MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Big Data Analytics
Mini Project Report on

Real Time Simulation of News Sentiment Prediction

SUBMITTED BY

Arnav Sanjay Karnik

Section B

Under the Guidance of:

Dr. Prakash Kalingrao Aithal

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal, Karnataka – 576104

October 2024

Topic 1- Abstract:

Our project compares two approaches to sentiment analysis of news headlines: traditional supervised learning methods (like Logistic Regression and Random Forest) and deep learning techniques (such as LSTM). Traditional methods rely on feature extraction like TF-IDF, while deep learning leverages word embeddings and sequence modelling to capture complex patterns in text. The analysis evaluates the performance, advantages, and limitations of each approach, highlighting the deep learning models' ability to better capture sentiment nuances while the classic methods offer simplicity and speed.

Topic 2 – Introduction:

Sentiment analysis, a subset of natural language processing (NLP), focuses on determining the emotional tone behind a body of text. It has gained significant importance in various domains, especially in analyzing public opinion through platforms like social media and news outlets. In particular, news headlines serve as critical indicators of public sentiment due to their concise and impactful nature.

The ability to classify headlines as positive, negative, or neutral provides valuable insights into prevailing sentiments regarding specific events, topics, or individuals. This capability is especially pertinent in today's fast-paced information age, where the media significantly influences public perception and decision-making.

Traditional sentiment analysis methods often rely on supervised learning techniques, which utilize labeled datasets to train models. Popular algorithms, such as Logistic Regression and Random Forest, have been widely employed for their interpretability and efficiency. However, these approaches may struggle with the complexity and subtlety of language, particularly in capturing context and nuanced meanings inherent in news headlines.

This introduction sets the stage for a comprehensive analysis of the strengths and limitations of both traditional and deep learning methods in the context of sentiment analysis on news headlines. By examining these approaches, we can better understand their applicability and effectiveness in interpreting public sentiment, ultimately contributing to more informed decision-making in various sectors, from marketing to politics.

Topic 3 – Literature Review:

Sentiment analysis has become a crucial area of research due to its applications in various fields, including finance, marketing, and social media monitoring. News headlines, in particular, play a significant role in shaping public perception, making their analysis vital for understanding trends and sentiments in society.

Traditional Supervised Learning Approaches:

Traditional supervised learning methods, such as Logistic Regression, Support Vector Machines (SVM), and Random Forests, have been widely used for sentiment classification tasks. These methods typically rely on a bag-of-words model or term frequency-inverse document frequency (TF-IDF) to extract features from the text. Studies have shown that Logistic Regression is effective for sentiment analysis due to its simplicity and interpretability (Pang & Lee, 2008).

Logistic Regression is a fundamental statistical method widely utilized for binary classification tasks, including sentiment analysis. It operates by modeling the probability that a given input belongs to a particular category (e.g., positive or negative sentiment).

Key Features of Logistic Regression in Sentiment Analysis:

1. **Probabilistic Framework**: Logistic Regression calculates the probability of the outcome based on a linear combination of input features. The output is transformed using the logistic function, producing a value between 0 and 1, which can be interpreted as a probability.
2. **Feature Extraction**: To analyze sentiment, features are often extracted from text using methods like Bag-of-Words or Term Frequency-Inverse Document Frequency (TF-IDF). These techniques convert textual data into numerical representations that the logistic regression model can process.
3. **Interpretability**: One of the significant advantages of Logistic Regression is its interpretability. Coefficients assigned to features indicate

the strength and direction of their influence on the outcome, making it easier for analysts to understand what drives sentiment.

4. **Performance**: Research has shown that Logistic Regression performs competitively in sentiment classification tasks, particularly when the data is well-prepared and features are appropriately selected. Its efficiency and low computational requirements make it suitable for applications with large datasets.
5. **Limitations**: Despite its strengths, Logistic Regression has limitations, particularly with non-linear relationships. In complex sentiment analysis tasks, particularly those involving nuanced expressions or multiple sentiments, more advanced models like Support Vector Machines or deep learning techniques may outperform it.

Challenges in Sentiment Analysis-

1. Data Scarcity: Developing labeled datasets is often challenging, particularly for specific domains or languages, limiting model training.

2. Bias in Training Data: Sentiment models can inherit biases present in the training data, leading to skewed predictions and perpetuating stereotypes.

3. Complexity of Language: Sentiment can be context-dependent, involving sarcasm, irony, or nuanced expressions, which traditional models may struggle to interpret accurately.

4. Feature Selection: Identifying relevant features from text for effective classification remains a complex task.

Future Directions -

1. Hybrid Models: Research could focus on integrating traditional statistical methods with deep learning techniques to balance interpretability and performance.

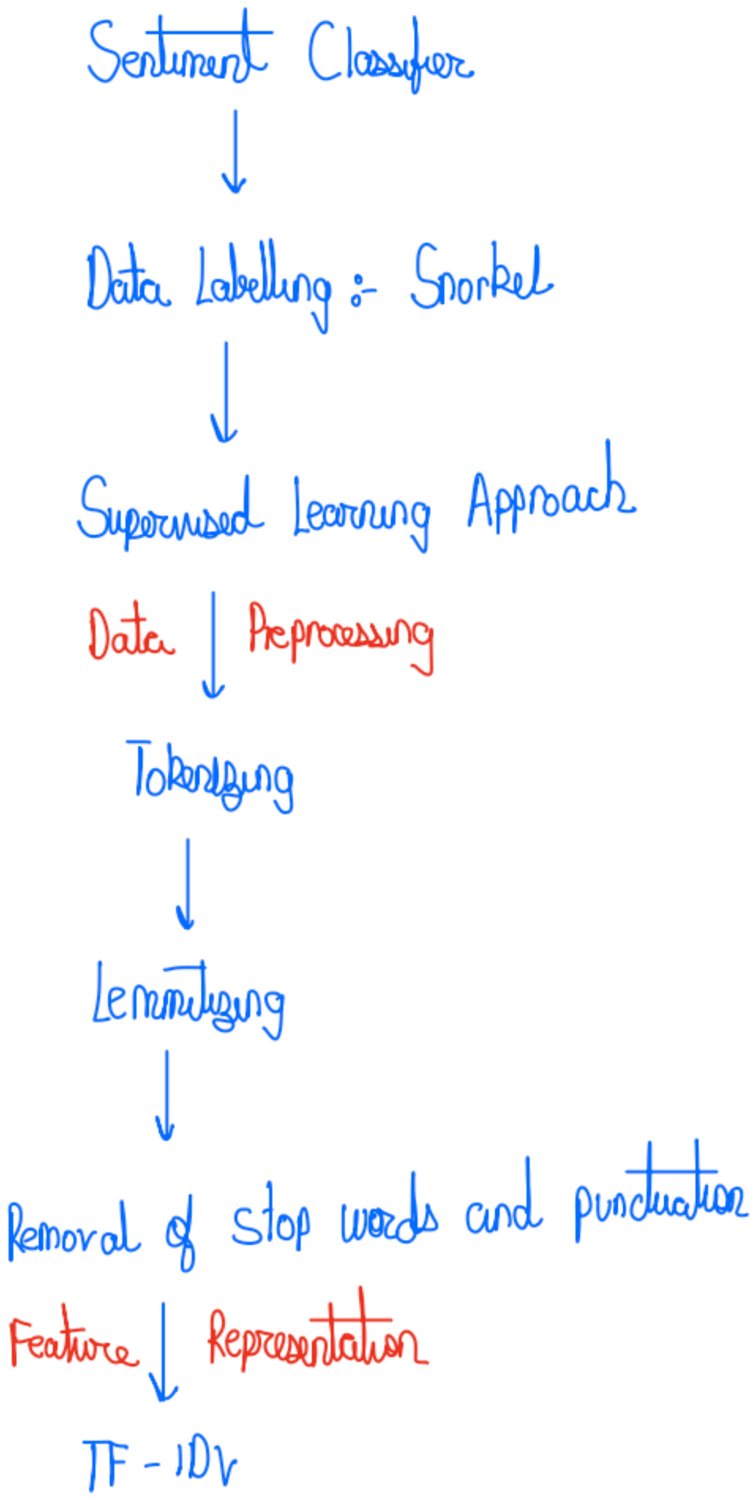
2. Unsupervised Learning: Exploring unsupervised and semi-supervised approaches can help mitigate the data scarcity issue, allowing models to learn from unannotated data.

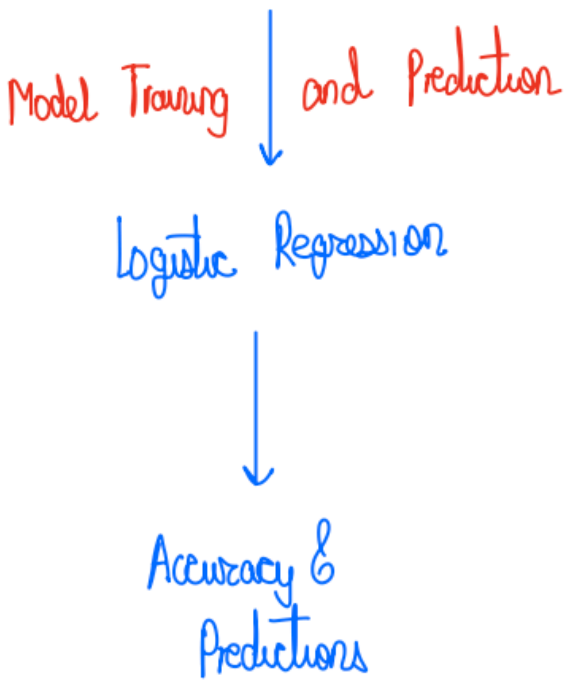
3. Incorporating External Factors: Future models might benefit from considering real-time events, user demographics, and contextual information to enhance sentiment understanding.

4. Adapting to New Languages and Domains: Developing models that adapt to different languages and domain-specific language will be crucial as sentiment analysis expands to various applications.

5. Mitigating Bias: Ongoing research into bias mitigation strategies is essential for creating fair and equitable sentiment analysis models.

Topic 4 – Methodology :


Sentiment Classifier
Data Labelling :- Snorkel
Supervised Learning Approach
Data Preprocessing
Tokenizing
Lemmatizing
Removal of stop words and punctuation
Feature Representation
TF - IDV



Step 1 – Web-scrapping for extracting the data:

1. Setup and Libraries: You import necessary libraries like JSoup (for parsing HTML) and define variables for the target website URL and the specific CSS selectors you'll use to extract data.

2. HTTP Requests: You send an HTTP request to the website to retrieve its HTML content. This can involve handling session cookies or authentication if required.

3. Parsing HTML: Once you receive the HTML response, you use JSoup to parse the document. You then select elements using CSS selectors to target specific data points, like product names or descriptions.

4. Data Extraction: You extract the required information from the selected elements and store it in a structured format (like lists or maps) for further processing or saving to a file.

5. Handling Multiple Pages: If scraping multiple pages, you may loop through pagination links, repeating the request and parsing process until you've gathered all the desired data.

Step 2 – Kafka Producer (Data Streaming):

1.) Kafka Producer Setup:

- The `KafkaProducer` from the `kafka-python` library is initialized to send data to a Kafka topic.
- The `bootstrap_servers` parameter specifies the Kafka server address (`localhost:9092`). This is the communication endpoint to send messages.
- The `value_serializer` parameter serializes Python objects (dictionaries) into JSON strings, so that Kafka can send them as byte streams.

2.) Data Source (CSV file):

- The data is read from a CSV file using `pandas.read_csv()`. This CSV contains article data scraped from Economic Times or similar sources.
- Each row in the DataFrame corresponds to a separate article, with fields such as `title`, `content`, `date`, etc.

3.) Iterating and Sending Data:

- The program iterates through each row of the DataFrame using a for-loop (`for _, row in df.iterrows()`).
- Each row is converted into a dictionary (`row.to_dict()`) and sent to the Kafka topic using `producer.send()`.
- After sending, a `time.sleep(3)` adds a delay of 3 seconds between each message to simulate real-time streaming of data, emulating a live data feed.

4.) Closing the Producer:

- Once all rows are sent, the producer flushes any unsent messages and closes the connection to Kafka (`producer.flush()` and `producer.close()`).

Parameters used in the producer:

- `csv_file`: The path to the CSV containing the article data. This can be replaced by any dynamically updated file that your scraper outputs.

- `kafka_topic`: The Kafka topic (`sentiment_analysis`) where the messages are sent. In real-world use, the topic might be named `et_articles` to reflect the data's source.

- `bootstrap_servers`: The address of the Kafka broker, typically `localhost:9092` for local setups.

Iterative Data Streaming:

This producer sends data iteratively, row by row. By using scheduling tools like **cron** or Python schedulers, this process can be automated to run periodically (e.g., daily), making it look like real-time data ingestion.

Step 3 – Kafka Consumer (Data Retrieval):

1.) Kafka Consumer Setup:

- The `KafkaConsumer` from the `kafka-python` library is initialized to consume messages from the `kafka_topic` (`sentiment_analysis`).

- The `bootstrap_servers` parameter specifies the Kafka server (`localhost:9092`), while the `group_id` parameter identifies the consumer group. Consumers in the same group share message load, improving scalability.

- The `auto_offset_reset='earliest'` ensures that the consumer fetches all messages from the beginning, and `enable_auto_commit=True` automatically commits the message offsets to Kafka.

2.) Message Listening:

- The `for message in consumer` loop listens for messages from the Kafka topic continuously. Each message contains the article's metadata (e.g., title) in JSON format.

- The `message.value` is deserialized from its raw byte format into a UTF-8 string using `v.decode('utf-8')`.

3.)Data Processing:

- The message, once deserialized, is passed to `json.loads()` to convert it from a string into a Python dictionary for further processing.

- The `Title` field from the message is extracted for sentiment analysis.

- A TF-IDF vectorizer (`tfidf_vectorizer.pkl`) is loaded via `joblib`, which transforms the article title into a numerical vector format for machine learning.

4.) Prediction:

- The transformed data is passed to a pre-trained model (`model_new.pkl`), which was likely built during the training phase (using Logistic Regression, Random Forest, etc.).

- The model predicts the sentiment (`Positive` or `Negative`) based on the input article title.

- The prediction is printed alongside the article title.

5.)Error Handling:

- If any JSON parsing errors occur (`json.JSONDecodeError`), the program logs the error and the raw message that caused it, ensuring robust error handling.

Use Case:

The consumer is designed to retrieve article data from Kafka, transform the text into a format suitable for sentiment analysis (TF-IDF vectorization), and apply a machine learning model to predict the sentiment of new data in near real-time.

This process efficiently simulates real-time data retrieval and analysis from a news feed (like Economic Times).

Step 4 – Create Labels - Snorkel Technique:

Since the dataset lacks labels, we will utilize Snorkel to develop heuristics and programmatic rules that classify headlines into two categories: positive (1) and negative (0). We will implement labeling functions that check for specific keywords in the headlines; for example, the presence of the word "promising" will result in a positive label, while "aggressive" will yield a negative one.

Additionally, we'll use TextBlob, a pretrained sentiment analysis tool, to create functions that extract polarity and subjectivity scores from each headline. By combining all labeling functions and applying them to our dataset, we can fit a LabelModel to generate the positive and negative classes. After filtering out unlabeled data points, we'll have approximately 12,300 positive labels and 6,900 negative labels, providing a solid foundation for constructing our sentiment classifier.

Step 5 – Apply Supervised Learning Approach – Logistic Regression:

To build the sentiment classifier, we begin with **Logistic Regression**, a widely used binary classifier known for estimating the probability that an instance belongs to a particular class. Before we can train the model, it’s essential to preprocess the data effectively.

3.1 Text Pre-processing

This stage is critical in Natural Language Processing (NLP) to convert raw text into clean, analyzable tokens. Key methods include:

1. Tokenizing: Splitting text into individual words or tokens. This process is fundamental for further analysis, as it breaks down the text into manageable units.

2. Lemmatizing: This step reduces words to their root forms, ensuring that different inflected forms of a word are treated as a single item. For example, "running" and "ran" would be reduced to "run."

3. Stop Word Removal: Common words (e.g., "the," "is," "and") are removed as they typically add little meaning to the analysis and can skew results.

4. Punctuation Removal: Unnecessary punctuation marks (like commas and periods) are stripped away to ensure a cleaner dataset, making it easier to analyze the text content.

After applying these techniques, the resulting data is represented as clean tokens, each in its root form and devoid of stop words and punctuation. This cleansing process enhances the model's ability to focus on the significant aspects of the text.

3.2 Text Representation

In this step, we transform the cleaned text data into vector representations that machine learning models can understand. We utilize TF-IDF (Term Frequency-Inverse Document Frequency), which evaluates the importance of a word in a document relative to a corpus. This method assigns a weight to each word based on its frequency in a document and its rarity across all documents, helping highlight terms that are significant for sentiment classification.

3.3 Model Training

With the data now in a suitable format, we proceed to train our Logistic Regression model. We first split the dataset into training and testing subsets. The model is fitted to the training data, learning to associate features with sentiment labels. Upon evaluating the model on the test set, we achieve an impressive accuracy score of 92%, indicating that the model effectively captures the sentiment expressed in the headlines.

3.4 Prediction on New Instances

After training, we can predict the sentiment of new instances. By inputting a new headline, such as one mentioning "war and sanctions," the model would classify it as negative based on the sentiment learned during training. This capability enables real-time sentiment analysis, allowing organizations to respond promptly to emerging trends and public sentiment shifts.

Overall, this systematic approach—spanning data preprocessing, representation, model training, and prediction—establishes a robust foundation for effective sentiment classification using Logistic Regression.

Topic 5 – Results:

Topic 6 – Conclusion:

In this project, we built a sentiment classifier for news headlines using Logistic Regression, a powerful yet interpretable binary classifier. We began by generating labels for the unlabelled dataset through Snorkel, using heuristic rules for positive and negative sentiment. Following text pre-processing and TF-IDF transformation, we trained the Logistic Regression model. The classifier achieved a 92% accuracy rate. Logistic Regression proves ideal for this task due to its simplicity and efficiency, especially when dealing with smaller datasets where deep learning's computational requirements would be unnecessary.

This method offers flexibility in terms of application, providing strong performance without the need for the massive datasets and computational power required by more complex deep learning models.

Topic 7 – References:

https://towardsdatascience.com/sentiment-analysis-on-news-headlines-classic-supervised-learning-vs-deep-learning-approach-831ac698e276

<u>Topic 8 – Individual Contributions of Team Members:</u>

1. Tejas- web scrapping
2. Pranav- random forests and entity resolution
3. Suchit- kafka and logistic regression
4. Arnav- sentiment analysis using textblob and ui
5. Yashyas- ui and sentiment analysis using snorkel