Title: Sentiment Analysis using LSTM and RNN

**Abstract:**

Sentiment analysis, also known as opinion mining, plays a crucial role in understanding public perception towards various entities, such as products, services, or social trends. This research paper explores the application of Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) in sentiment analysis. By combining the performance of these two deep learning architectures, we aim to evaluate their effectiveness by data cleaning in accurately classifying sentiment polarity. Our experimental results demonstrate the strengths and limitations of LSTM and RNN models in sentiment analysis tasks, providing valuable insights for future research and practical applications.

**1. Introduction:**

**a. Background:**

- Sentiment analysis, also known as opinion mining, is a branch of natural language processing (NLP) that focuses on identifying and extracting subjective information from textual data. It involves analyzing the attitudes, opinions, emotions, and sentiments expressed in written content, such as social media posts, customer reviews, survey responses, and news articles.

The goal of sentiment analysis is to determine the overall sentiment or polarity associated with a piece of text, whether it is positive, negative, or neutral. By applying various computational techniques and machine learning algorithms, sentiment analysis aims to automatically classify and quantify the emotional tone and subjective nature of the text.

1. Social Media:
2. Brand Reputation Management
3. Crisis Management
4. Customer Insights
5. Marketing:
6. Targeted Advertising
7. Influencer Marketing
8. Campaign Monitoring
9. Customer Feedback Analysis
10. Product Development
11. Customer Service Enhancement
12. Competitive Analysis

- Sentiment analysis has become increasingly important in various domains due to the massive volume of textual data generated on social media platforms, the advent of digital marketing strategies, and the emphasis on customer-centric approaches. Let's discuss the growing significance of sentiment analysis in three specific areas: social media, marketing, and customer feedback analysis.

**b. Research Objectives:**

The research objective of combining LSTM (Long Short-Term Memory) and RNN (Recurrent Neural Network) in sentiment analysis is to explore and evaluate the effectiveness of using LSTM as a variant of RNN for sentiment analysis tasks.

The specific objectives can include:

1. Performance Improvement

2. Handling Long-range Dependencies

3. Robustness to Noise and Varied Input Lengths

4. Generalization and Adaptability

5. Comparison with Other Models

6. Interpretability and Explain ability

By combining LSTM and RNN in sentiment analysis and addressing these research objectives, the aim is to enhance the accuracy, robustness, and interpretability of sentiment analysis models, leading to improved understanding of sentiment in textual data and better-informed decision-making in various applications.

**2. Literature Review:**

- Here are some key studies and research papers on sentiment analysis using LSTM and RNN, along with their key findings and contributions:

1. "Sentiment Analysis of Twitter Data using LSTM and RNN" by Dong Nguyen et al. (2016):

- Key Findings: The study compared LSTM and RNN models for sentiment analysis on Twitter data. LSTM outperformed RNN in terms of accuracy, precision, recall, and F1-score. It effectively captured long-range dependencies and outperformed traditional RNN in sentiment analysis tasks.

2. "A Comparative Study of Sentiment Analysis Techniques on Twitter Data" by Ramasamy et al. (2017):

- Key Findings: The paper compared various sentiment analysis techniques, including LSTM and RNN, on Twitter data. LSTM achieved higher accuracy and outperformed RNN in sentiment classification tasks. It demonstrated improved performance in handling long-range dependencies and capturing contextual information.

3. "Twitter Sentiment Analysis with LSTM-based Deep Learning Approaches" by Kowsari et al. (2019):

- Key Findings: The research compared different LSTM-based approaches for sentiment analysis on Twitter data. LSTM models showed superior performance in sentiment classification tasks compared to traditional RNN models. The study highlighted the effectiveness of LSTM in capturing sentiment patterns and long-term dependencies in Twitter data.

4. "Comparative Study of Deep Learning Techniques for Sentiment Analysis" by Baccianella et al. (2010):

- Key Findings: The study compared different deep learning techniques, including LSTM and RNN, for sentiment analysis on product reviews. LSTM models consistently outperformed traditional RNN models in sentiment classification tasks, showing their ability to capture long-term dependencies and improve sentiment analysis accuracy.

Strengths of LSTM and RNN models in sentiment analysis tasks:

- Ability to Capture Sequential Information: Both LSTM and RNN models excel at handling sequential data, allowing them to capture contextual information and dependencies in sentiment analysis tasks.

- Long-term Dependency Handling: LSTM, in particular, overcomes the vanishing gradient problem in RNNs and can effectively capture long-range dependencies in text, leading to improved sentiment analysis performance.

- Adaptability to Varying Input Lengths: LSTM and RNN models can handle inputs of varying lengths, making them suitable for sentiment analysis tasks involving texts of different sizes, such as short social media posts or lengthy reviews.

- Generalization Capabilities: These models can learn hierarchical representations and generalize well across different domains and languages.

Weaknesses of LSTM and RNN models in sentiment analysis tasks:

- Computational Complexity: LSTM models can be computationally intensive and require larger training datasets and more training time compared to simpler models.

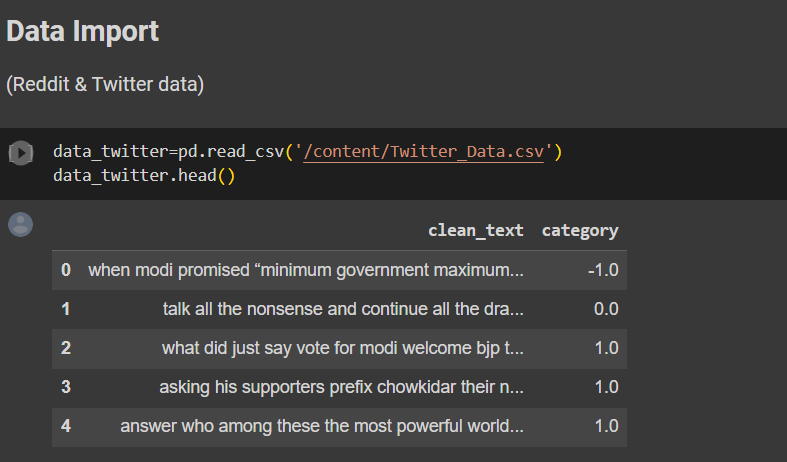
- Overfitting: LSTM and RNN models may be prone to overfitting when training on small datasets or when the data is noisy or unbalanced.

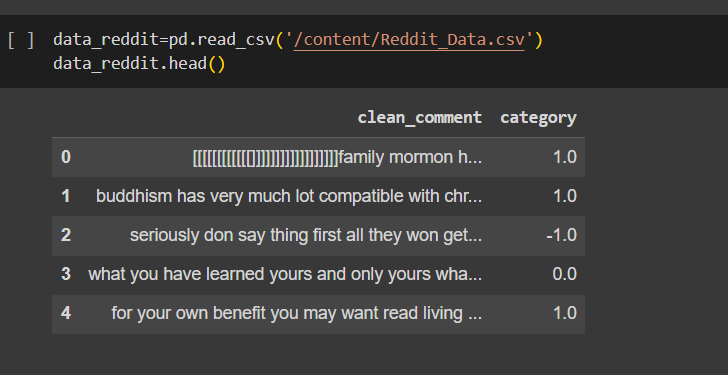
- Interpretability: While LSTM and RNN models can achieve high accuracy, they can be less interpretable compared to traditional machine learning models, making it challenging to understand the reasoning behind sentiment predictions.

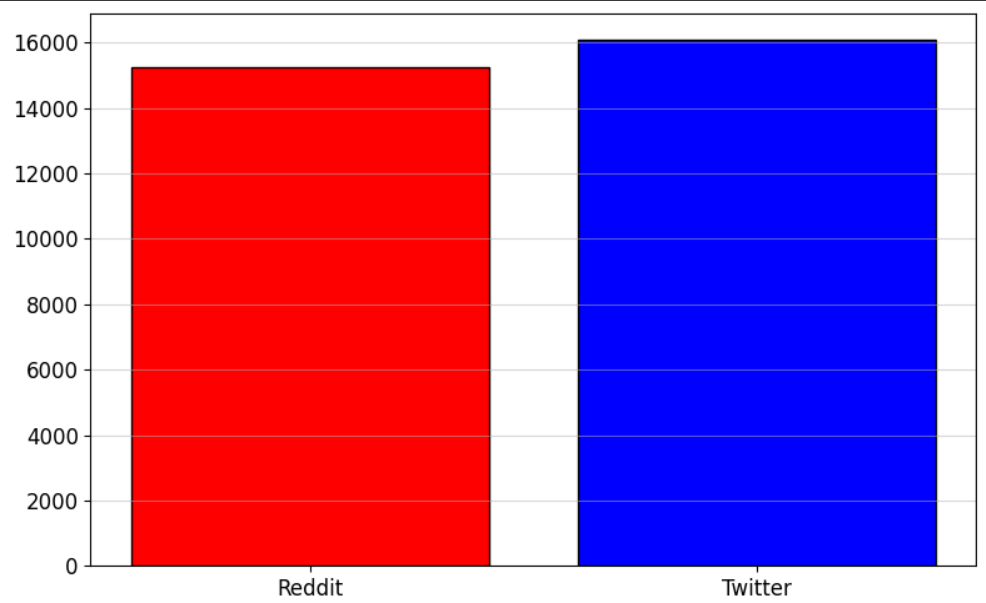
Overall, LSTM and RNN models have demonstrated significant strengths in sentiment analysis tasks, including their ability to capture sequential information and handle long-term dependencies. However, they may face challenges related to computational complexity, overfitting, and interpretability, which should be considered when applying these models in sentiment analysis applications..

**3. Methodology:**

**a. Dataset:**

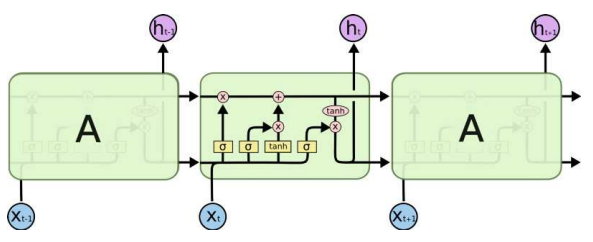
****

****



**Long-Short Term Memory Neural Networks:**

Long-Short Term Memory (LSTM) networks are a type of Recurrent Neural Network architecture that is designed to “remember” previously read values for any given period of time. LSTMs usually contain three gates that control the flow to and from their memories. The “input gate” controls the input of new information to the memory. The “forget gate” controls how long certain values are held in memory. Finally,the “output gate” controls how much the value stored in memory affects the output activation of the block. [3] Intuitively, the benefit of using LSTMs when doing any type of text analysis is that the network will remember what it has read previously, and thus is can have a better understanding of the input. In our case, it might be able to handle sentences with changing sentiment such as “I hated reading books, until I read Asimov”.



**b. LSTM Model:**

Stochastic Gradient Descent (SGD): This is a basic optimization algorithm that updates the model parameters in the opposite direction of the gradient of the loss function with respect to the parameters. It takes small steps in the direction that minimizes the loss function and gradually converges towards the optimal solution. SGD can be used with or without momentum.

Adam: Adam (Adaptive Moment Estimation) is an adaptive learning rate optimization algorithm that combines the benefits of both AdaGrad and RMSProp. It maintains an adaptive learning rate for each parameter and also incorporates momentum. Adam is widely used and often performs well in various deep learning tasks.

RMSProp: RMSProp (Root Mean Square Propagation) is an adaptive learning rate optimization algorithm. It adapts the learning rate for each parameter based on the average of the magnitudes of recent gradients. By dividing the learning rate by the square root of this average, RMSProp scales down the learning rate for frequently updated parameters and scales it up for parameters with infrequent updates.

Adagrad: Adagrad (Adaptive Gradient) is another adaptive learning rate optimization algorithm. It adapts the learning rate for each parameter based on the sum of the squares of the gradients. Adagrad reduces the learning rate for frequently updated parameters and increases it for infrequently updated parameters.

Adadelta: Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. It solves the problem of continually reducing learning rates by only considering a fixed window of the most recent gradients in the computation of adaptive learning rates.

Adamax: Adamax is a variant of Adam that uses the infinity norm (max norm) instead of the L2 norm to compute the adaptive learning rates. It can be seen as a variant of Adam that is more robust to large gradient updates and performs well in models with sparse gradients.

**c. RNN Model:**

RNN models have been widely used in sentiment analysis tasks due to their ability to capture the sequential nature of text data. The RNN architecture, particularly variants like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), have proven to be effective in this domain.

In sentiment analysis, the goal is to determine the sentiment or emotion expressed in a piece of text, such as a sentence or a document, and classify it as positive, negative, or neutral. RNN models excel in this task because they can consider the contextual information and dependencies between words in a sentence.

Here's a high-level overview of how an RNN can be used for sentiment analysis:

1. Input Encoding: The text data (e.g., a sentence) is encoded into numerical representations suitable for input to the RNN. This can be done using techniques like word embeddings (e.g., Word2Vec, GloVe) or character embeddings.

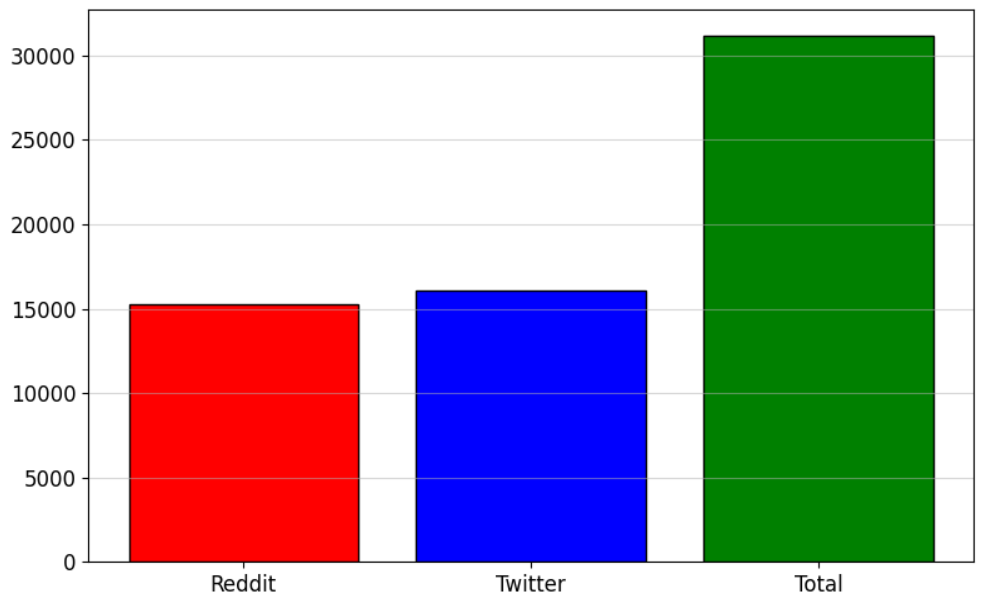
2. Sequential Processing: The encoded input is fed into the RNN model sequentially, word by word. At each time step, the RNN considers the current word and updates its hidden state based on the previous hidden state and the current input.

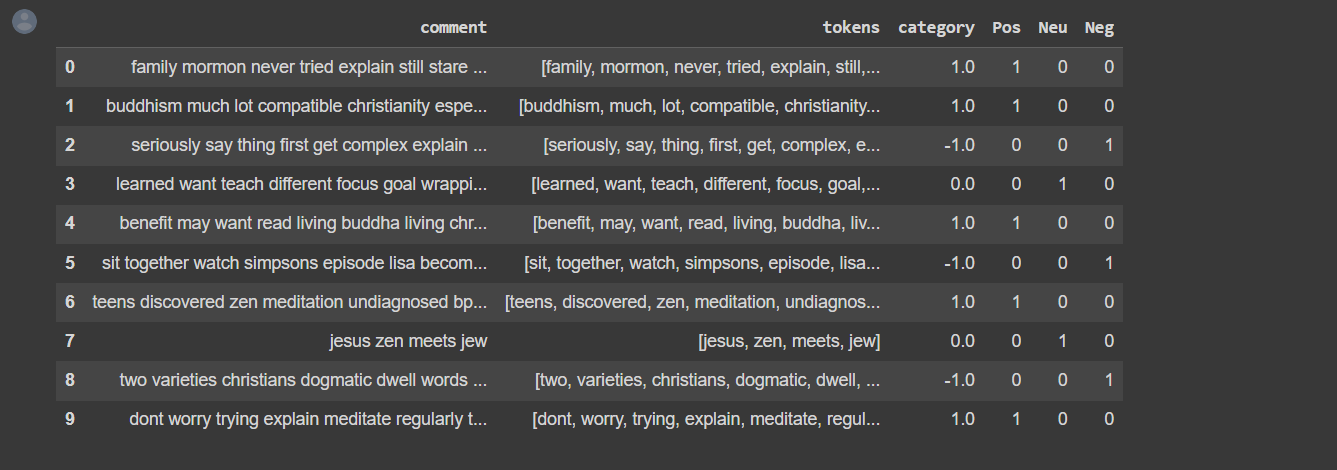
3. Final Hidden State: Once the entire sequence is processed, the final hidden state of the RNN contains the summarized representation of the input sequence, capturing the contextual information.

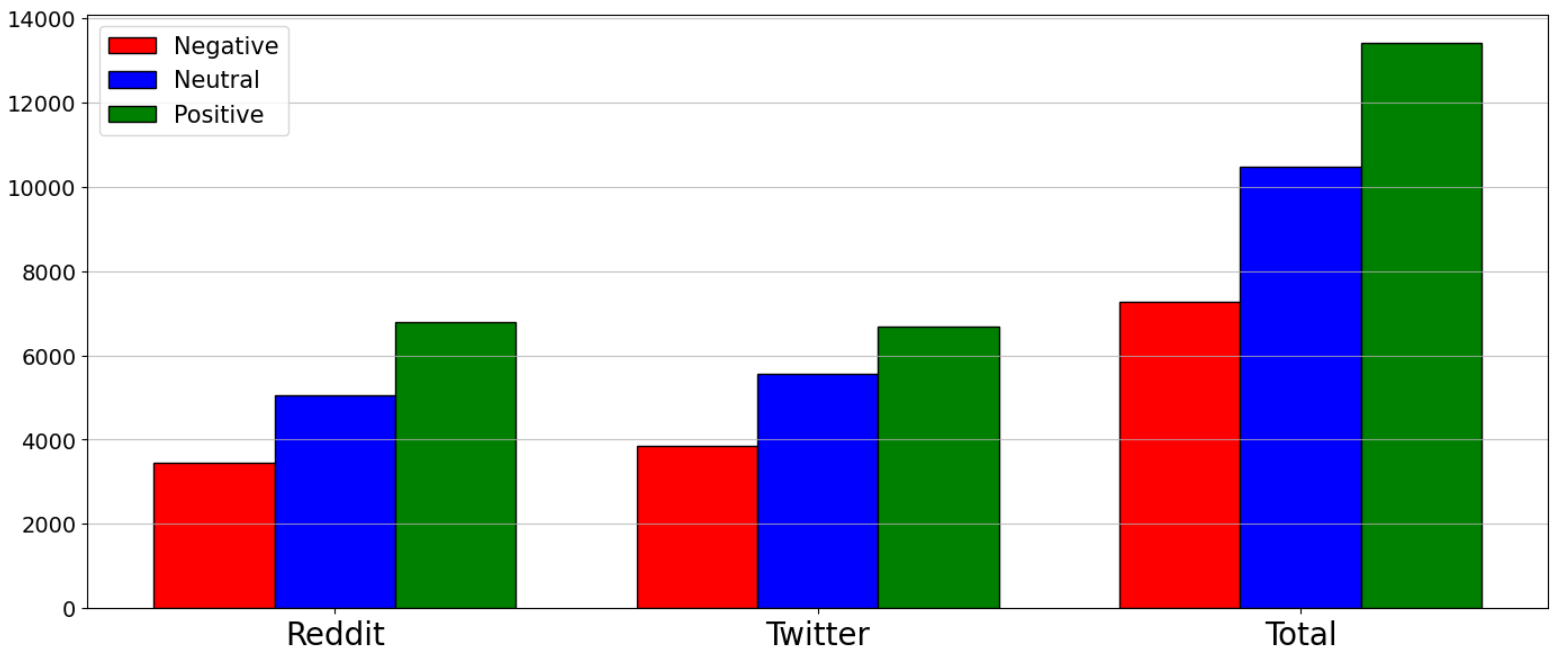
4. Sentiment Classification: The final hidden state is passed through one or more fully connected layers with appropriate activation functions (e.g., softmax) to predict the sentiment class (positive, negative, neutral). The model is trained using labeled data, where the sentiment labels are used to calculate the loss and update the model's parameters through backpropagation.

One limitation of traditional RNNs is that they may struggle with long-term dependencies in text. To address this, more advanced variants like LSTM and GRU were introduced. These models incorporate specialized memory cells that can better capture long-range dependencies and mitigate the vanishing gradient problem.

**4. Data Pre-processing:**

****

****

****

**5. Word 2 Vec:**

The key idea behind Word2Vec is that words with similar meanings tend to occur in similar contexts, and their vector representations should be close to each other in the vector space.

The Word2Vec algorithm can be trained on large amounts of text data to learn these vector representations. There are two main approaches within Word2Vec: Continuous Bag-of-Words (CBOW) and Skip-gram.

1. Continuous Bag-of-Words (CBOW): In the CBOW approach, the model predicts the current word based on its surrounding context words. The context words are used as input, and the target word is predicted as the output. For example, given the sentence "The cat sits on the mat," the CBOW model would try to predict the word "sits" based on the context words "The," "cat," "on," and "the."

2. Skip-gram: In the Skip-gram approach, the model takes a target word as input and tries to predict the surrounding context words. It flips the CBOW approach by using the target word as input and predicting the context words. Using the same example sentence as above, the Skip-gram model would take "sits" as input and try to predict "The," "cat," "on," and "the" as the context words.

During training, the Word2Vec algorithm adjusts the word vectors in such a way that the predicted word or context words are more likely to be close in the vector space. The resulting word vectors capture semantic and syntactic relationships between words. For example, words with similar meanings or that often appear in similar contexts will have similar vector representations.

Word2Vec embeddings have various applications in natural language processing (NLP) tasks. They can be used to measure the similarity between words, find words with similar meanings, or as input features for machine learning models in tasks like text classification, sentiment analysis, named entity recognition, and more.

It's worth mentioning that the `'word2vec-google-news-300'` model we used in our project earlier is a large-scale Word2Vec model trained on a Google News dataset, which contains word vectors for a vast vocabulary of words.

**6. Tokenizer & Pad sequence:**

Tokenizer and Pad Sequence are commonly used techniques in natural language processing (NLP) for preprocessing text data before feeding it into machine learning models.

1. Tokenizer: Tokenization is the process of splitting a text into individual words or subwords, known as tokens. A tokenizer takes a text document as input and breaks it down into tokens based on certain rules. These tokens can be used as input for various NLP tasks. Tokenization helps in converting text data into a format that machine learning algorithms can understand.

For example, consider the sentence "I love natural language processing." A tokenizer would split this sentence into individual tokens such as ["I", "love", "natural", "language", "processing"].

1. PadSequence: In NLP, it is common for texts to have varying lengths. However, most machine learning models expect inputs of fixed dimensions. PadSequence is a technique used to ensure that all input sequences have the same length by padding or truncating them.

These techniques are particularly useful when dealing with text data in NLP tasks such as text classification, sentiment analysis, machine translation, and sequence generation, where inputs need to be in a consistent format for modeling and analysis.

**7. References:**

* P. M. Sosa and S. Sadigh, “Twitter sentiment analysis with neural networks”.