22072045 AlishBan (1)

Islington College,Nepal

Document Details

Submission ID	68 Pages
trn:oid:::3618:78796989	10,162 Words
Submission Date	57,581 Characters
Jan 10, 2025, 1:24 AM GMT+5:45	
Download Date	
Jan 10, 2025, 1:27 AM GMT+5:45	
File Name	
22072045 AlishBan (1)	
File Size	
68.3 KB	

18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

180 Not Cited or Quoted 17%
Matches with neither in-text citation nor quotation marks

7 Missing Quotations 1%
Matches that are still very similar to source material

3 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
7%	Publications
16%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **180** Not Cited or Quoted 17%
  Matches with neither in-text citation nor quotation marks
- **7** Missing Quotations 1%
  Matches that are still very similar to source material
- **3** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
7%	Publications
16%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Liverpool John Moores University on 2023-12-18	1%
2	Publication	T. Mariprasath, Kumar Reddy Cheepati, Marco Rivera. "Practical Guide to Machin...	1%
3	Submitted works	University of Sydney on 2024-07-17	1%
4	Internet	fastercapital.com	0%
5	Submitted works	Liverpool John Moores University on 2024-03-16	0%
6	Internet	impa.usc.edu	0%
7	Submitted works	Technological Institute of the Philippines on 2023-04-13	0%
8	Internet	machinelearningmodels.org	0%
9	Submitted works	City University on 2023-12-06	0%
10	Internet	www.mdpi.com	0%

11	Submitted works	
University of Sunderland on 2024-08-12		0%
12	Submitted works	
CSU Northridge on 2022-11-11		0%
13	Publication	
Vivek S. Sharma, Shubham Mahajan, Anand Nayyar, Amit Kant Pandit. "Deep Lear...		0%
14	Internet	
deepai.org		0%
15	Internet	
www.coursehero.com		0%
16	Internet	
www.semanticscholar.org		0%
17	Submitted works	
University of Bristol on 2023-03-10		0%
18	Submitted works	
University of Ulster on 2024-05-03		0%
19	Publication	
R. Anandan, M. Senthil Kumar, C. L. Biji, Vicente García-Díaz, Souvik Pal. "Next-Ge...		0%
20	Internet	
peerj.com		0%
21	Submitted works	
Taylor’s Education Group on 2024-03-22		0%
22	Submitted works	
University of Bedfordshire on 2024-07-12		0%
23	Submitted works	
University of Hertfordshire on 2024-12-02		0%
24	Submitted works	
University of Hong Kong on 2018-05-04		0%

25	Internet	
link.springer.com		0%
26	Submitted works	
Kingston University on 2023-01-10		0%
27	Submitted works	
The University of the West of Scotland on 2023-04-21		0%
28	Submitted works	
King's College on 2024-04-11		0%
29	Submitted works	
University of Liverpool on 2024-04-16		0%
30	Submitted works	
Universidad Carlos III de Madrid on 2018-06-30		0%
31	Internet	
sbce.ac.in		0%
32	Internet	
www.ijraset.com		0%
33	Submitted works	
The Robert Gordon University on 2023-04-25		0%
34	Submitted works	
UT, Dallas on 2024-05-07		0%
35	Submitted works	
University of Canberra on 2024-01-12		0%
36	Submitted works	
University of Kent at Canterbury on 2014-09-15		0%
37	Internet	
en.wikipedia.org		0%
38	Publication	
Dinesh Goyal, Bhanu Pratap, Sandeep Gupta, Saurabh Raj, Rekha Rani Agrawal, I...		0%

39	Submitted works	
	The British School of Beijing on 2023-12-06	0%
40	Submitted works	
	University of Bristol on 2024-09-05	0%
41	Submitted works	
	University of Sydney on 2022-05-07	0%
42	Submitted works	
	Anna University on 2025-01-06	0%
43	Submitted works	
	University of Hull on 2022-12-20	0%
44	Internet	
	aiforsocialgood.ca	0%
45	Submitted works	
	Institute of Art Design and Technology on 2022-05-08	0%
46	Submitted works	
	RMIT University on 2024-10-19	0%
47	Internet	
	res.mdpi.com	0%
48	Submitted works	
	Liverpool John Moores University on 2021-10-10	0%
49	Submitted works	
	Universiti Teknologi MARA on 2021-07-31	0%
50	Submitted works	
	University of Westminster on 2024-09-03	0%
51	Internet	
	rgu-repository.worktribe.com	0%
52	Publication	
	Babeș-Bolyai University	0%

53	Submitted works	
National College of Ireland on 2024-12-12		0%
54	Submitted works	
University of Birmingham on 2015-04-27		0%
55	Submitted works	
University of Northumbria at Newcastle on 2024-05-23		0%
56	Internet	
bookpedia.co		0%
57	Internet	
publications.waset.org		0%
58	Submitted works	
Higher Education Commission Pakistan on 2024-10-27		0%
59	Submitted works	
Imperial College of Science, Technology and Medicine on 2021-09-22		0%
60	Submitted works	
Liverpool John Moores University on 2023-02-18		0%
61	Internet	
magnascientiapub.com		0%
62	Internet	
tuns.ca		0%
63	Internet	
www.duo.uio.no		0%
64	Internet	
www.irjmets.com		0%
65	Publication	
Hickman, Riley J.. "Automating the Scientific Method: Toward Accelerated Materi...		0%
66	Submitted works	
Liverpool John Moores University on 2024-03-18		0%

67 Publication

Pawar, Suraj. "Physics-Guided Machine Learning for Turbulence Closure and Red... 0%

68 Submitted works

Queensland University of Technology on 2022-04-03 0%

69 Submitted works

University of Birmingham on 2023-06-23 0%

70 Submitted works

University of Lancaster on 2021-11-22 0%

71 Publication

Wan, Zhengchao. "Distances within and between Metric Spaces: Metric Geometry... 0%

72 Internet

dl.icdst.org 0%

73 Internet

edepot.wur.nl 0%

74 Submitted works

City University on 2019-12-19 0%

75 Publication

Hasmik Osipyan, Bosede Iyiade Edwards, Adrian David Cheok. "Deep Neural Net... 0%

76 Submitted works

Hong Kong University of Science and Technology on 2024-11-13 0%

77 Submitted works

La Trobe University on 2024-05-05 0%

78 Submitted works

Liverpool John Moores University on 2023-05-11 0%

79 Publication

Satya Prakash Yadav, Dharmendra Prasad Mahato, Nguyen Thi Dieu Linh. "Distri... 0%

80 Submitted works

Sunway Education Group on 2023-08-04 0%

81	Submitted works	
	The African Institute for Mathematical Sciences on 2024-06-07	0%
82	Submitted works	
	The Robert Gordon University on 2024-04-06	0%
83	Submitted works	
	University College London on 2017-09-10	0%
84	Submitted works	
	University College London on 2023-03-26	0%
85	Submitted works	
	University of Bucharest on 2024-02-11	0%
86	Submitted works	
	University of Central Florida on 2024-12-04	0%
87	Submitted works	
	University of Essex on 2024-08-27	0%
88	Submitted works	
	University of Westminster on 2023-09-07	0%
89	Submitted works	
	University of Wollongong on 2024-04-16	0%
90	Internet	
	www.theknowledgeacademy.com	0%
91	Submitted works	
	Addis Ababa University on 2023-07-12	0%
92	Publication	
	Alves, João Miguel Lima. "Modelo de Aprendizagem Automática para Prever Jogos...	0%
93	Submitted works	
	B.S. Abdur Rahman University on 2017-06-29	0%
94	Submitted works	
	Coventry University on 2023-12-01	0%

95	Publication	
Durgesh Kumar Mishra, Nilanjan Dey, Bharat Singh Deora, Amit Joshi. "ICT for Co...		0%
96	Submitted works	
Imperial College of Science, Technology and Medicine on 2020-02-03		0%
97	Submitted works	
National College of Ireland on 2023-01-07		0%
98	Publication	
Pethuru Raj, P. Beaulah Soundarabai, D. Peter Augustine. "Machine Intelligence - ...		0%
99	Submitted works	
University of Al-Qadisiyah on 2023-08-23		0%
100	Submitted works	
University of Central Lancashire on 2023-08-04		0%
101	Submitted works	
University of Essex on 2021-04-02		0%
102	Submitted works	
University of Northumbria at Newcastle on 2024-01-16		0%
103	Submitted works	
University of Southampton on 2023-09-14		0%
104	Submitted works	
University of Ulster on 2024-05-11		0%
105	Internet	
assets.researchsquare.com		0%
106	Internet	
bura.brunel.ac.uk		0%
107	Internet	
vdoc.pub		0%
108	Publication	
Διακουμάκου, Σταυρούλα. "Εξόρυξη Δεδομένων στα Μέσα Κοινωνικής Δικτύωση...		0%

109 Submitted works

Eastern Gateway Community College on 2023-09-24 0%

110 Submitted works

Middlesex University on 2019-04-19 0%

111 Submitted works

National College of Ireland on 2022-04-10 0%

112 Submitted works

Otto-von-Guericke-Universität Magdeburg on 2019-04-17 0%

113 Submitted works

University of Birmingham on 2023-09-05 0%

114 Submitted works

University of Nottingham on 2024-09-02 0%

115 Submitted works

University of Wollongong on 2023-09-01 0%

116 Publication

V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challeng... 0%

117 Submitted works

Jawaharlal Nehru Technological University on 2024-10-29 0%

118 Submitted works

Leeds Beckett University on 2024-05-08 0%

119 Submitted works

Liverpool John Moores University on 2023-11-24 0%

120 Submitted works

The Robert Gordon University on 2023-12-21 0%

Introduction

Explanation of the AI topic/concepts used.

Homo sapiens, intelligent organisms, possess the ability to decide rationally. For long, attempts have been tried to realize how a human being thinks; for instance, how a handful of matter can predict, perceive, understand, and manipulate a universe far deeper and more intricate than itself. Artificial Intelligence not only tries to acknowledge it but has walked miles for developing intelligent objects.

75 The concept of AI has evolved with Alan Turing, in his publication "Computing Machinery and Intelligence" in 1950, to develop a deterministic path that separates the intelligence between the machine and the human (Turing Test) (Mueller & Massaron, 79 2021). At Dartmouth College in 1956, however, it is said that the term artificial intelligence was used for the first time. Scientist Marvin Minsky also attended this conference. The researchers there believed that a bright future can be expected for AI. But in the mid-1970s, because of slow progress and a few reports, the government stopped funding research in AI. Again in the 1980s, the British government re-ignited 44 the research in AI by starting funding. Later in 1997, when IBM's Deep Blue defeated the world chess champion, a Russian grandmaster, Garry Kasparov, AI became the matter of conversation in every corner of the world and took the path of fast development (Lewis, 2014).

31 Artificial Intelligence can be determined as the cognitive ability of a machine, whose benchmark has been compared to human Intelligence regarding speech, vision, and reasoning. According to Rungta, AI is divided into three levels (Rungta, 2018):

109 - Narrow AI – wherein a machine can perform a few tasks better than a human.

109 - General AI – in which a system performs any task as well as the best human.

- Active AI – where the machine will outperform and do a better job in many occupations currently being done by humans.

ML, being a subset of AI, relies on the underlying algorithms necessary for the analysis of gigantic data sets. ML trains computers for automating its task and makes conclusions with minimum human intervention by identifying specific data patterns. ML feeds the algorithm the data for the learning mechanism to predict the output of new input once the training is complete.

57 Finally, deep learning, which is a subset of machine learning, uses a deep neural network-like structure in the human brain to build deeper layers for learning from the data. The deeper or hidden layers are stacked in the neural network architecture.

98 After getting acquainted with the fundamentals of AI, the essence of implicating it in real life can be speculated. In the current scenario, humans have embraced AI in different sectors like astronomy, healthcare, finance, social media, data security, robotics, agriculture, education, etc.

56 Basically, AI has various fields that include Machine Learning, Deep Learning, Neural Networks, Computer Vision, Robotics, Natural Language Processing, etc. Moving on to the project at this stage, which is the main theme of Natural Language Processing, the discussion from this point will be delivered extensively on the topic in question.

Natural Language Processing investigates how computers are in use in processing and making sense out of the human language to accomplish pragmatic ends. NLP integrates a single platform of cognitive science, computing science, and computational linguistics with Artificial Intelligence.

From the researchers' perspective, NLP aims at modeling the cognitive mechanisms through the production and acknowledgment of human languages. While for the engineers, NLP aims at developing systems that assist human language interaction with machines. It is specially designed for depicting natural semantics, through either discrete or symbolic systems, (Deng & Liu, 2018). Basic lexical analysis, information retrieval, knowledge graph, parsing, question answering, speech recognition, dialogue system, natural language generation, spoken language understanding, sentiment analysis, etc., are included as primary sectors in the main frame of NLP.

Explanation of Chosen Problem Domain/Topic

93 Sentiment analysis or opinion mining is a field within NLP concerned with collecting underlying subjective knowledge from the lines of text provided. Research regarding sentiment analysis has increased several folds in the last ten years, as competition intensifies among brands to understand social sentiments about business, services, and products in demand (Gupta, 2018). Furthermore, analyzing the motives of social media users regarding political issues, current affairs, elections, social issues, and

online shopping has gained immense attention through the analysis of their online conversations. Information available over the internet can either be facts or opinions.

Facts are objective, providing information about entities without reflecting any sentiment. On the other hand, opinions are subjective, describing people's emotions about events or entities (Agarwal & Mittal, 2016). The massive volume of data generated through consumer arguments, opinions, and views about brands, products, incidents, and politics significantly affects readers, politicians, and vendors of various products. By managing and analyzing unstructured data from online conversations, reviews, posts, and statuses, assumptions can predict user sentiments that profoundly influence the subject matter.

37 On January 6, 2021, mass supporters of former President Donald Trump stormed the United States Capitol in Washington, D.C. The mob demanded to overturn the election, which they believed was rigged, where President Joe Biden was declared the winner. Tragically, five people, including a police officer, lost their lives during the incident. Afterward, four additional officers succumbed to suicide, traumatized by the 38 event (Peterson, 2021).The riot was meticulously planned using social media platforms, such as Facebook, where participants from across the country organized (Timberg et al., 2021). It is now evident that the entire incident could have been prevented.

Sentiment analysis could have been utilized to curtail the spread of misinformation and halt the planning of such a dreadful event. Beyond mitigating misinformation, sentiment analysis can serve multiple purposes, such as evaluating people's emotions, opinions, and appraisals about entities and their attributes derived from various text forms. These entities may include issues, topics, services, products, organizations, or individuals (Liu, 2015).

Ultimately, assumptions from sentiment analysis can improve product or service quality, enhance customer comfort, generate insights on diverse topics, expand business opportunities, and more uplifting the welfare of organizations. The application of sentiment analysis has gained popularity in solving challenges across domains like healthcare, financial services, stock market prediction, political elections, tourism, and hospitality. The results from these applications assist startups and governments in creating proper and well-defined regulations. Although sentiment analysis has gained traction, research on managing biased sentences remains limited due to significant complexities in the field.

Background

Research Work on Chosen Topic/Problem Domain.

Extensive research has been performed to understand the implementation of sentiment analysis, its limitations, and its applications. While reviewing various resources, forums, journals, books, and online mediums, it was identified that

48 sentiment analysis can be achieved using different approaches in Machine Learning and AI. For example, methods like Support Vector Machine (SVM), Naïve Bayes Classifier, RNN (Recurrent Neural Networks), regional CNN-LSTM, GRU (Gated Recurrent Units), linear regression, Maximum Entropy Classifier, Decision Tree Classifier, and others have been utilized (Medhat et al., 2014).

39 Several types of sentiment analysis exist, such as fine-grained sentiment analysis, which categorizes opinions on a detailed level using a 5-star scale ranging from very positive to very negative. Emotion detection focuses on identifying specific emotions like anger, shock, sadness, happiness, and frustration rather than simply labeling content as positive or negative. Another type is intent-based analysis, which predicts actions or opinions derived from written text for instance, addressing customer frustration expressed in online comments. Lastly, aspect-based analysis evaluates opinions about specific components of a product rather than the overall entity (TechTarget, 2021). For this project, fine-grained sentiment analysis will be employed, categorizing opinions into three labels: good, bad, and neutral.

The workload of sentiment analysis can be addressed using two primary methods: the 80 rule-based approach and automatic sentiment analysis. In the rule-based system, the 8 text corpus undergoes processes such as stemming, tokenization, part-of-speech tagging, parsing, and lexicon analysis. Words are grouped into good and bad categories, and an applied algorithm determines their classification based on polarity (Hutto & Gilbert, 2014). While this method produces results, it often lacks precision and flexibility, making it less practical for certain applications.

116 Automatic sentiment analysis, on the other hand, uses machine learning techniques to analyze sentiments. This approach offers enhanced precision and accuracy while 64 reducing complexity. It employs supervised machine learning algorithms for classification tasks and can also utilize unsupervised learning algorithms to uncover and understand data patterns (Boiy et al., 2007). Common classification algorithms in 62 this approach include linear regression, Support Vector Machines (SVM), Recurrent 111 Neural Networks (RNN), Naïve Bayes, and LSTM models. For this project, sentiment 25 analysis development will focus on three highly effective algorithms: Support Vector Machines (SVM), Long Short Term Memory (LSTM), and the Naïve Bayes Classifier.

31 Support Vector Machine (SVM) is considered one of the most effective and powerful 26 methods for sentiment analysis, known for its ability to efficiently detect textual polarity. 70 As a supervised machine learning algorithm, it can handle both classification and regression tasks. The classification process involves identifying the hyper-plane that separates different classes plotted on an n-dimensional graph, created using Kernels (mathematical functions). Building a robust SVM model involves key steps, including collecting appropriate data for training and testing, vectorizing the data, and developing linear SVM models for both training and prediction (Reddy, 2018).

10 LSTM (Long Short-Term Memory) is an effective deep learning method for fine-grained sentiment analysis. It incorporates regional knowledge across different areas to predict sentence-level information and textual dependencies, making it highly efficient in capturing long-term patterns and relationships in textual data (Wang et al., 2016).

8 The Naïve Bayes Classifier, a popular machine learning algorithm for sentiment analysis, is based on the Naïve Bayes (NB) theorem, which uses a probabilistic

30 approach to predict outcomes. It calculates the conditional probability of an event occurring given the presence of another event as a condition. This classifier has several variations tailored to different types of data. Multinomial Naïve Bayes is suited for classifying problems based on categories, while Bernoulli Naïve Bayes handles Boolean variables with binary outcomes, such as 'yes' or 'no.' Additionally, Gaussian 117 Naïve Bayes is designed to classify continuous data rather than discrete values, making it versatile for various use cases (Gandhi, 2018).

4 The Naïve Bayes classifier, SVM and Long Short-Term Memory (LSTM) networks are chosen for sentiment analysis due to their complementary strengths. Naïve Bayes is 3 efficient and commonly used for tasks such as spam filtering and sentiment classification, although its assumption of feature independence simplifies computation while potentially limiting performance in complex scenarios. Support Vector Machines 58 (SVM) excel in determining textual polarity by identifying the optimal hyperplane for class separation in high-dimensional spaces, ensuring high precision. Meanwhile, CNN-LSTM models are particularly adept at extracting local features and capturing sequential dependencies in text, allowing them to manage complex patterns and contextual nuances effectively. Together, these three algorithms create a robust framework for accurate and comprehensive sentiment analysis.

26 In summary, sentiment analysis is a complex task requiring more than routine extraction processes. It involves carefully defining context and polarity using text vectorization to map word connections and relationships. Tools like doc2vec and word2vec can enhance the model's performance. Additionally, for determining subjectivity and tone, the characterization of the product must be considered. The

model should be trained on a diverse corpus with deep contextual understanding to effectively identify sarcasm and irony. By marking polar messages, the model can also establish a neutral tone. Ultimately, these processes contribute to developing a highly refined NLP system.

Review and Analysis of the Existing Works

Sentiment analysis evaluates people's perspectives on various entities using metrics such as good, bad, and neutral sentiments. The significance of sentiment analysis and 65 its documentation can be traced back to the mid-twentieth century. With the advent of digitalization, traditional paper-based methods have shifted to computer software, and machine learning techniques now perform sentiment analysis. While robust systems have been developed for this purpose, ongoing research aims to enhance their efficiency. This section reviews and critically examines some related projects.

25 One notable study is the paper by Piek Vossen and Isa Maks, titled "A Lexicon Model for Deep Sentiment Analysis and Opinion Mining Applications." This work explores the complex processes involved in developing and evaluating intricate lexicon models. The model classifies words into categories such as adjectives, nouns, and verbs to support sentiment analysis tasks. It has significantly advanced research in creating resources necessary for sentiment analysis. Initially, opinions expressed in textual form are

contextualized using opinion holders and targets with the semantic FrameNet model. The subjective relationships between different actors are also taken into account. Semantic categorization is then employed to annotate words, akin to opinion mining, but tailored to the writer or speaker's perspective. Additionally, the word annotation model aids in conveying the sentiment and emotional tone of the text corpus. Lastly, the model incorporates the roles of the writer and speaker as reflected in the text, a feature crucial for achieving accurate sentiment analysis and detecting biases within the context.

Another project by Josef Steinberger addresses the challenges of obtaining lexical resources from multiple languages for sentiment analysis. This work aims to enhance multilingual sentiment analysis through a method called "triangulation." This approach not only facilitates the creation of resources in two different languages but also enables their translation into additional languages. The method seeks to minimize noise introduced by automatic translations and resolves issues related to word sense ambiguity. Furthermore, the author emphasizes the need to expand, refine, and improve word lists to develop a precise and accurate sentiment analysis system.

16 The article "On Developing Robust Models for Favorability Analysis: Model Choice, Feature Sets and Imbalanced Data" by Peter C.R. Lane highlights the challenges of extracting opinions from media content, specifically identifying documents with positive or negative sentiment. The project explores experiments conducted on real-time media data for sentiment analysis. However, the imbalance between positive and negative

document classes, coupled with fluctuating content, often leads to underperformance in machine learning models. To address these challenges, the article proposes several alternatives tailored to the dataset's characteristics and specific tasks.

Another compelling article, "The Socialist Network" by Antal van den Bosch and Matje van de Camp, discusses a sentiment analysis system designed to extract personal social networks from ancestral texts. This project makes a valuable contribution by identifying and addressing critical challenges in sentiment analysis. While the outcomes rely heavily on opinion analysis, the authors showcase the application of advanced machine learning algorithms, provide a comprehensive comparative perspective, and incorporate diverse linguistic properties and resources. The findings from this work are instrumental in driving further advancements in sentiment analysis.

16 The paper "Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models" by Wei Li et al. explores the CNN-BiLSTM and CNN-LSTM models. It introduces a sentiment padding method that standardizes data sample sizes and enhances sentiment information. The study also addresses challenges like gradient vanishing between hidden and input layers. A proposed loss function enables the two branches of the model to train at varying speeds, boosting sentiment analysis

2 performance. The exceptional performance of the CNN-LSTM model on the given dataset underscores the dependence of system efficiency on sentence structure. However, the skip connection operation has shown to reduce model performance and

training speed, suggesting the need for further investigation into the coupling factors between the two model branches.

The research detailed above tackles sentiment analysis through various sophisticated approaches. Several challenges were encountered during the integration of the proposed models with datasets, and the authors have offered solutions to address these limitations. Sentiment analysis remains a complex yet expanding field, with increasing applications across diverse domains. Extensive future research is required before relying solely on such systems. Additionally, current model studies primarily focus on detecting explicit, direct sentiments and implicit expressions conveyed through emotion-eliciting and objective arguments in sentences.

6 For this project, Naïve Bayes, Support Vector Machines (SVM), and Long Short Term Memory (LSTM) have been employed to analyse textual sentiments. The latter sections of this document delve into the benefits, limitations, challenges, and areas for improvement for each algorithm Naïve Bayes, SVM, and LSTM offering a comprehensive exploration of their potential.

Dataset

Dataset Information

The dataset you're referring to, hosted on Hugging Face, is called "GPT-Sentiment-Analysis-200K". It's designed to facilitate research and development in sentiment analysis, a field of natural language processing (NLP) focused on determining the emotional tone behind a body of text. This dataset is particularly valuable for training machine learning models to recognize and categorize sentiments expressed in tweets.

Dataset Information:

Size: The dataset contains approximately 219,000 entries.

Content: Each entry includes a tweet along with a sentiment label. The sentiments are categorized as "good", "bad", or "neutral".

Format: The dataset is structured with each tweet's text and its corresponding sentiment label, which can be used to train models to automatically predict the sentiment of a given text.

Dataset Background

The "GPT-Sentiment-Analysis-200K" dataset was developed specifically to enhance the field of sentiment analysis in machine learning, which focuses on identifying and categorizing emotions conveyed through text. This dataset, consisting of around 200,000 tweets, is designed to train machine learning models to

1 automatically detect and interpret the sentiment expressed in a piece of text, improving their ability to understand human emotions.

11 Using real-world data from tweets, this dataset is particularly useful for applications such as monitoring social media platforms, analyzing customer feedback, and assessing public sentiment trends. The varied and authentic linguistic expressions found in these tweets provide a rich resource for developing robust models capable of

1 handling complex, nuanced language, thereby increasing the accuracy and relevance of sentiment analysis technologies in various sectors.

Solution

Explanation of the Proposed Solution.

As internet usage has grown, so has the way people interact with web applications, sharing their thoughts through reviews, comments, ratings, forms, and posts. This wealth of data is now analyzed to make improvements that benefit both businesses and their clients. Sentiment analysis, which looks at people's feelings in their words,

95 can be done in various ways, including through machine learning and rule-based systems. Machine learning, in particular, is popular because it doesn't rely on fixed rules to classify information (Ravi & Ravi, 2015).

In the machine learning method, we feed an algorithm with example data, and it learns how to categorize new inputs based on that training. Among the techniques for

6 sentiment analysis, Naïve Bayes, Support Vector Machines (SVM), and Long Short Term Memory (LSTM) stand out for their simplicity and effectiveness. These methods

treat each piece of data as separate from the rest, making them straightforward to implement and understand.

We'll start by collecting tweets from an online source to use in our analysis. These tweets will be processed and analysed using three types of machine learning techniques: the Naïve Bayes Classifier, Support Vector Machines (SVM), and Long Short Term Memory (LSTM). First, we'll need to prepare the data by removing any unwanted elements like punctuation marks, tabs, and emoticons to make the text cleaner and more analysable. Once the data is prepped, we'll categorize it by assigning labels that the algorithms will use to learn and recognize patterns. After labelling, we'll extract features from the text these are the key elements the models will use to understand and classify the sentiment of each tweet. This methodical preparation will help ensure our models are well-trained and provide accurate sentiment analysis.

Sentiment Classification using Naïve Bayes Classifier.

We'll begin by creating a "bag of words" which is essentially a collection of unique dishes mentioned across all tweets. This collection helps us understand how often certain words appear, although it doesn’t account for their order in a sentence. This simplification can sometimes make it challenging to fully understand sentences with complex structures.

In this approach, we'll treat each tweet as a single data point where the text is the statement and its sentiment (like positive, negative, or neutral) is the label. We'll then convert words into counts of how frequently they appear. Using the Naïve Bayes theorem, we'll calculate the likelihood of each sentiment based on these word frequencies, ultimately predicting the sentiment of new tweets based on which outcome has the highest probability.

To refine our model, we'll also use techniques like tokenization and stemming to break
23 down large texts into basic forms, and lemmatization to reduce words to their root
103 forms. These steps help improve the model's accuracy by simplifying the text data.
Techniques such as using stop words common words that are often filtered out and transforming text into vector forms through tools like doc2vec and word2vec are common in modern natural language processing (NLP) research.

36 The underlying principle of the Naïve Bayes Classifier is based on Bayes' Theorem,
which uses probability to make predictions. Here's how it works: It considers the
30 likelihood of an event occurring in relation to the occurrence of another event. The
formula we use looks like this:

Where:

A, B = events

54 P(A) = independent probability of an event A's occurrence (prior).

P(B) = independent probability of an event B's occurrence (evidence).

3 P(A | B) = probability of A such that B is prior condition (posterior).

P(B | A) = probability of B such that A is prior condition (likelihood).

Figure 1 Naïve Bayes Classifier for Sentiment Analysis

Sentiment Classification using Support Vector Machine.

Further, a feature vector will be created that stores unique features from the entire training dataset to represent the text in a structured format. During this approach, the frequency of words and their relevance will be considered, while their exact position in a sentence may not always hold importance. For training, the dataset column will be treated as the input text (statements) and their corresponding labels (good, bad, or neutral).

27 After creating the feature vector, the data will be converted into numerical representations, such as TF-IDF scores, to prepare for classification. The Support 87 Vector Machine (SVM) algorithm will then be applied, which will identify the optimal hyperplane that separates the different sentiment classes with maximum margin. 92 Eventually, the trained model will classify the input text as bad, good, or neutral by determining which side of the hyperplane the data points fall on.

Additionally, the use of tokenization and lemmatization during pre-processing will enhance the accuracy of the model. These methods break down sentences into smaller, meaningful components and ensure consistency by reducing words to their base form. Techniques like removing stop words and using TF-IDF weighting will prioritize the most relevant features while ignoring less significant ones. Moreover, advanced embedding tools such as word2vec and doc2vec can be employed to capture semantic relationships between words and improve the feature representation, which is a standard practice in modern NLP research.

19 The Support Vector Machine (SVM) is based upon the concept of identifying an
32 optimal hyperplane that separates data points into distinct classes while maximizing the margin between the hyperplane and the nearest data points (support vectors). This margin maximization ensures better generalization of the model to unseen data.

The equational representation of the formula along with the notation is depicted below:

f(x)=wx+b

5 Where:

w is the weight vector perpendicular to the hyperplane.

X represents the feature vectors.

b is the bias.

Figure 2Support Vector Machine (SVM) classifiers for sentiment analysis

4 Sentiment Classification using Long Short Term Memory (LSTM).

An LSTM model will be developed to handle the complexities of sentence structures

104 that change over time, making it ideal for text data like tweets where context and word order are important. Initially, we create a feature vector that captures unique characteristics from the entire training dataset, representing each piece of text in a structured way. Unlike simpler models, the LSTM pays attention to the sequence of words, not just their occurrence or relevance.

For training, the dataset's text columns will serve as the input sequences, with their corresponding labels (good, bad, or neutral) as targets. These input texts are transformed into numerical vectors, often using embedding's like word2vec or GloVe, which capture more contextual information than basic frequency counts.

The LSTM algorithm processes these sequences, learning from the temporal dynamics

21 of the words. It's designed to remember important information and forget non-relevant

63 data through gates in its architecture, namely the forget gate, input gate, and output gate. This capability allows it to manage longer texts and understand the sentiment expressed better.

2 During the pre-processing phase, techniques such as tokenization and lemmatization are used to refine the input data. Tokenization breaks the text into tokens or words,

77 and lemmatization simplifies words to their base or root form, enhancing the model's ability to process natural language. Additionally, removing stop words and applying TF-IDF can help focus the LSTM on the most impactful parts of the text.

The LSTM network then classifies the sentiment of the text by outputting the probabilities of each sentiment class based on the learned weights in its neural network layers. The output layer typically uses a softmax function to classify the input text into categories such as negative, positive, or neutral based on the highest probability.

LSTMs excel in tasks where the context from earlier in the sequence is crucial for understanding the entire piece, making them particularly effective for comprehensive and nuanced sentiment analysis. This ability to remember and use past information makes LSTMs superior for complex sentence structures, enabling them to predict sentiments more accurately and handle nuances in language that simpler models might miss.

The mathematical representation of the convolutional operation is depicted below:

$g(x,y)=i=-a\sum aj=-b\sum bf(x-i,y-j)\cdot\ h(i,j)$

Where:

$g(x,y)$ is the output of the convolution at position (x, y).

$f(x-i,y-j)$ represents the pixel values of the input image.

$h(i,j)$ is the filter values, and the sums over iii and jjj represent the area of the filter.

Figure 3 Long Short Term Memory(LSTM) classifier for sentiment analysis

Explanation of the Algorithm Used

From Naïve Bayes

10 Naïve Bayes is a simple yet powerful machine learning algorithm based on Bayes
Theorem. It's perfect for quickly developing robust models, particularly in supervised
100 learning where a dataset is used to train the model. In practice, Naïve Bayes evaluates
the frequency of each word in the dataset to compute probabilities. For instance, it can
determine whether words in a text are likely to convey good, bad, or neutral
sentiments, assigning the label with the highest probability to each piece of text.

19 Setting up involves importing the necessary libraries and the dataset, followed by pre-
processing. This step cleans the text, removing any punctuation, tabs, and emoticons
that aren't needed. The next step is to extract features and labels such as good, bad,
36 and neutral sentiments from the data. Then, a bag of words approach is used to count
how often each word appears. Based on these counts, the algorithm calculates the
probability of each word under each label. Ultimately, after training on the dataset, the
Naïve Bayes model is ready to make predictions, helping to categorize new texts
based on their content.

Without eliminating the actual concept, the Naïve Bayes formula has been modified and tuned to enhance the model performance. The modified version of the Naïve Bayes formula is depicted below:

The conditional probability of different labels can be simplified by removing the denominator from the equation when we're trying to determine which label has the highest probability. In the equations we've discussed, 'y' represents the features, while 'x' is the label, which could be good, bad, or neutral. When we have multiple features, this relationship can be described as follows:

24 P(y | x) = P(a1, a2, ... an | y) × P(y)

Where,

P(a1, a2, ... an | y) × P(y) = P(a1 | y) × P(y) × P(a2 | y) × P(y).......P(an | y) × P(y)

Now, by calculating the probability of each word within the given labels, we can determine the conditional probability using this method. If we further modify the formula, it helps us refine our calculations,

74 The equation we're discussing is used to calculate the probability of a specific word Wk given a label, which could be good, bad, or neutral. In this equation, n represents the total number of words in the dataset under each label. nk refers to how often each word appears within these labeled categories. The term | Vocabulary| refers to the total number of unique words in the dataset, essentially a compilation of all the words used.

88

10

If there are no words in our vocabulary, the formula could potentially mislead us by leading to incorrect probabilities. To avoid the risk of dividing by zero, which would yield an infinite outcome, | Vocabulary| is included in the denominator of our equation. Additionally, if a word doesn't appear under a specific label at all (nk=0), the formula could result in a probability of zero, which isn't helpful. To address this, we add one to the numerator to ensure every word has at least a minimal impact on the probability.

Ultimately, this approach allows us to compute the probability of every word based on its label from the training dataset. Once trained, the model can then predict whether a given input is good, bad, or neutral, relying on the highest probability from these calculations.

The example below has been provided to clearly illustrate how the algorithm works.

Table 1 Sentiment Analysis Training Data of Naïve Bayes

Statements	Labels
You are awesome.	Good
Thank you.	Good
Will be back.	Neutral
You are so dumb	

Bad

Don't like you

Bad

You are jealous.

Bad

Now the bag of words will be:

['you', 'are', 'awesome', 'thank', 'will', 'be', 'back', 'fell', 'asleep', 'don’t', ‘like’, ‘jealous’]

Then the words and their corresponding frequencies will be listed as follows:

Table 2 Sentiment Analysis Bag of Words Frequency Table. Of word i

Statements

You

Are

Awesome

Thank

Will

Be

Back

so

Labels

You are awesome.

1

1

1

0

0

0

0

0

Good

Thank you.

1

0

0

1

0

0

0

0

Good

Will be back.

0

0

0

0

1

1

1

0

Neutral

You are so dumb

1

1

0

0

0

0

0

1

Bad

Don’t like you

1

0

0

0

0

0

0

0

Bad

You are jealous.

1

1

0

0

0

0

0

0

Bad

Table 3 Sentiment Analysis Bag of Words Frequency Table. Of word ii

Statements

Dumb

Don’t

Like

Jealous

Labels

You are awesome.

0

0

0

0

Good

Thank you.

0

0

0

0

Good

Will be back.

0

0

0

0

Neutral

You are so dumb

1

0

0

0

Bad

Don’t like you

0

1

1

0

Bad

You are jealous.

0

0

0

1

Bad

Now, the probability of good, bad, and neutral is:

P(good) = 2/6

P(neutral) = 1/6

P(bad) = 3/6

For calculating the probabilities for entire labels, the formula will be:

P(Wk| label)=n+| Vocabulary| nk+1

At first, for Good sentiments:

Total number of words in Good sentences (n) = 5

Total number of words in vocabulary (|Vocabulary|) = 12

15 P(you | good) = = 0.18 P(don’t | good) = = 0.06

P(are | good) = = 0.12 P(like | good) = = 0.06

P(awesome | good) = = 0.12 P(jealous | good) = = 0.06

15 P(thank | good) = = 0.12 P(will | good) = = 0.06

P(be | good) = = 0. 06 P(back | good) = = 0.06

P(so | good) = = 0.06 P(dumb | good) = = 0.06

Secondly, for neutral sentiments:

Total number of words in neutral sentences (n) = 3

Total number of words in vocabulary (|Vocabulary|) = 12

P(you | neutral) = = 0.07 P(don’t | neutral) = = 0.07

P(are | neutral) = = 0.07 P(like | neutral) = = 0.07

P(awesome | neutral) = = 0.07 P(jealous | neutral) = = 0.07

P(thank | neutral) = = 0.07 P(will | neutral) = = 0.13

P(be | neutral) = = 0. 13 P(back | neutral) = = 0.13

P(so | neutral) = = 0.07 P(dumb | neutral) = = 0.07

Finally , for Bad sentiments:

Total number of words in Bad sentences (n) = 10

Total number of words in vocabulary (|Vocabulary|) = 12

P(you | bad) = = 0.18	P(don’t | bad) = = 0.09
P(are | bad) = = 0.14	P(like | bad) = = 0.09
P(awesome | bad) = = 0.05	P(jealous | bad) = = 0.09
P(thank | bad) = = 0.05	P(will | bad) = = 0.05
P(be | bad) = = 0. 05	P(back | bad) = = 0.05
P(so | bad) = = 0.09	P(dumb | bad) = = 0.09

After training, the model is now capable of determining whether the given context is positive, negative, or neutral by evaluating the independent probabilities of words.

Text to classify: "You will be jealous"

Ygood = P(good) * P(you | good) * P(will | good) * P(be | good) * P(jealous | good)

= 1.296e-5

Yneutral = P(neutral) * P(you | neutral) * P(will | neutral) * P(be | neutral) *

P(jealous | neutral)

= 1.38e-5

Ybad = P(bad) * P(you | bad) * P(will | bad) * P(be | bad) *

P(jealous | bad)

= 2.025e-5

Since the probability of a sentence being bad is the highest, we can conclude that the sentence carries a bad sentiment.

This discussion illustrates how the Naïve Bayes theorem functions in solving natural language processing problems. Going forward, we will apply the same core principles of this algorithm to develop a sentiment analysis model. 23

4 From Support Vector Machine (SVM).

Support Vector Machine (SVM) is a powerful machine learning tool, especially useful for tasks like sentiment analysis. Its job is to classify data, such as text, into categories like good, bad, or neutral emotions. Imagine SVM as an expert artist carefully drawing the best possible line through a scatter of points, separating them into distinct groups with precision.

To get started, we prepare the dataset by importing the necessary tools and cleaning up the text. This involves removing things like extra punctuation, tabs, or emoticons anything that doesn't contribute to understanding the sentiment. Once cleaned, the text is broken down into features, often based on word frequencies or occurrences. These features are then converted into numbers, making them ready for analysis by the SVM. This preparation ensures the data is structured and ready for accurate classification. SVM works by creating a boundary, called a hyperplane, in a multi-dimensional space. Think of it as finding the perfect line through a cloud of points, where each point represents a piece of text like a tweet or a review. The position of each point is determined by the words it contains. The goal of SVM is to position this boundary so it leaves the most space, or margin, on either side. This creates a clear separation between groups, ensuring the distinctions between good, bad, and neutral sentiments are as sharp and reliable as possible.

In SVM for sentiment analysis, the decision function calculates the sentiment of a text by applying a formula to its data, helping classify it as good, bad, or neutral.

f(x)=wx+b

5 Where:

w is the weight vector perpendicular to the hyperplane.

X represents the feature vectors.

b is the bias.

66 In sentiment analysis using SVM, the sentiment of a text is determined by the sign of the functionf(x). Iff(x). is greater than zero, the text is classified as good; if it's less than 40 zero, it's classified as bad. The actual distance of f(x). From zero reflects how confident the model is in its prediction the farther from zero, the higher the confidence.

Table 4 Sentiment Analysis Training Data of SVM

Statements

Labels

You are awesome.

Good

Thank you.

Good

Will be back.

Neutral

You are so dumb

Bad

Don't like you

Bad

You are jealous.

Bad

The SVM model would process this data as follows:

Convert each statement into a feature vector. Example feature extraction (bag of words model):

"You" = 1, "are" = 1, "awesome" = 1, "thank" = 0, ..., "jealous" = 0

23 Train the SVM to find the hyperplane that maximally separates good and bad sentiments.

For the statement "You will be jealous":

Feature vector might look like: "You" = 1, "will" = 1, "be" = 1, "jealous" = 1

Calculatef(x). Using the trained SVM model.

Suppose the SVM model calculates the following (hypothetical values):

w= [0.2,−0.1,0.1,...,0.3]

b=−0.05b = -0.05b=−0.05

Then, f(x) for "You will be jealous" might be computed as:

f(x)=(0.2×1+−0.1×0+0.1×0+...+0.3×1)−0.05

71 Assume this computes to f(x)=0.45f(x) = 0.45f(x)=0.45

Given that f(x)>0, and it's greater than the margin threshold, the model classifies the statement as good.

In simple terms, SVM is excellent for sentiment analysis because it identifies the best boundary to separate good, bad, and neutral emotions. It does this by finding the widest possible gap between the classes, ensuring precise and confident predictions. This makes SVM reliable for distinguishing between different sentiments based on patterns in the data.

4 From Long Short Term Memory (LSTM).

The process of training a sentiment analysis model using an LSTM network involves several detailed steps, beginning with data preparation. Initially, a labelled dataset is created where each text sample is tagged with a sentiment, such as good, bad, or neutral. The text is then pre-processed to strip away non-essential elements like punctuation, emoji’s, and special symbols to focus solely on meaningful content.

39 Subsequent steps include tokenization, where text is split into individual words or

39 tokens, and padding, which standardizes the length of these sequences to ensure consistency across data inputs.

Each word in these sequences is transformed into a numeric vector using embedding techniques such as Word2Vec, GloVe, or an embedding layer within the neural network. These vectors capture the semantic meaning of words, facilitating the machine's understanding of language nuances. 12

The LSTM model processes this data through its specialized architecture, which includes three gates: the input gate, which filters incoming information; the forget gate, which discards non-essential information; and the output gate, which determines the next layer's output. The model leverages these gates to maintain important context and handle long-term dependencies in text data. 51 89

Training involves tuning the model's internal weights through a method known as backpropagation through time (BPTT). This technique adjusts the weights to minimize the error between the predicted sentiment and the actual sentiment label, enhancing the model's accuracy over time. Through these complex but systematic processes, the LSTM model becomes capable of accurately predicting sentiments based on textual inputs. 13

Table 5 Sentiment Analysis Training Data of LSTM

Statements

Labels

She enjoys movies

Good

He dislikes rain.

bad

Weather is fine.

Neutral

After preprocessing and tokenization, the vocabulary for this dataset might look like this:

['She', 'enjoys', 'movies', 'He', 'dislikes', 'rain', 'Weather', 'is', 'fine']

Assuming each word is transformed into an embedding vector, it could be represented as follows:

She → [0.2, 0.1, 0.3]

enjoys → [0.5, 0.6, 0.4]

movies → [0.3, 0.7, 0.6]

He → [0.1, 0.3, 0.2]

dislikes → [0.4, 0.5, 0.5]

rain → [0.6, 0.4, 0.2]

Weather → [0.3, 0.3, 0.5]

is → [0.2, 0.5, 0.4]

fine → [0.5, 0.4, 0.6]

For the sentence "She enjoys movies", the input sequence after embedding would look like:

[[0.2, 0.1, 0.3], [0.5, 0.6, 0.4], [0.3, 0.7, 0.6]]

The LSTM handles the sequence as described below:

19 Input Gate Operation:

73 The Input Gate's primary function is to determine the extent to which new information should be stored in the cell state. It performs this by calculating:

Formula:

33 it=(Wi[ht-1,xt]+bi)

Where:

it: Input gate activation at time ttt

81 σ: Sigmoid activation function that scales the output between 0 and 1

Wi: Weight matrix for the input gate

28 bi: Bias term for the input gate

ht−1: Previous hidden state

xt: Current input vector

Forget Gate:

The Forget Gate's role is to determine which parts of the previous cell state are no longer needed and should be forgotten:

Formula:

ft=(Wf[ht-1,xt]+bf)

Where:

ft: Forget gate activation at time ttt

σ: Sigmoid activation function

Wf: Weight matrix for the forget gate

Bf: Bias term for the forget gate

ht−1: Previous hidden state

xt: Current input vector

Cell State Update:

The cell state is updated based on the outputs from the input and forget gates:

Formula:

Ct=ftCt-1+itC~t

Where:

Ct: Updated cell state at time ttt

ft: Output from the forget gate

Ct−1: Previous cell state

it: Output from the input gate

C~t: Candidate cell state, representing the new memory content to be added.

Output Gate:

9 This gate controls what is transmitted from the current cell state to the next layer or the output:

Formula:

3 ot=(Wo[ht-1,xt]+bo)

Where:

ot: Output gate activation at time ttt

σ: Sigmoid activation function

Wo: Weight matrix for the output gate

17 bo: Bias term for the output gate

ht−1: Previous hidden state

xt: Current input vector

Hidden State:

86 The hidden state is updated based on the output from the output gate and the current cell state, determining the final output and the state passed to the next timestep:

Formula:

60 ht=ottanh(Ct)

Where:

ht: Hidden state at time ttt

ot: Output from the output gate

112 tanh(Ct): Hyperbolic tangent of the current cell state, providing the non-linear transformation to the output.

Pseudocode for the solution.

The pseudocode provided for addressing the problem domain is outlined below:

Initial Pseudocode of Naïve Bayes algorithm.

```
IMPORT NumPy
    IMPORT pandas
    IMPORT training dataset
        PRE-PROCESS dataset
        SELECT features
        CREATE vocabulary
        CALCULATE frequency
        CLASSIFY labels
TRAIN model
    APPLY the Naïve Bayes Classifier
```

```
CALCULATE probabilities
FUNCTION predict_text:
    DO
        INPUT text to classify
        APPLY model
        CALCULATE good probability
        CALCULATE bad probability
        CALCULATE neutral probability
        IF p(text | good)
            RETURN good
        ELSE IF p(text | neutral)
            RETURN neutral
        ELSE:
            RETURN bad
        END IF
    END DO
```

94 Updated Pseudocode of the Naïve Bayes algorithm.

```
  IMPORT pandas
IMPORT numpy as np
IMPORT train_test_split
IMPORT stopwords
IMPORT PorterStemmer
```

```
READ data from the CSV

SPLIT data into 80% train_data and 20% test_data

EXTRACT the neutral_data from the train_data having neutral sentiments

EXTRACT the positive_data from the train_data having positive sentiments

EXTRACT the negative_data from the train_data having negative sentiments

INIT stop_words

INIT punctuations


FUNCTION remove_punctuations_stopwords(text):

    FOR symbols in punctuations:

        REMOVE punctuations

    END FOR

    FOR each_word in text:

        REMOVE stopwords

    END FOR

    RETURN text

END


FUNCTION stem_words(text):

    INIT stemmed_words by stemming words

    RETURN stemmed_words

END
```

```
FUNCTION preprocess_text(text):
    INIT text by CALLING remove_punctuations_stopwords with text
    INIT stemmed_words by CALLING stem_words with text
    RETURN stemmed_words
END


INIT bag_of_words as list
INIT neutral_sentences as a list
INIT positive_sentences as a list
INIT negative_sentences as list


FUNCTION create_vocabulary(classified_data):
    FOR text in classified_data:
        EXTEND bag_of_words by preprocessing texts
        IF classified_data['sentiment'] is ['neutral']:
            APPEND to neutral_sentences by preprocessing texts
        END IF
        ELIF classified_data['sentiment'] is ['positive']:
            APPEND to positive_sentences by preprocessing texts
        END ELIF
        ELSE:
            APPEND to negative_sentences by preprocessing texts
        END ELSE
    END FOR
END

CALL create_vocabulary with neutral_data
CALL create_vocabulary with positive_data
CALL create_vocabulary with negative_data
SET bag_of_words as a set of bag_of_words

FUNCTION create_text_vector(sentences):
    INIT word_counts as list
    FOR sentence in sentences:
        INIT word_count as list
        FOR word in bag_of_words:
            COUNT word in sentence and APPEND to word_count
        END FOR
        APPEND word_count to word_counts
    END FOR
    RETURN word_counts
END

INIT neutral by CALLING create_text_vector with neutral_sentences
INIT positive by CALLING create_text_vector with positive_sentences
INIT negative by CALLING create_text_vector with negative_sentences
```

```
CALCULATE neutral_probabilities using neutral
CALCULATE positive_probabilities using positive
CALCULATE negative_probabilities using negative
SET neutral_probability as len(neutral_data)/len(train_data)
SET positive_probability as len(positive_data)/len(train_data)
SET negative_probability as len(negative_data)/len(train_data)

FUNCTION predict(sentence):
    INIT gross_neutral = neutral_probability
    INIT gross_positive = positive_probability
    INIT gross_negative = negative_probability
    INIT refined_words by pre-processing the sentence
    FOR word in refined_words:
        IF word in bag_of_words:
            CALCULATE gross_neutral
            CALCULATE gross_positive
            CALCULATE gross_negative
        END IF
    END FOR
    IF gross_neutral > gross_positive and gross_neutral > gross_negative:
        RETURN 'neutral'
    END IF
    ELIF gross_positive > gross_neutral and gross_positive > gross_negative:
        RETURN 'positive'
    END ELIF
    ELSE:
        RETURN 'negative'
    END ELSE
END


FUNCTION calculate_accuracy(raw_data):
    FOR each in length of raw_data:
        IF predict(row['selected_text']) is row['sentiment']:
            CALCULATE accuracy
        END IF
    END FOR
    RETURN "The accuracy of the model."
END


FUNCTION new_input():
    INIT text_to_classify as input from the user
    IF text_to_classify is not empty:
        RETURN predict(text_to_classify)
    END IF
END
```

```
CALL new_input
```

Initial Pseudocode of Support Vector Machine

```
IMPORT NumPy
IMPORT pandas
IMPORT training dataset
PRE-PROCESS dataset
SELECT features
    TRANSFORM data using TF-IDF
    NORMALIZE data
TRAIN model
    APPLY the SVM Classifier
        SET kernel type
        DEFINE hyperparameters
    TRAIN SVM with training data
    FUNCTION predict_text:
        DO
            INPUT text to classify
            PRE-PROCESS input text same as training data
            TRANSFORM input using same TF-IDF vocabulary
            APPLY model
            GET classification result
            IF classification score:
```

```
            RETURN class label
        ELSE:
            RETURN alternative class
        END IF
    END DO
```

Updated Pseudocode of Support Vector Machine (SVM).

```
IMPORT pandas
IMPORT numpy
IMPORT train_test_split
IMPORT nltk for stopwords and PorterStemmer
IMPORT TfidfVectorizer, SVC from sklearn
READ data from the CSV file
SPLIT data into 80% train_data and 20% test_data
INIT stop_words
INIT punctuations
DEFINE FUNCTION remove_punctuations_stopwords(text):
    CONVERT text to lowercase
    REMOVE punctuations
    EXCLUDE stopwords from the text
    RETURN clean text
END FUNCTION
```

```
DEFINE FUNCTION stem_words(text):
    STEM each word in text using PorterStemmer
    RETURN stemmed text
END FUNCTION

DEFINE FUNCTION preprocess_text(text):
    CALL remove_punctuations_stopwords with text
    CALL stem_words with cleaned text
    RETURN preprocessed text
END FUNCTION

APPLY preprocessing on train_data and test_data tweets
INIT TfidfVectorizer
FIT and TRANSFORM train_data tweets into TF-IDF matrix
TRANSFORM test_data tweets into TF-IDF matrix
INIT SVC model
TRAIN the SVC model on TF-IDF matrix of train_data
PREDICT labels for test_data
CALCULATE precision and f1_score
DISPLAY model performance
END
```

Initial Pseudocode of Long Short Term Memory (LSTM).

```
IMPORT NumPy
    IMPORT pandas
    IMPORT training dataset
        PRE-PROCESS dataset
            TOKENIZE text
            PAD sequences
            ENCODE labels
TRAIN model
    CREATE LSTM model
        DEFINE number of units
        SET dropout rate
        ADD fully connected layers
        DEFINE output layer based on the number of classes
    COMPILE model
        SET optimizer
        DEFINE loss function
        SET metrics
    TRAIN model on training data
        SET epochs
        SET batch size
FUNCTION predict_text:
    DO
```

```
        INPUT text to classify
        TOKENIZE input text
        PAD tokenized text
        APPLY LSTM model
        GET predicted class probabilities
        DETERMINE class with highest probability
            IF highest probability class score:
                RETURN class label
            ELSE:
                RETURN alternative class
        END IF
    END DO
```

Updated Pseudocode of Long Short Term Memory (LSTM).

```
IMPORT pandas
IMPORT numpy
IMPORT train_test_split
IMPORT tensorflow libraries for neural networks
READ data from the CSV file
14 SPLIT data into 80% train_data and 20% test_data
INIT Tokenizer for tweets
FIT tokenizer on train_data['tweets']
```

```
TRANSFORM tweets to sequences
PAD sequences to have the same length
INIT Label Tokenizer for labels
TRANSFORM labels to numerical format
INIT LSTM Model with layers: Embedding, LSTM, Dense
COMPILE the model
TRAIN the model on padded tweet sequences and numerical labels
EVALUATE the model on test_data
CALCULATE precision and f1_score
DISPLAY model accuracy and evaluation metrics
END
```

Flowchart of the solution

The diagram below illustrates the solution for the problem domain in a clear and simplified manner.

Flowchart of the Naïve Bayes algorithm.

Figure 4 Flowchart of the Naive Bayes Algorithm

Flowchart of the Support Vector Machine.

Figure 5 Flowchart of the Support Vector Machine (SVM).

Flowchart of the Long Short Term Memory.

Figure 6 Flowchart of Sentiment Analysis using LSTM Algorithm.

Development Process

Development process of Naïve Bayes algorithm.

The Jupyter notebook has been used to integrate Python code to develop the program 11 for implementing the proposed solution. The available libraries like Pandas, NumPy array, Scikit learn, and NLTK (Natural Language Toolkit) have been imported to alleviate dealing with the datasets and implementing operations over the data for preparing the model.

After importing the required libraries, the dataset consisting of more than 50,000 rows 20 of data is imported using Pandas and stored as the DataFrame. Now using the 101 'train_test_split' library from the Scikit-learn helps partition the dataset into the train and the test data. Then the train data are classified based on their specific labeled sentiments.

Next, texts' pre-processing is to be conducted at different levels. Firstly, the punctuations from the sentences are removed, followed by the removal of the stop words from the same texts. Each word of the sentence is stemmed and made ready for creating the bag of words.

97 Now, using each word from the sentences bag of words is created, and a list of sentences has unique sentiments like; neutral, good, or bad. From the list of 12 punishments, text vectors (frequency of words) compared to that of a bag of words are created. Then, the probability of the terms likely to be neutral, negative, or positive is calculated using the Naïve Bayes theorem. Also, the possibilities for the appearance of expressions are evaluated.

85 After that, the sentiment of the test data is evaluated, and the model's performance is determined by calculating the accuracy score. Ultimately, the probabilities for each word in a sentence are calculated using the Naïve Bayes classifier to predict the new input's sentiment. As a result, the system will return the opinion having a higher probability. The brief description of codes, along with their snippets, is depicted below:

110 Importing libraries and the CSV file.

7 Figure 7 : Importing libraries and the CSV file

Splitting the dataset

Figure 8 Splitting the Data

Classifying the train data.

Figure 9 Classifying the Train Data

53 Removing punctuations and stop words.

53 Figure 10 Removing Punctuations and Stop Words

Stemming the words

Figure 11 Stemming the Words

Pre-processing Dataset

Figure 12 pre-processing Dataset

45 Function to create the bag of words

45 Figure 13 Function to create the bag of words

Creating bag of words

Figure 14 Creating Bags of Words

Creating text vectors

Figure 15 Creating Text Vectors

Calculating probabilities

Figure 16 Calculating Probabilities

Achieve Result of Naïve Bayes algorithm.

The model has been developed so that after completing the training and testing, it highlights the model's accuracy. Moving forward, it asks the user in a loop to provide a sentence to predict the sentiment and continues to display the sentiment for the text unless the user wishes to stop. The results based on accuracy score and prediction are briefly described along with the code snippets.

Predicting the sentiment

Figure 17 Predicting the Sentiment

Calculating the accuracy score

Figure 18 calculating the accuracy score

52 Calculating the F1 Score and the Precision

52 Figure 19 Calculating the F1 Score and the Precision

Development Process of Support Vector Machine (SVM).

After importing the necessary libraries, the dataset comprising over 50,000 rows is loaded using Pandas and stored as a DataFrame. 35 The 'train_test_split' function from Scikit-learn is 50 utilized to partition the dataset into training and testing sets, ensuring that 80% of the data is used for training and 20% for testing. This partition enables the isolation of data for effective 42 model training and unbiased evaluation.

Once the data is divided, the training data are categorized according to their labeled sentiments, such as neutral, Good, or Bad. The next crucial step involves text preprocessing, which is conducted at several levels to prepare the text for the SVM model. Initially, all punctuations are removed from the sentences, followed by the elimination of stop words. Subsequently, each word in the sentences is stemmed to reduce it to its root form, simplifying the text and reducing its complexity.

Following the preprocessing phase, a bag of words is constructed using the cleaned and stemmed text. This bag of words serves as a foundation for creating text vectors. Using the TF-IDF (Term Frequency-Inverse Document Frequency) method, text vectors are generated. This technique transforms the text into a numerical format that highlights the importance of words based on their frequency across documents but penalizes words that are too common, thereby balancing word relevance.

The SVM model is then trained on these TF-IDF vectors. SVM operates by finding the optimal hyperplane that separates the data points in a high-dimensional space into their respective categories, based on their sentiments. This separation is key to the model's ability to classify new text data effectively.

After training, the sentiment of the test data is analyzed using the trained SVM model, and the performance of the model is assessed by calculating metrics such as accuracy, precision, and F1-score. These metrics provide insights into the effectiveness of the SVM model in classifying sentiments accurately.

Importing libraries and the CSV file.

Figure 20 Importing libraries and the CSV file.

Loading and Pre-processing Data.

Figure 21 Loading and Pre-processing Data.

Vectorization and Model Training.

Figure 22 Vectorization and Model Training.

Achieved Result of Support Vector Machine (SVM).

Prediction Function.

Figure 23 Prediction Function.

Additional Model Evaluation Accuracy.

Figure 24 Accuracy Score.

Precision and F1 Score Evaluation.

Figure 25 Precision and F1 Score Evaluation.

113 Development Process of Long Short Term Memory (LSTM).

After the necessary libraries are imported, the dataset containing over 50,000 rows is loaded using Pandas and stored as a DataFrame. Using the 'train_test_split' function from Scikit-learn, the dataset is divided into training and testing sets, with 80% allocated for training to ensure adequate data is available for learning, and 20% reserved for testing to validate the model's performance.

The training data are then categorized based on their labeled sentiments—neutral, Good, or Bad. Text preprocessing is crucial for the LSTM model and involves several detailed steps. Initially, all punctuation is removed from the text to reduce noise. Following this, stop words are eliminated to focus on more meaningful words in the text. Each word is also stemmed to its root form, simplifying the vocabulary and reducing dimensionality.

Subsequently, the preprocessed text is prepared for the LSTM model by tokenizing the words. This process converts the text into sequences of integers, where each integer represents a unique word in a dictionary created from the training data. These sequences are then padded to ensure they all have the same length, which is necessary for batching in neural network training, allowing the LSTM to efficiently process batches of input data.

The LSTM model, a type of recurrent neural network, is particularly suited for text processing due to its ability to remember long-term dependencies. It learns the sequence and context of words using gates and cells that regulate the flow of information. This capability enables it to capture the nuances of language that are crucial for accurate sentiment analysis.

1 The model is trained on the tokenized and padded text data, learning to classify
2 sentiments based on the patterns it discerns. After training, the LSTM model's
performance is evaluated on the testing set, and metrics such as accuracy, precision,
and recall are calculated to gauge its effectiveness.

For practical applications, when a new input is received, it undergoes the same pre-
99 processing, tokenization, and padding before being fed into the LSTM model. The
model predicts the sentiment based on its learned parameters, outputting the
sentiment classification that it determines to be most likely based on the sequential
context provided by the input text.

7 Importing libraries and the CSV file.

7 Figure 26 Importing libraries and the CSV file

Loading and Processing the Data

Figure 27 Loading and Processing the Data

Defining the LSTM model.

20

Figure 28 Defining the LSTM Model.

Training the LSTM Model

Figure 29 Training the LSTM Model.

120

Achieved Result of Long Short Term Memory (LSTM)

Prediction Function

Figure 30 prediction Function

Accuracy Score.

Figure 31 Accuracy Score of LSTM.

F1 Score and Precision Score.

Figure 32 F1 Score and Precision Score.

Conclusion

Analysis of the Work Done

Artificial Intelligence encompasses philosophy, psychology, and linguistics on top of computer science to develop intelligent systems. The task that generally requires human Intelligence can now be performed by intelligent agents (Duin & Bakhshi, 2017). Along with the rapid advancement in this field, the system that could imitate the simulation of human brains has been developed and concludes with promising results. Overall, this project is based on the research, analysis, and development of a sentiment analysis system under the Natural Language Processing (NLP) subfield of AI. NLP is a fascinating but deeply intricated field concerned with synthesizing and analyzing written or spoken languages. This field focuses on developing applications that could effortlessly and efficiently distinguish human writings and speeches. But particularly in this project, a wide variety of algorithms, their limitations, and mitigating measures has been discussed for developing a sentiment analysis system.

The essentiality for developing an intelligent system that could detect human attitudes, emotions, and behavior towards services, products, or events is crucial for the remarkable growth of business and similar fields. The necessity of deploying such a system to mitigate probable incidents using enormous data generated through the internet has been a critical reason behind the project.

This report presents detailed research on sentiment analysis in AI, including a proposed algorithm for developing similar applications. It covers various algorithms used to tackle sentiment analysis challenges, such as SVM, Naïve Bayes, CNN, and RNN. The report highlights the effectiveness of the LSTM, SVM, and Naïve Bayes algorithms in addressing issues like bias and tone detection. Additionally, it notes that the Naïve Bayes and SVM algorithms perform well with limited data and require less computational power. To enhance the Naïve Bayes algorithm's performance, techniques such as tokenization, lemmatization, removing stop words, and stemming are recommended. Converting words to vectors is also suggested to improve model accuracy. To further refine the accuracy of the LSTM model, efforts were made to enrich the dataset with a broader range of examples, optimize the neural network architecture, fine-tune essential hyperparameters like learning rate and batch size, and incorporate dropout and regularization strategies to mitigate overfitting.

In conclusion, this project delves into sentiment analysis within AI, highlighting the use of algorithms such as LSTM, SVM, and Naïve Bayes. LSTM excels in capturing long-term dependencies in text data, which is essential for understanding nuanced sentiments, while SVM and Naïve Bayes are particularly effective for smaller datasets. Enhancements such as tokenization have further improved Naïve Bayes, solidifying its utility in accurately classifying sentiments.

How the solution addresses the real-world problems

The way we consider others' opinions before making decisions is crucial. For example, when someone plans to buy a laptop, they often seek advice from friends or family and do extensive research through blogs, forums, and social media. They weigh the pros and cons based on other users' experiences online to make an informed purchase.

E-commerce companies are using machine learning algorithms to analyze customer sentiments, which helps them improve their products and services. These insights also help companies spot current market trends. Sentiment analysis applies broadly, identifying popular products, movies, and music by analyzing online reviews.

The impact of sentiment analysis extends beyond marketing to areas like brand perception and politics. Governments can use it to gather citizen feedback on policies, making necessary improvements. Major tech companies invest heavily in sentiment analysis to refine their business strategies and understand customer opinions on their brands. Some companies even offer sentiment analysis services to help others understand public trends about their offerings.

In summary, the model we're proposing is designed to tackle real-world challenges that industries face. By focusing on an independent probabilistic model, the accuracy and effectiveness of the analyzed opinions on specific topics will be significantly enhanced.

Further work

After completing our research, we've detailed the issues at hand, reviewed similar projects and their results, and explored methods to tackle these challenges in this report. We propose a solution for sentiment analysis that leverages a mix of algorithms, including Naïve Bayes, SVM, and LSTM. We've also discussed the limitations of these algorithms and suggested ways to enhance their performance. To clarify the implementation process, a pseudocode and a flowchart was included.

105 To date, the Jupyter Notebook, leveraging the Python programming language, has
69 been utilized for constructing sentiment analysis models. Relevant Python libraries such as NumPy, Pandas, Scikit-learn, NLTK, and TensorFlow have been imported and

96 employed in various stages of the model development process. Initially, the dataset was preprocessed—removing punctuations, eliminating stopwords, and stemming words—and imported into the system for training and testing. Using the Naive Bayes Classifier, probabilities of sentiments were calculated, achieving a prediction accuracy of 77.77%, with a precision of 76% and an F1 Score of 75%, indicating the model struggles with more complex sentence structures and could benefit from enhancements. Additionally, the Support Vector Machine (SVM) model was trained using TF-IDF vectors to classify sentiments, achieving an accuracy of 82%, a precision of 81%, and an F1 Score of 80%, with improved handling of linear separations in high-dimensional spaces. Meanwhile, the Long Short-Term Memory (LSTM) network was applied to capture long-term dependencies within text data, achieving an accuracy of 85%, a precision of 84%, and an F1 Score of 83%, enhancing the ability to understand context in sequences, which is vital for processing more complex sentence structures. Each model demonstrated distinct strengths and limitations, suggesting that while they provide robust frameworks for sentiment analysis, there is ample room for improvement to increase accuracy and handle more nuanced linguistic features. Moreover, the model can be integrated with the various applications for predicting the sentiment of the texts, which eventually may assist vendors or brands in realizing the public opinions upon their product or services. Also, in sectors where the response of the mass must be filtered, this model can come handy to integrate and analyse the results. Alternatively, to address complex scenarios requiring nuanced understanding and superior accuracy, integrating Transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) may be explored, as they

13 provide state-of-the-art performance in natural language processing tasks by capturing contextual relationships in text more effectively.

Further to add, as the NLP is skyrocketing, various techniques are being discovered to enhance the algorithms' performance, including aggressive filtering of texts, creating a continuous bag of words, adding a relation between words, predicting upcoming terms, and more. By applying these rules and techniques, the model will tune accurately to analyse the corpus's core intent.