Deep Learning-Based Sentiment Analysis on Twitter Data

Sudipta Ghosh
Roll No.: 23MA60R03

A report submitted as part of the requirements for the course of MA67024
at the Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur, India

17th April 2025

Supervisor: prof. Adrijit Goswami

Abstract

Twitter, a leading microblogging platform, stands out as one of the most impactful social media sites, enabling users to connect, voice their thoughts, and exchange information on diverse topics and products. This activity generates a vast volume of unstructured data, drawing considerable interest from researchers eager to explore and interpret the sentiments embedded in this user-generated content. Sentiment analysis involves detecting and extracting opinions from text, with numerous lexicon-based and machine learning techniques developed over time to tackle this challenge. Recently, deep learning methods have emerged as the preferred approach due to their exceptional performance. This study outlines standard preprocessing steps and various word embedding techniques used in data preparation. It then presents a detailed taxonomy summarizing deep learning-based methods for sentiment analysis. The research also gathers information on widely used benchmark datasets, evaluation metrics for assessing performance, and publicly available resources that support sentiment analysis efforts. Additionally, the survey explores practical, domain-specific applications of sentiment analysis. In closing, it addresses key research challenges and suggests potential directions for future exploration.

Contents

Chapter 1

Introduction

Sentiment analysis, a significant subfield of Natural Language Processing (NLP), focuses on identifying and interpreting human emotions expressed through text. With the rapid expansion of social media and microblogging platforms, sentiment analysis has gained increasing prominence among researchers. Users frequently share their opinions and experiences on various topics, events, and products, making these platforms rich sources of real-time, opinion-based data. Among social networks, **Twitter**—now rebranded as **X** stands out due to its massive user base and high volume of activity. As of 2025, Twitter reported over 600 million monthly active users and approximately 500 million tweets per day. This vast and continuous flow of user-generated content makes Twitter a focal point for sentiment analysis research. Companies leverage this data to gain deeper insights into customer reactions and public perception of their products and services. However, sentiment analysis in social media poses unique challenges due to the informal and unstructured nature of the content. Unlike traditional reviews, tweets are constrained by a character limit (140 characters as of the referenced time) and often contain abbreviations, slang, URLs, emoticons, user mentions, and hashtags. These characteristics make preprocessing and accurate sentiment classification more complex compared to other text domains like product or movie reviews.

Today, when someone wants to purchase a consumer product, they are no longer restricted to seeking advice from friends or family. A wealth of user reviews and discussions is readily accessible on public online platforms. Similarly, organizations may not need to rely solely on traditional methods like surveys, opinion polls, or focus groups to understand public sentiment. The internet offers a vast amount of publicly shared opinions. In recent years, the rise of social media has significantly influenced consumer behavior, business strategies, and public emotion, leaving a deep impact on both our societal and political landscapes.

In existing research, a wide range of techniques have been developed to address different aspects of sentiment analysis, utilizing supervised and unsupervised approaches. In supervised learning, initial studies employed a variety of machine learning algorithms -such as Naive Bayes, Maximum Entropy, and Support Vector Machines (SVM) - in addition to diverse feature engineering strategies. However, unsupervised approaches often leverage sentiment lexicons, syntactic structures, and grammatical patterns to extract sentiment information. Numerous survey papers and books have thoroughly documented these foundational methods and their applications.

The advent of deep learning has revolutionized sentiment analysis, providing powerful tools to address these limitations. Models such as long-short-term memory (LSTM) networks and bidirectional LSTM (BiLSTM) networks have demonstrated remarkable success in modeling sequential data, making them well-suited for processing Twitter's short, yet complex, text. By leveraging their ability to retain long-term dependencies and, in the case of BiLSTM, process text bidirectionally, these models offer improved accuracy and robustness over earlier techniques. This report explores the application of LSTM and BiLSTM models to Twitter sentiment analysis, aiming to classify sentiments expressed in tweets with high precision. The study builds on a foundation of standard preprocessing techniques, word embeddings, and deep learning architectures, while also evaluating their performance against established benchmarks.

1.1 Motivation

The driving force behind this project lies in the increasing demand to leverage social media data for practical, real-world purposes, including brand monitoring, political analysis, and public health surveillance. By focusing on Twitter sentiment analysis, this study aims to enhance the understanding of how deep learning techniques can extract valuable insights from the noisy, unstructured nature of tweet data. The report is organized as follows: it starts with a comprehensive review of prior literature, proceeds with an in-depth methodology detailing data collection, preprocessing, and model development, and continues with sections on results, their implications, and a conclusion addressing challenges and future research opportunities. Through this investigation, the project seeks to highlight the power of deep learning in deciphering Twitter users' sentiments, while also identifying areas ripe for further improvement.

1.2 Problem Statement

Twitter is a widely used social media platform where users share short messages called "tweets" to express their opinions and emotions on various topics. These tweets have become valuable for analyzing public sentiment, and many stakeholders—including businesses and marketers—leverage sentiment analysis to gain insights into consumer opinions and perform market research. With recent progress in machine learning, the accuracy of sentiment prediction models has significantly improved.

In this project, I focus on performing sentiment analysis on tweets using a range of deep learning techniques. The goal is to determine the overall sentiment of each tweet—either positive or negative. In cases where a tweet contains both sentiments, the dominant one is assigned as the final label. The dataset used for this analysis is sourced from Kaggle, which includes pre-labeled tweets indicating sentiment polarity. Since the raw data contains elements like emoticons, usernames, and hashtags, a preprocessing step is essential to standardize the text.

Feature extraction techniques such as unigrams and bigrams are employed to effectively represent textual content. Using these features, various deep learning models are trained and fine-tuned through hyperparameter optimization to achieve improved prediction performance. The outcomes of these experiments and the insights gained are presented in the final section of this report.

1.3 Literature Servey

Aditya Timmaraju, Vikesh Khanna. Sentiment Analysis on Movie Reviews Using Recursive and Recurrent Neural Network Architectures. Department of Electrical Engineering, Stanford University, Stanford, CA - 94305, adityast@stanford.edu.

Aditya Timmaraju and Vikesh Khanna proposed an innovative model for classifying movie reviews through the use of Recursive RNN. This model addresses sentiment classification in natural language text, extending sentence-level sentiment classification tasks.

Jan Deriu, Mark Cieliebak. Sentiment Analysis Using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets. Zurich University of Applied Sciences, Switzerland. deri@zhaw.ch, ciel@zhaw.ch.

Jan Deriu and Mark Cieliebak developed a system for classifying the sentiment of Twitter messages. Their approach builds on deep learning, particularly employing a 2-layer convolutional neural network. They divided the sentiment analysis task into three distinct sub-tasks to improve efficiency.

M. Anand Kumar, S. Sachin Kumar. Sentiment Analysis of Tweets in Malayalam Using Long Short-Term Memory Units and Convolutional Neural Networks. Mining Intelligence and Knowledge Exploration: 5th International Conference, MIKE 2017, Hyderabad, India, December 13–15, 2017.

Anand Kumar and Sachin Kumar introduced a novel system that utilizes CNN for sentiment analysis in Malayalam tweets. This paper discusses how most texts contain emotional content and opinions, which are categorized into negative, positive, or neutral sentiments. Their model provides an in-depth sentiment analysis using deep learning approaches.

Fenna Miedema, Prof. Dr. Sandjai Bhulai. Sentiment Analysis with Long Short-Term Memory Networks, Vrije Universiteit Amsterdam, August 1, 2018.

Fenna Miedema proposed an advanced sentiment analysis system that classifies tweets in text format. The model employs Long Short-Term Memory (LSTM) networks, making it one of the most effective models currently available for sentiment classification tasks.

Tony Mullen and Nigel Collier. Sentiment Analysis Using Support Vector Machines with Diverse Information Sources, National Institute of Informatics (NII), Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo, 101-8430, Japan.

Tony Mullen and Nigel Collier developed a sentiment classification system utilizing support vector machines. Their model uses multiple information sources, with successful outcomes in many cases. However, it faces challenges in classifying longer sentences accurately. In contrast, models that incorporate LSTM networks show improved performance, particularly in handling long-text data.

Chapter 2

Sentiment Analysis on Twitter Data

Opinion Mining or **Sentiment analysis** is a widely studied problem in the field of natural language processing, aiming to determine the emotional tone or attitude conveyed in a given piece of text. This sentiment can typically be categorized as positive, negative, or neutral.

2.1 Understanding Sentiment Analysis

Sentiment analysis is a technique in natural language processing (NLP) used to determine the emotional tone or polarity of a piece of text. Among the various approaches to sentiment analysis, one of the most common is categorizing the text as positive, negative, or neutral. For instance, consider the following tweets that mention @VarizonSupport:

- "dear @varizonsupport your service is straight in dallas.. been with y'all over a decade and this is all time low for y'all. i'm talking no internet at all." $\rightarrow$ Would be tagged as "Negative".
- "@varizonsupport ive sent you a dm" $\rightarrow$ would be tagged as "Neutral".
- "thanks to michelle et al at @varizonsupport who helped push my no-show-phone problem along. order canceled successfully and ordered this for pickup today at the apple store in the mall." $\rightarrow$ would be tagged as "Positive".

Sentiment analysis makes it possible to handle large volumes of textual data efficiently and in real-time. For instance, if you need to examine thousands of tweets, customer reviews, or support queries, doing it manually would be time-consuming. Instead, sentiment analysis can automatically evaluate the tone of the content, helping you understand public opinion on a particular subject, extract valuable insights for informed decision-making, and streamline various business operations.

2.2 Importance of Sentiment Analysis

Sentiment analysis plays a crucial role in helping businesses gain insights into how customers perceive their brand. By automatically detecting emotions expressed in social media posts, customer reviews, and other forms of communication, companies can make better-informed decisions. It encompasses a variety of techniques used to analyze data reflecting customer opinions on products or services.

To effectively perform sentiment analysis, several aspects are considered:

- **Polarity:** Determines whether the expressed sentiment is positive, negative, neutral.
- **Subject:** Identifies the topic or item being discussed.
- **Opinion Holder:** Refers to the individual or entity expressing the sentiment.

Sentiment analysis involves automatically interpreting natural language input, extracting key opinions or statements, and categorizing them based on the sentiment they convey.

Some key applications in business include:

- **Enhancing Customer Experience:** By analyzing feedback, businesses can refine their products and services, address issues promptly, and differentiate themselves in the market.
- **Customer Satisfaction Monitoring:** Users often express their product experiences through natural language feedback. Analyzing this input offers valuable insights into customer satisfaction and helps identify areas for improvement.
- **Real-Time Issue Detection:** Social media enables customers to instantly share dissatisfaction with a wide audience. Sentiment analysis helps detect and respond to such issues quickly.

2.3 How Sentiment Analysis Operates

SA in Natural Language Processing (NLP) is used to detect the sentiment conveyed in a piece of text, such as a review, comment, or social media post. The objective is to determine whether the expressed sentiment is positive, negative, or neutral. In general, the process can be understood through the following key steps:

2.3.1 Data Description:

The data set used for this project is the Sentiment 140 Dataset, which contains **1.60 million tweets** collected via the Twitter API. It is commonly used for training sentiment analysis models. The data set includes the following columns:

- **target:** Indicates the sentiment polarity of the tweet — either positive or negative.

- **ids:** A unique identifier assigned to each tweet.
- **date:** The timestamp when the tweet was posted.
- **flag:** Refers to the query used to extract the tweet. If no specific query was used, this field is marked as NO QUERY.
- **user:** The username of the account that posted the tweet.
- **text:** The actual content or message of the tweet.

2.3.2 Data Cleaning:

Raw tweets collected from Twitter often result in a **noisy dataset** due to the informal and spontaneous nature of social media communication. Tweets typically contain unique elements such as **retweets**, **emoticons**, **hashtags**, **user mentions**, and other non-standard text, which need to be properly handled and extracted. To make the data suitable for machine learning models, it is essential to **normalize** the raw Twitter data. This ensures that the dataset is cleaner, more consistent, and easier for classifiers to learn from.

In our approach, we applied a comprehensive set of **preprocessing steps** aimed at standardizing the text and reducing the dataset's overall size. The initial phase involved general preprocessing techniques, outlined as follows:

- Convert the tweet to lowercase.
- Replace consecutive dots (more than one) with a single space.
- Remove leading and trailing spaces, as well as quotation marks (" and ').
- Replace multiple consecutive spaces with a single space.

Here's how we manage unique Twitter features.

Html Tags and URL:

When working with textual data, it is essential to remove HTML tags to extract clean text without any formatting. This step is crucial to ensure consistency across the data set. The regular expression used to match Html Tags is `<.*?>`. Also users frequently include **hyperlinks** to external websites in their tweets. However, these specific URLs are generally not useful for text classification, as they contribute to high sparsity in the feature space. To address this, we replace all URLs in the tweets with the placeholder word "**URL**". This helps in reducing noise while preserving the presence of a link as a meaningful token. The following regular expression is used to identify and replace URLs: `((www\.[\S]+)|(https?\://[\S]+))`.

User Mention:

Each Twitter user is identified by a unique handle. Users frequently reference other users in their tweets using the format `@handle`. To handle such mentions, we replace them with the placeholder `USER__MENTION`. The regular expression used to identify user mentions is `@[\S]+`.

Emotion:

On social media, users frequently include a variety of emoticons in their tweets to express diverse emotions. Given the continually growing number of emoticons, it is impossible to entirely catalog them all. Nevertheless, we do identify and categorize some commonly used emoticons. These identified emoticons are then substituted with either `EMO_POS` for positive emotions or `EMO_NEG` for negative ones. Figure 2.1 contains a list of all the emoticons identified by our methodology.

Emoticon(s)	Type	Regex	Replacement
`:), :  ), :-), (:, ( :, (-:, :')`	Smile	`(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))`	`EMO_POS`
`:D, :  D, :-D, xD, x-D, XD, X-D`	Laugh	`(:\s?D|:-D|x-?D|X-?D)`	`EMO_POS`
`;-), ;), ;-D, ;D, (;, (-;`	Wink	`(:\s?\(|:-\(|\)\s?:|\)-:)`	`EMO_POS`
`<3, :*`	Love	`(<3|:\*)`	`EMO_POS`
`:-(, :  (, :(, ):, )-:`	Sad	`(:\s?\(|:-\(|\)\s?:|\)-:)`	`EMO_NEG`
`:,(, :'(, :"(`	Cry	`(:,\(|:\'\(|:"\()`	`EMO_NEG`

Figure 2.1: List of emoticons matched by our method

Hashtag:

Hashtags are sequences of words or phrases preceded by a hash (#) symbol, often used by users to emphasize trending topics on Twitter. To process these hashtags, we replace them by removing the hash symbol. For example, `#hello` becomes `hello`. The regular expression used to detect hashtags is `#(\S+)`.

Retweet:

Retweets are originally posted by someone else and then shared by users. They start with `RT`. We eliminate `RT` from tweets because it is not a crucial element for text classification. The regular expression employed to identify retweets is `\brt\b`.

Following the application of tweet-level pre-processing, we processed each tweet's individual words as follows.

- Remove punctuation characters `['"?!,.():;]` from the word.
- Reduce occurrences of repeated letters to two. Some users write tweets such as *I am sooooo happpppy* by repeating letters for emphasis. This rule addresses these cases by converting them to *I am soo happy.*
- Eliminate - and ' symbols. For instance, transform words like t-shirt and their's to the more standard forms tshirt and theirs.

- Check the word's legitimacy. A word is deemed legitimate if it begins with a letter and is followed by a string of letters, digits, or the characters underscore texttt(_) and dot (.).

ChatWords:

Eliminating chat words enhances comprehension, thereby enabling the model to interpret content more accurately. This process ensures language consistency and reduces noise in the data.

Table 2.1: Common Chat Abbreviations and Their Meanings

Chat Word	**Meaning**	**Chat Word**	**Meaning**
BRB	Be right back	BTW	By the way
OMG	Oh my God/goodness	TTYL	Talk to you later
OMW	On my way	SMH/SMDH	Shaking my head/shaking my darn head
LOL	Laugh out loud	TBD	To be determined
IMHO/IMO	In my humble opinion	HMU	Hit me up
IIRC	If I remember correctly	LMK	Let me know
OG	Original gangsters (used for old friends)	FTW	For the win
NVM	Nevermind	OOTD	Outfit of the day
Ngl	Not gonna lie	Rq	real quick
Iykyk	If you know, you know	Ong	On god (I swear)
YAAAS	Yes!	Brt	Be right there
Sm	So much	Ig	I guess
Wya	Where you at	Istg	I swear to god
Hbu	How about you	Atm	At the moment
Asap	As soon as possible	Fyi	For your information

Punctuation and StopWords:

Everybody the command string typically displays the punctuation marks. For example `` !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ ``. It has to do with enhanced tokenisation. Eliminating the punctuation marks helps in properly recognising and distinguishing the words present in the Data. It minimises the Noise of the Data. The eliminating of stopwords helps to tooks only the most relevant content of the Data. The elimination of stopwords assists for the clear analysis

Emojies:

emoji.demojize(text) is a function from the **emoji** library in Python. It transforms emojis in the input text into their corresponding descriptive names in text format. For instance , becomes *grinning_face*. This approach helps in replacing emojis with their textual representations, making the data more consistent and easier to process—especially when preparing text data for analysis. This function is typically applied to each entry in the 'Text' column to ensure all emojis are standardized or removed.

Lemmatization:

Lemmatization is a more sophisticated and precise method of text normalization than stemming. Rather than just removing suffixes, lemmatization analyzes the context and part of speech to return a word to its meaningful base form, known as the lemma. For instance, it transforms "running" into "run" and "better" into "good," preserving the intended meaning. In contrast, stemming might reduce "running" to "runn," which can be less accurate.

2.3.3 Word Embeddings

Unlike images, where input vectors are directly derived from pixel values (which are inherently numeric), processing textual data for neural networks is more complex since text is composed of strings or characters. To convert this text into numerical form, word embeddings are commonly used. Word embeddings transform vocabulary terms into vector representations that can be fed into machine learning models.

A basic method is *One-hot encoding*, which assigns each word a vector of size $|V|$, where $|V|$ is the vocabulary size. In this vector, all elements are zero except for a single one at the index corresponding to the word. While simple, one-hot encoding has significant limitations: it generates sparse, high-dimensional vectors, is computationally expensive, and fails to capture any semantic similarity between words.

An alternative is *Term Frequency-Inverse Document Frequency* (TF-IDF). TF-IDF scores each word based on its importance in a document relative to a larger corpus. This method is computationally efficient and useful for highlighting relevant terms in a document, making it suitable for tasks like document ranking. However, TF-IDF ignores word order and semantic context, and may struggle with documents containing highly specialized or rare vocabulary.

To address these challenges, dense word embeddings were developed. These embeddings capture both syntactic and semantic connections between words, ensuring that words with similar meanings are located near each other in the vector space. This enhances generalization, representation, and computational efficiency for sentiment analysis tasks. Several techniques for generating word embeddings have been proposed to transform textual data into dense numerical vectors that are compatible with neural networks. Below is a summary of some of the most commonly used methods.

Word2Vec[1] is a widely used word embedding technique based on neural network architectures that aim to capture the linguistic context of words. It utilizes a shallow, two-layer

[1]Source code:https://code.google.com/archive/p/word2vec/

neural network that takes a corpus of text as input and produces dense vector representations (embeddings) for each word. Word2Vec offers two model architectures: *Skip-gram* and *Continuous Bag of Words* (CBOW).

- The Skip-gram model takes a single word as input and attempts to predict its surrounding context words. It generates a D-dimensional vector representation for each word. In this setup, the input layer has as many neurons as the vocabulary size, and the hidden layer has D neurons corresponding to the desired embedding dimension. The weights connecting these layers form a matrix $W_{H \times D}$ where H represents the size of the hidden layer. This weight matrix captures the probability distribution of word co-occurrence, which is refined by computing the error at the output layer and updating the embeddings through backpropagation.
- On the other hand, the CBOW model works in reverse: it takes the surrounding context words as input and tries to predict the target word. This approach is particularly effective when training data is limited or when faster training is desired.

Both models learn word embeddings by optimizing a loss function, enabling the resulting vectors to capture meaningful relationships between words in terms of both syntax and semantics.

GloVe (Global Vectors for Word Representation)[2] is a widely-used word embedding technique based on word co-occurrence statistics. Unlike purely predictive models, GloVe uses an unsupervised learning approach that constructs word embeddings by factorizing a matrix containing information about how frequently words appear together in a corpus. It effectively blends two major methodologies: local context window models (like Skip-gram) and matrix factorization techniques (such as Latent Semantic Analysis), capturing both local and global statistical information. GloVe outperforms traditional matrix factorization methods in capturing word analogies, which typically only uncover latent structures in vector space without utilizing context windows. At the same time, while the Skip-gram model handles word analogies well through local context, it doesn't fully exploit the global co-occurrence statistics of the corpus. GloVe bridges this gap, providing a more comprehensive and meaningful representation of word relationships.

In addition to the widely used word embedding techniques like Word2Vec, GloVe, several advanced embedding approaches have been developed. One such method is *BERT* (Bidirectional Encoder Representations from Transformers), a powerful pre-trained language model that has been fine-tuned for tasks such as sentiment analysis on Twitter data, where it has demonstrated superior performance over traditional models.

Another notable model is *ELMo* (Embeddings from Language Models), which generates contextualized word embeddings using a bi-directional LSTM architecture. ELMo captures the meaning of words based on their context within a sentence and has achieved state-of-the-art results in various NLP tasks, particularly in sentence-level sentiment classification.

[2]Source code: https://github.com/stanfordnlp/GloVe

2.3.4 LSTM Networks

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN), designed specifically to capture and learn long-term dependencies in sequential data. Originally introduced by *Hochreiter and Schmidhuber* in 1997, LSTMs have since been refined and widely adopted due to their effectiveness across a range of applications.

A major advantage of LSTMs is their ability to address the long-term dependency issue, which is a common limitation in conventional RNNs. While standard RNNs typically face difficulty in preserving information over long sequences, LSTMs are designed to maintain important information over extended time periods, enabling them to better capture long-range dependencies.

Like all RNNs, LSTMs consist of chains of repeating neural network modules. However, while a standard RNN module might have a simple structure (such as a single tanh layer), an LSTM module includes a more complex internal structure designed to regulate the flow of information, making it highly effective at sequence learning tasks. LSTMs also follow a chain-like architecture similar to traditional RNNs; however, their repeating module is significantly more complex. Instead of a single neural network layer, each LSTM unit contains four interconnected layers that interact in a carefully designed manner to control the flow of information.

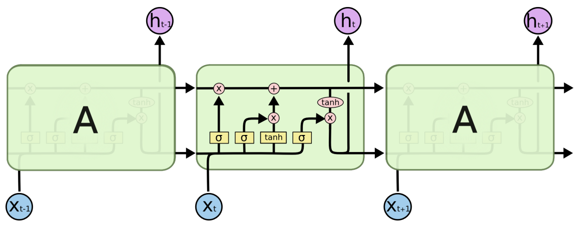


Figure 2.2: Four interacting layers make up an LSTM's repeating module.

First, we become familiar with the notation that will be used throughout.

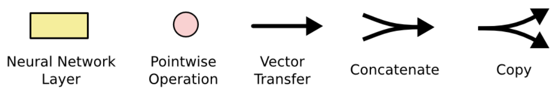


Figure 2.3

In the diagram above, each line represents a full vector that flows from the output of one node to the inputs of others. The *pink circles* represent element-wise operations, such as vector addition, while the *yellow rectangles* signify learnable layers within the neural network. When *multiple lines converge*, it indicates the concatenation of vectors. On the

[2]Image source: Adapted from *Colah's blog* (colah.github.io)

other hand, when a *line splits*, it shows that the same vector is being duplicated and sent to several destinations.

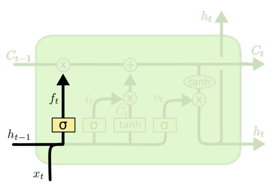
The Core Idea Behind LSTMs

At the heart of an LSTM network lies the *cell state*, which acts like a central pathway or conveyor belt, carrying information across the sequence with minimal modifications. This streamlined flow allows the network to maintain relevant information over long time steps. What makes LSTMs powerful is their ability to selectively modify the information in the cell state. This is achieved through components called *gates*. Gates regulate the information flow by using a combination of a sigmoid activation function and element-wise multiplication. The sigmoid function produces values between 0 and 1, which determine how much of the data should pass through — with 0 blocking all information and 1 allowing it all through. This mechanism ensures precise control over what is remembered, forgotten, or updated at each step in the sequence.

Three of these gates are present in an LSTM in order to maintain safety and regulate the cell state.

The initial step in an LSTM involves determining which information should be removed from the cell state. This is managed by a sigmoid activation layer known as the *forget gate*. It takes as input the previous hidden state h_{t-1} and the current input x_t, and produces an output vector with values ranging between 0 and 1. Each value corresponds to an element in the previous cell state C_{t-1}, where 1 signifies "retain completely" and 0 means "discard entirely."

Consider, for instance, a language modeling task where the goal is to predict the next word based on preceding context. In such a case, the cell state might store information like the gender of the current subject to help select appropriate pronouns. When a new subject appears, the forget gate ensures that outdated information—such as the gender of the prior subject—is removed from the cell state.



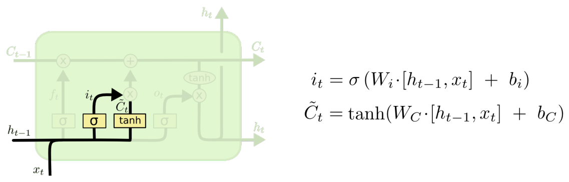
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 2.4: Showing the forget gate

The subsequent step in the LSTM involves determining which new information should be added to the cell state. This process consists of two components. First, a sigmoid activation layer known as the *input gate* identifies which values should be updated. Then, a tanh layer generates a vector of candidate values, denoted as $\tilde{C}_t$, that may be incorporated into the

[2]Image source: Adapted from *Colah's blog* (colah.github.io)

cell state. These two outputs are then combined to form the update to the cell state. For example, in a language modeling task, this stage might involve encoding the gender of a new subject into the cell state—effectively replacing outdated information being discarded in the previous step.

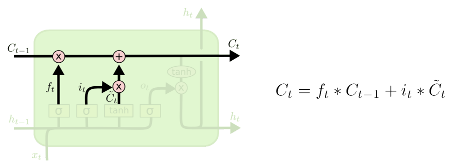


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 2.5

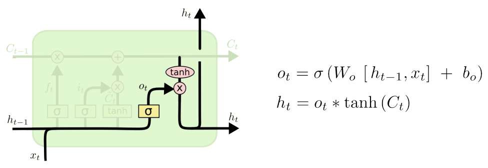
At this stage, we update the previous cell state C_{t-1} to form the new cell state C_t(Figure 2.6). The decisions made in the earlier steps now come into effect.

We begin by multiplying the old cell state C_{t-1} by the forget gate output f_t, thereby removing information that is no longer relevant. Next, we add the element-wise product of the input gate output i_t and the candidate values $\tilde{C}_t$. This operation introduces new, relevant information into the cell state. In the context of a language model, this step would correspond to discarding the gender information of the previous subject and incorporating the gender of the new subject, as determined in the prior processing steps.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.6: Updation of cell state



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.7: Output of current cell state

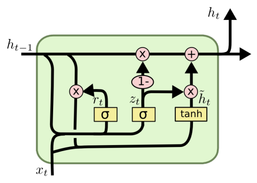
In the final step, we determine what the LSTM should output(Figure 2.7). This output is a filtered representation derived from the current cell state. First, a sigmoid layer o_t is applied to decide which parts of the cell state should influence the output. Then, the cell state C_t is passed through a tanh activation function to scale its values between -1 and 1. The result is then multiplied element-wise by o_t, effectively selecting the relevant components to form the final output h_t. For instance, in a language modeling task, after identifying a new subject, the model may need to output information relevant to an upcoming verb. This could include whether the subject is singular or plural, to ensure proper verb conjugation.

2.3.5 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is an advanced variant of the Long Short-Term Memory (LSTM) network, introduced by Cho et al. in 2014. GRUs aim to provide a simpler and more efficient alternative to LSTMs by reducing architectural complexity while maintaining performance on sequential tasks.

[2]Image source: Adapted from *Colah's blog* (colah.github.io)

Unlike LSTMs, which have separate forget and input gates, GRUs combine them into a single update gate, effectively reducing the number of parameters. Moreover, GRUs eliminate the need for a separate cell state by merging it with the hidden state, streamlining the memory mechanism.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.8: GRU Unit

This simplified structure makes GRUs faster to train and often more effective on smaller datasets or in situations where computational efficiency is a concern. Despite their simplicity, GRUs have demonstrated comparable—and sometimes superior—performance to LSTMs across a variety of natural language processing and time-series prediction tasks, contributing to their rising popularity in recent years.

2.3.6 BiLSTM

A Bidirectional LSTM (BiLSTM) is an enhancement of the standard LSTM that enhances contextual understanding by processing input data in both forward and backward directions. It consists of two LSTM layers:

- One layer processes the sequence from left to right (past to future),
- The other processes it from right to left (future to past).

By combining the outputs from both directions, BiLSTMs capture context from both previous and subsequent tokens, making them highly effective for tasks such as sentiment analysis, named entity recognition, and machine translation.

To illustrate how a BiLSTM works, let's consider the sentence "He will present tommorow." In the forward LSTM, the word "He" will be passed to the network at time step $t = 0$, "will" at $t = 1$, "present" at $t = 2$, and "tommorow" at $t = 3$. In the backward LSTM, the word "tommorow" will be passed at $t = 0$, "present" at $t = 1$, "will" at $t = 2$, and "He" at $t = 3$. The outputs from both forward and backward LSTMs are then combined at each time step.

2.3.7 Model Building

LSTM model architecture:

- *Embedding Layer:* This layer is responsible for converting the input words, represented as integer indices, into dense vectors of size 128. It acts as a word lookup table where each word in the vocabulary is mapped to a learnable embedding vector. The vocabulary size is set to 20,000, and the maximum sequence length is 150.

- *LSTM Layer:* This is the main recurrent layer of the model. It uses 128 LSTM units that are capable of learning long-term dependencies in sequential data. The return_sequences=False argument ensures that only the final hidden state is passed to the next layer, making it suitable for classification tasks. L2 regularization is applied to reduce overfitting.
- *Dropout Layer:* This layer randomly deactivates 50% of the neurons during training. It helps prevent the model from overfitting by adding noise to the network and forcing it to generalize better.
- *Dense Layer:* This fully connected layer contains 64 units and uses the ReLU activation function. It learns abstract features from the output of the LSTM layer and applies L2 regularization to avoid overfitting.
- *Output Layer:* The final layer consists of a single neuron with a sigmoid activation function. It outputs a probability value between 0 and 1, representing the likelihood that the input text expresses a positive sentiment. This makes it ideal for binary classification tasks.

BiLSTM:

- *Sequential*: Initializes a sequential model, where layers are stacked in a linear order.
- *Embedding Layer:*: This layer is the first in the model and is responsible for converting each word in the input sequence into a dense vector of fixed size. It takes in a vocabulary of 20,000 words plus one for padding and maps each word to a 128-dimensional space. By learning these embeddings during training, the model can capture semantic relationships between words. The input sequences are padded or truncated to a maximum length of 150 words to maintain consistent input dimensions.
- *Bidirectional LSTM Layer:*: Following the embedding, this layer processes the sequence using a bidirectional LSTM with 128 units. By moving both forward and backward over the input, the layer captures context from both the past and future, which is especially useful in sentiment analysis where understanding surrounding words can change the meaning of a sentence. Additionally, L2 regularization is applied to the LSTM weights to prevent overfitting, and since return_sequences is set to False, only the final output is passed to the next layer.
- *Dropout Layer:*: To help reduce overfitting and improve generalization, a dropout layer is applied after the LSTM. This layer randomly sets 50% of its inputs to zero during training, which forces the model to learn more robust and distributed representations by not relying too heavily on any single feature.
- *Dense Layer:*: After dropout, a dense (fully connected) layer with 64 units is used.

It applies a ReLU activation function, introducing non-linearity into the model and helping it learn complex relationships between features. Like the LSTM, this layer also includes L2 regularization to further discourage overfitting.

- *Output Layer:*: The final layer is a dense layer with a single neuron and a sigmoid activation function. This setup is ideal for binary classification problems, where the model must decide between two classes — in this case, positive or negative sentiment. The sigmoid function outputs a probability value between 0 and 1.

GRU Model Architecture:

- *Embedding Layer:* This layer transforms input words into 128-dimensional dense vectors, allowing the model to learn word relationships. It handles sequences of up to 150 tokens using a vocabulary size of 20,000 words plus padding.
- *GRU Layer:* The embedded input is passed through a bidirectional GRU with 128 units. It processes the sequence in both directions to capture context from past and future tokens. L2 regularization is applied to reduce overfitting.
- *Dropout Layer:* A dropout rate of 0.5 is used to randomly deactivate half the neurons during training. This helps prevent overfitting and improves generalization.
- *Dense Layer:* A fully connected layer with 64 units and ReLU activation extracts higher-level features. L2 regularization is applied here as well.
- *Output Layer:* The final layer has one neuron with a sigmoid activation function to output a probability for binary sentiment classification.

Compile The Model:

- *optimizer='adam'*: Sets the optimization algorithm to Adam, a widely used and efficient method for training deep learning models due to its adaptive learning rate capabilities.
- *loss='binary_ crossentropy'*: Defines the loss function as binary cross-entropy, which is appropriate for binary classification tasks by measuring the difference between predicted probabilities and actual class labels.
- *metrics=['accuracy']*: Specifies accuracy as the performance metric, which reflects the proportion of correctly classified instances during training and evaluation.

2.3.8 Plotting The Loss And Accuracy

Plotting the training and validation loss and accuracy helps in visualizing the learning progress of the model over epochs. It allows us to detect issues like overfitting or underfitting and assess the model's generalization capability.

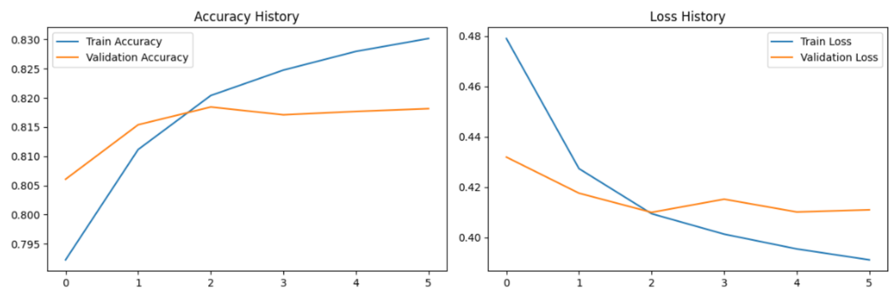


Figure 2.9: BiLSTM Model Accuracy and Model Loss

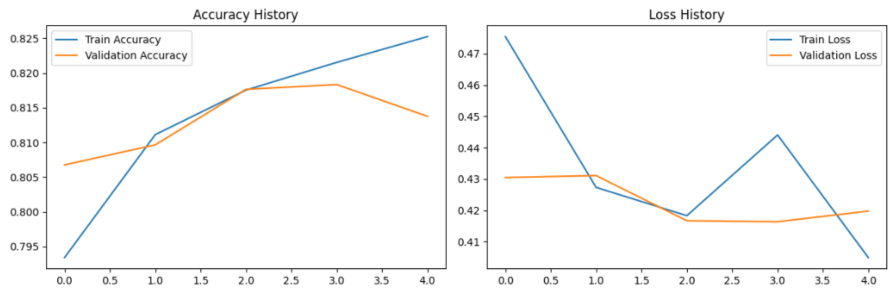


Figure 2.10: GRU Model Accuracy and Model Loss

BERT:

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model created by Google that has made significant strides in the field of Natural Language Processing (NLP). Unlike conventional models that process text in a single direction, BERT utilizes a bidirectional transformer architecture to understand the meaning of a word by considering both its left and right context. This ability to capture deep contextual relationships enables BERT to excel in various NLP applications, such as sentiment analysis, question answering, and named entity recognition. Pre-trained on an extensive text corpus, BERT can be fine-tuned with relatively small datasets to achieve cutting-edge performance on specific tasks.

2.3.9 Comparison of different Models

This section provides a performance evaluation of several deep learning architectures—GRU, BiLSTM, LSTM, and Simple RNN—applied to the Twitter Sentiment140 dataset. The evaluation uses metrics such as accuracy, precision, recall, and F1 score to measure and compare the performance of each model. Through this comparative analysis, the strengths and weaknesses of each approach in sentiment classification tasks are identified. Additionally, we examine how the models perform under varying levels of data complexity and input size. The results of this study offer insights into which architectures are most suited for real-world sentiment analysis applications. Finally, the impact of hyperparameter tuning and model optimization on performance is also explored.

Model	Accuracy	Precision	Recall	F1 Score
GRU	0.8183	0.8305	0.8000	0.8149
BiLSTM	0.8184	0.8211	0.8142	0.8177
LSTM	0.8158	0.8153	0.8181	0.8167
Simple RNN	0.5000	0.5000	1.0000	0.6667
BERT	0.8737	0.8627	0.8849	0.8795

Table 2.2: Evaluation of Various RNN Models' Performance on Twitter SA

2.3.10 Implementation

I have deployed the Sentiment Analysis model using Flask. First, I trained the model and saved it as a pickle file. For the user interface, I designed an index.html file using HTML and CSS. I also created a Python script named app.py to handle interactions between the model and the user interface. All these files are organized within a single working directory. Inside this directory, I created a virtual environment and installed all the necessary dependencies. Finally, I ran the app.py file from the terminal, and the application successfully launched on my local machine. A screenshot of the web application is attached below.

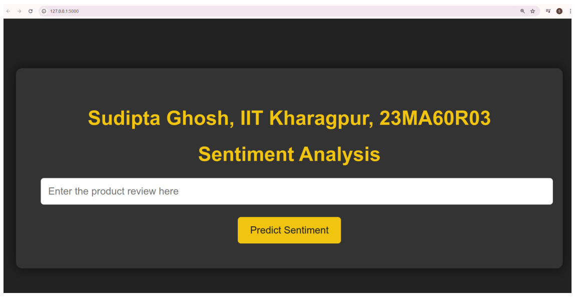

Sudipta Ghosh, IIT Kharagpur, 23MA60R03
Sentiment Analysis
Enter the product review here
Predict Sentiment

Figure 2.11: Web Interface

2.4 Practical Applications and Related Case Studies

1.Improving customer support: Support teams use sentiment analysis to tailor responses that match the customer's mood. AI-powered chatbots can quickly detect issues needing escalation and prioritize urgent cases. Machine learning models also help rank support topics by urgency and identify negative feedback about products or features. This leads to faster responses and a better overall customer experience.

2.Customer Feedback Analysis: In the fields of e-commerce and business intelligence, sentiment analysis helps organizations assess customer reviews and feedback to identify business strengths and areas for improvement. It enables companies to extract user opinions on products or events and gain valuable insights into customer preferences and market trends. For instance, Jain and Dandannavar developed a fast, scalable sentiment analysis model using Apache Spark and machine learning techniques on a Twitter dataset. Similarly, Yasir et al. applied a deep neural network (DNN) model to predict interest rates across five countries.

3.Recommendation Systems: Sentiment analysis also plays a valuable role in enhancing recommendation systems by enabling personalized user suggestions. Jain et al. proposed a hybrid CNN-LSTM model to classify customer review sentiments for recommending tailored products. Preethi et al. designed an RNN-based model to analyze review sentiments and enhance recommendations in the domains of movies and restaurants. Furthermore, sentiment analysis can support behavioral analysis in commodity markets, offering deeper insights into market trends and consumer behavior.

4.Public Healthcare: Sentiment analysis can also be applied in monitoring public health. For example, Reshi et al. analyzed tweets related to COVID-19 to identify regions most affected by the outbreak. Their findings helped inform and improve public health policies.

5.Finance Management: Sentiment analysis can assist investors in making more informed decisions by revealing market sentiment trends. By analyzing news articles and social media content, investors can better understand the emotions driving market fluctuations. For example, Swathi et al. conducted a study that analyzed tweets related to the stock market to predict changes in stock prices.

2.5 Challenges of sentiment analysis

Sentiment analysis is evolving rapidly, yet it remains a relatively young field. As highlighted by Liu Bing in Sentiment Analysis (2020), the term only became widely used around 2003[3]. Despite progress, several key challenges and limitations still need to be addressed.

1. **Lack of context:** Understanding the context behind a piece of text is essential for accurately interpreting the emotion it conveys—yet this is where sentiment analysis tools often stumble. Consider a customer survey with the question, "What did you like about our app?" A user might respond with "functionality" or "UX." These words seem positive. However, if the actual question were "What didn't you like about our app?" the same answers would reflect criticism instead. Without awareness of the original question, the sentiment analysis tool could misinterpret the response. Addressing this issue typically requires including the surrounding context through techniques like pre- or post-processing, which can be complex and resource-intensive.

2. **Use of irony and sarcasm:** No matter how advanced a sentiment analysis system is, it often struggles to detect irony or sarcasm in written text. This difficulty arises because sarcasm is typically communicated through tone or facial expressions—cues that are absent in text. For instance, consider the phrase, "Awesome, another thousand-dollar parking ticket—just what I need." A sentiment analysis tool might misclassify this statement as positive due to the word "awesome," failing to recognize the sarcastic intent behind it.

3. **Negation:** Negation occurs when a negative word alters the meaning of a sentence.

[3]"Sentiment Analysis (Second edition),", Liu, Bing, Cambridge University Press, September 23, 2020

Take the example: “I wouldn’t say the shoes were cheap.” While the intended message is that the shoes were likely expensive or not very affordable, a sentiment analysis tool might overlook this nuance and misinterpret the sentiment.

2.6 Future Outlook and Existing Research Gaps

Although deep learning models have demonstrated remarkable progress and strong performance in sentiment analysis, several research gaps and unresolved challenges remain. These areas warrant further investigation. This section highlights the existing limitations and outlines promising directions for future research in sentiment analysis.

- **Decision-making Tool:** Deep learning models are widely applied across industries such as marketing, customer service, government, and academia to analyze sentiment and support decision-making. These models can be adapted and fine-tuned to handle the complexities of text data, aiming for high accuracy in real-world applications. Research has shown that noisy features can reduce classification performance, suggesting the need for DL techniques that iteratively refine and optimize feature selection. Furthermore, these models can be enhanced to perform sentiment analysis, opinion mining, and topic detection concurrently for more comprehensive insights.
- **Limited Attention to Domain-Specific SA:** Sentiment analysis research has largely centered on developing general-purpose models, with comparatively little focus on domain-specific applications. However, there is a growing need for models tailored to specialized fields such as medical, financial, or legal domains, where understanding subtle sentiment cues is crucial. Future studies should prioritize the creation of domain-specific sentiment analysis models that can effectively interpret the unique language and context inherent in these areas.
- **Performance Measures:** Current evaluation metrics in sentiment analysis research primarily emphasize accuracy. However, this singular focus may overlook the complexities involved in interpreting sentiment. There is a growing need for more robust and comprehensive evaluation frameworks that go beyond traditional metrics like precision, recall, and F1-score, and better reflect the nuanced nature of sentiment analysis tasks.
- **Incorporating User Feedback:** Enhancing sentiment analysis models to learn from user feedback and recognize user behavior patterns can significantly improve their accuracy. Incorporating adaptive mechanisms that respond to evolving user preferences makes the models more dynamic and relevant. Furthermore, exploring interactive sentiment analysis tools that allow real-time user input can enable continuous learning, allowing the models to adjust promptly to shifting sentiments and individual preferences.

Chapter 3

Conclusions and References

In conclusion, the use of deep learning for sentiment analysis on Twitter has emerged as a significant and promising area of research, driven by the vast and diverse nature of user-generated content on the platform. This study offers an in-depth overview of recent advancements in deep learning techniques tailored for sentiment analysis on Twitter. It highlights essential preprocessing steps and the application of word embeddings, providing a clear understanding of how data is prepared for analysis.

The work introduces a simplified taxonomy that classifies existing literature into two main categories: conventional methods (lexicon-based and machine learning) and deep learning approaches, each discussed with their advantages and limitations. In addition, the study explores practical applications of sentiment analysis and identifies key research gaps, especially those relevant to domain-specific challenges.

This review also covers various evaluation metrics used in prior research to assess model performance, offering a well-rounded perspective on how sentiment analysis models are tested and compared. Deep learning techniques, in particular, have demonstrated strong potential due to their ability to capture complex linguistic patterns and manage the inherent noise and brevity of Twitter data. Methods like fine-tuning and transfer learning have shown effectiveness in customizing pre-trained models for Twitter-specific content.

Despite these advancements, several challenges persist. Twitter data is often noisy and informal, featuring short texts, typos, abbreviations, slang, and emojis—factors that complicate sentiment detection. Additionally, class imbalance remains a concern, as datasets often contain more neutral or negative sentiments than positive ones. Integrating sentiment analysis with other NLP tasks, such as named entity recognition and text summarization, could further enhance analytical insights and model performance.

Overall, this survey affirms that while deep learning methods have made substantial progress in sentiment analysis on Twitter, ongoing research and innovation are crucial to overcoming existing limitations and fully unlocking their potential.

References

[1] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011, June). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 142-150). Association for Computational Linguistics

[2] ApoorvAgarwal, BoyiXie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis on twitter data. Department of Computer Science, Columbia University, New York, NY 10027 USA. apoorv@cs, xie@cs, iv2121@, rambow@ccls, becky@cs.columbia.edu

[3] PreranaSinghal, Pushpak Bhattacharya. Sentiment Analysis and Deep Learning: A survey. Dept. of Computer Science and Engineering , Indian Institute of Technology, Powai Mumbai, India.

[4]AdityaTimmaraju, VikeshKhanna.Sentiment Analysis on Movie Reviews using Recursive and Recurrent Neural Network Architectures. Department of Electrical Engineering, Stanford University, Stanford, CA -94305,adityast@stanford.edu.

[5] Jan Deriu, Mark Cieliebak. Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets. Zurich University of Applied Sciences, Switzerland. deri@zhaw.ch, ciel@zhaw.ch.

[6] M. Anand Kumar, S. Sachinkumar. Sentiment Analysis of Tweets in Malayalam Using Long Short-Term Memory Units and Convolutional Neural Nets. Mining Intelligence and Knowledge Exploration: 5th International Conference, MIKE 2017, Hyderabad, India, December 13–15, 2017.

[7] FennaMiedema, Prof. dr. SandjaiBhulai. Sentiment Analysis with Long Short-Term Memory networks, VrijeUniversiteitAmsterdam,August 1, 2018.

[8] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources,National Institute of Informatics (NII),Hitotsubashi 2-1-2, Chiyoda-ku,Tokyo,101-8430,Japan.

[9] N. Girdhar and K. K. Bharadwaj, "Social status computation for nodes of overlapping communities in directed signed social networks," *Integrated Intelligent Computing, Communication and Security*, pp. 49–57, 2019.

[10] T. B. Mirani and S. Sasi, "Sentiment analysis of isis related tweets using absolute location," in 2016 *international conference on computational science and computational intelligence (CSCI).* IEEE, 2016, pp. 1140–1145.