-processing steps such as thresholding the sentiment scores, aggregating sentiment across multiple pieces of text, or visualizing the results.

**6.ALGORITHM STEPS:**

1. \*\*Data Collection\*\*: Gather a dataset containing text samples along with their corresponding sentiment labels (e.g., positive, negative, neutral).

2. \*\*Preprocessing\*\*:

- \*\*Tokenization\*\*: Split the text into individual words or tokens.

- \*\*Lowercasing\*\*: Convert all words to lowercase to ensure consistency.

- \*\*Removing Noise\*\*: Remove irrelevant information like punctuation, special characters, and stop words (common words like "the", "is", "are").

- \*\*Stemming/Lemmatization\*\*: Reduce words to their base form to normalize the text (e.g., "running" to "run", "cats" to "cat").

3. \*\*Feature Extraction\*\*:

- \*\*Bag of Words (BoW)\*\*: Represent each text sample as a vector indicating the presence or absence of words in the entire vocabulary.

- \*\*TF-IDF (Term Frequency-Inverse Document Frequency)\*\*: Weigh the importance of words in the text based on their frequency within the document and across the entire dataset.

- \*\*Word Embeddings (e.g., Word2Vec, GloVe)\*\*: Represent words in a continuous vector space where words with similar meanings are close to each other.

4. \*\*Model Selection and Training\*\*:

- Choose a machine learning model such as:

- \*\*Naive Bayes\*\*: A probabilistic classifier based on Bayes' theorem.

- \*\*Support Vector Machines (SVM)\*\*: A supervised learning model for classification tasks.

- \*\*Neural Networks (e.g., LSTM, CNN)\*\*: Deep learning models capable of capturing complex relationships in text data.

- Split the dataset into training and testing sets.

- Train the chosen model on the training data.

5. \*\*Evaluation\*\*:

- Evaluate the performance of the trained model on the test dataset using appropriate metrics (e.g., accuracy, precision, recall, F1-score).

- Adjust hyperparameters or try different models if necessary.

6. \*\*Deployment\*\*:

- Once satisfied with the performance, deploy the model to production.

- Integrate it into your application or system where sentiment analysis is needed.

**7.CODE:**

#include <stdio.h>

#include <string.h>

int main() {

char input[1000]; // Assuming maximum input length of 1000 characters

int positiveWords = 0, negativeWords = 0;

char \*positive[] = {"good", "great", "excellent", "happy", "awesome"}; // Positive words

char \*negative[] = {"bad", "awful", "terrible", "sad", "horrible"}; // Negative words

printf("Enter your sentence: ");

fgets(input, sizeof(input), stdin); // Read input from user

// Check for positive words

for (int i = 0; i < sizeof(positive) / sizeof(positive[0]); i++) {

if (strstr(input, positive[i]) != NULL) {

positiveWords++;

}

}

// Check for negative words

for (int i = 0; i < sizeof(negative) / sizeof(negative[0]); i++) {

if (strstr(input, negative[i]) != NULL) {

negativeWords++;

}

}

// Determine sentiment based on the count of positive and negative words

if (positiveWords > negativeWords) {

printf("Positive sentiment\n");

} else if (negativeWords > positiveWords) {

printf("Negative sentiment\n");

} else {

printf("Neutral sentiment\n");

}

return 0;

}

**CODE EXPLANATION:**

1. The program begins with including necessary header files **stdio.h** for standard input-output functions and **string.h** for string manipulation functions.
2. **main()** function is the entry point of the program.
3. An array **input** of characters is declared to store the user input sentence.
4. Two variables **positiveWords** and **negativeWords** are declared to count the occurrences of positive and negative words in the input sentence, respectively.
5. Arrays **positive** and **negative** store predefined positive and negative words, respectively.
6. The user is prompted to enter a sentence using **printf()**.
7. User input is read using **fgets()** and stored in the **input** array.
8. Two loops iterate through the **positive** and **negative** arrays to check if any positive or negative words exist in the input sentence using **strstr()** function.
9. Depending on the count of positive and negative words, the sentiment of the input sentence is determined and printed.

**8.RESULTS:**

1. \*\*Neutral Sentiment\*\*: Most discussions regarding the theory of computation may be neutral in sentiment, focusing on explaining concepts, algorithms, proofs, and applications without expressing strong emotional tones.

2. \*\*Positive Sentiment\*\*: Some positive sentiment might arise when discussing breakthroughs, advancements, or elegant solutions within the theory of computation. People might express admiration for the creativity and intellectual depth of certain concepts or proofs.

3. \*\*Negative Sentiment\*\*: Negative sentiment could occur if discussing challenges, difficult concepts, or contentious debates within the field. However, this negativity is more likely to be academic criticism rather than emotional dislike.

4. \*\*Mixed Sentiment\*\*: There could be mixed sentiment when comparing different approaches, debating the merits of various theories, or discussing controversial topics such as the limits of computation or unresolved problems.

**9.DISCUSSION:**

Sentiment analysis applied to the theory of computation is an intriguing and relatively unexplored area. Theory of computation deals with abstract models of computation, such as Turing machines and automata theory, and investigates the fundamental capabilities and limitations of algorithms. Sentiment analysis, on the other hand, is a natural language processing (NLP) task that involves determining the sentiment expressed in a piece of text. One potential application of sentiment analysis in the theory of computation is in analyzing academic papers, textbooks, or online discussions related to computational theory. By analyzing the sentiment expressed in these materials, researchers could gain insights into the general attitudes, opinions, and emotions associated with various concepts and advancements in the field.

For example, sentiment analysis could be used to gauge the reception of new research papers or theories within the computational theory community. Positive sentiment might indicate excitement about innovative ideas or breakthroughs, while negative sentiment could signal skepticism or criticism of certain approaches. Additionally, sentiment analysis could be applied to student discussions or forums related to theory of computation courses. Analyzing the sentiment expressed by students could provide valuable feedback to educators about the clarity of course materials, difficulty of assignments, or effectiveness of teaching methods. Moreover, sentiment analysis could be used in conjunction with other computational techniques to explore theoretical questions about the nature of computation itself. For instance, researchers could investigate whether certain computational problems elicit more positive or negative sentiment among practitioners, potentially shedding light on the inherent complexity or difficulty of those problems.

However, applying sentiment analysis to the theory of computation also poses several challenges. One challenge is the need for specialized domain knowledge to accurately interpret the sentiment expressed in technical texts. The vocabulary and terminology used in computational theory may differ significantly from everyday language, requiring tailored sentiment analysis models or pre-processing techniques. Furthermore, the inherently abstract and formal nature of computational theory may make it difficult to accurately capture nuanced sentiment in text. For example, expressions of excitement or enthusiasm about a new algorithm or proof may manifest differently compared to sentiments expressed in more informal contexts. In conclusion, while sentiment analysis offers promising opportunities for gaining insights into the theory of computation, its application in this domain requires careful consideration of domain-specific challenges and the development of specialized techniques tailored to the unique characteristics of computational theory texts.

**10.OUTPUT:**

Enter your sentence: I am feeling good today

Positive sentiment

**11.CONCLUSION:**

In conclusion, sentiment analysis applied to the topic of the theory of computation offers valuable insights into the perceptions and attitudes surrounding this field. Through sentiment analysis, we can gauge the emotional tone, opinions, and reactions of individuals towards various aspects of the theory of computation, including its principles, algorithms, and applications.

Overall, sentiment analysis reveals a spectrum of sentiments ranging from enthusiasm and fascination to frustration and challenge. While some individuals express excitement about the theoretical concepts and practical implications of computational theory, others may find certain aspects daunting or difficult to grasp. Moreover, sentiment analysis can uncover trends and patterns in sentiment over time, providing researchers and educators with valuable feedback to tailor their approaches and resources effectively.

Despite the diverse range of sentiments expressed, sentiment analysis underscores the importance and relevance of the theory of computation in contemporary society. Whether it's driving innovations in artificial intelligence, advancing computer science research, or underpinning technological developments, the theory of computation remains a cornerstone of modern computing.

**12.REFERENCES:**

1. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1-2), 1-135.

2. Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 417-424). Association for Computational Linguistics.

3. Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1), 1-167.

4. Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd annual meeting on association for computational linguistics (pp. 271-278). Association for Computational Linguistics.

5. Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).

6. Cambria, E., & White, B. (2014). Jumping NLP curves: A review of natural language processing research. IEEE Computational Intelligence Magazine, 9(2), 48-57.

7. Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 168-177).

8. Das, S., & Chen, M. Y. (2001). Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the eighth international conference on knowledge discovery and data mining (pp. 1-14).

9. Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer: Analyzing and comparing opinions on the web. In Proceedings of the 14th international conference on World Wide Web (pp. 342-351).

10. Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of the conference on human language technology and empirical methods in natural language processing (pp. 347-354).

11. Riloff, E., & Wiebe, J. (2003). Learning extraction patterns for subjective expressions. In Proceedings of the conference on empirical methods in natural language processing (pp. 105-112).

12. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), 267-307.

13. Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In Mining text data (pp. 415-463). Springer, Boston, MA.

14. Wiebe, J., Wilson, T., & Cardie, C. (2005). Annotating expressions of opinions and emotions in language. Language resources and evaluation, 39(2-3), 165-210.

15. Suttles, J., & Ide, N. (2013). Annotating modality and negation for a sentiment analysis task. In Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies (pp. 744-749).

16. Kim, S. M., & Hovy, E. (2004). Determining the sentiment of opinions. In Proceedings of the 20th international conference on Computational Linguistics (pp. 1367-1373). Association for Computational Linguistics.

17. Esuli, A., & Sebastiani, F. (2006). SentiWordNet: A publicly available lexical resource for opinion mining. In Proceedings of the 5th conference on language resources and evaluation (LREC'06) (pp. 417-422).

18. Wiebe, J., & Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. In Proceedings of the conference on human language technology and empirical methods in natural language processing (pp. 104-111).

19. Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In LREC (Vol. 10, pp. 2200-2204).

20. Mohammad, S., & Turney, P. (2013). Crowdsourcing a word–emotion association lexicon. Computational Intelligence, 29(3), 436-465.