

Nafisa Hossain Nujat

**EVALUATING THE PERFORMANCE OF
CLOUD BASED NLP APIS FOR
SENTIMENT AND EMOTION
DETECTION IN SARCASTIC TEXT**

Master's Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Zheyang Zhang
Konstantinos Stefanidis
May 2025

ABSTRACT

Nafisa Hossain Nujat: Evaluating the Performance of Cloud Based NLP APIs for Sentiment and Emotion Detection in Sarcastic Text

Master's Thesis

Tampere University

Master's Programme in Computing Sciences

May 2025

Sentiment and emotion analysis has become an integral component of Natural Language Processing (NLP), widely applied across domains such as social media monitoring and customer feedback analytics. Despite significant advancements, sarcasm detection remains a persistent challenge, as sarcastic expressions often convey meanings opposite to their literal content, posing difficulties not only for human interpretation but even more so for automated systems. This thesis evaluates the capabilities of four widely used cloud-based NLP APIs—Google Cloud NLP, AWS Comprehend, IBM Watson, and Azure Text Analytics—in interpreting sentiment and emotion in sarcastic text, using the publicly available News Headlines Dataset for Sarcasm Detection [57]. Preprocessing involved tokenizing headlines, removing noise such as punctuation, stopwords, and special characters, and formatting the data into JSON and CSV formats for API compatibility, while custom Python scripts automated the request-response workflow and collected structured output logs. Performance assessment was conducted through quantitative metrics—sentiment and emotion labeling accuracy, average response latency, and estimated cost per 15,000 samples—and qualitative factors such as ease of use, setup complexity, maintenance requirements, and analysis of common misinterpretations. The results indicate that the accuracy for all four APIs degrades significantly in the presence of sarcasm, often misinterpreting overtly positive or negative language at face value and missing the underlying ironic intent. These findings suggest that organizations working with sarcasm-heavy content—such as satirical journalism, online reviews, or political commentary—must account for these limitations and consider hybrid solutions, such as rule-based post-processing or context-aware enhancements. Ultimately, this research contributes empirical evidence to the discourse on sentiment analysis in complex linguistic environments, highlighting that while commercial NLP APIs offer scalable and user-friendly tools, they remain insufficient for nuanced text interpretation without supplementary strategies; future research could explore integrating API outputs with classifiers fine-tuned on sarcasm-specific datasets or leveraging transformer-based architectures better equipped to capture contextual subtleties.

Keywords: Sarcasm Detection, Sentiment Analysis, Emotion Detection, Natural Language Processing (NLP), Cloud-based APIs, Google Cloud NLP, AWS Comprehend, IBM Watson, Azure Text Analytics, API Evaluation, Sarcastic Text, Text Mining, Usability Assessment, Benchmarking, Latency, Cost Analysis

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

USE OF ARTIFICIAL INTELLIGENCE IN THIS WORK

Artificial intelligence (AI) has been used in generating this work:

- No
 Yes

I hereby declare, that the AI-based applications used in generating this work are as follows:

Application	Version
ChatGPT	4.0
Microsoft Copilot	4.0

Purpose of the use of AI

AI was used throughout the development of this thesis for various purposes, including:

- Refining and improving text clarity, structure, and coherence in abstract and all the chapters.
- Assisting in the generation and correction of code related to the thesis project, particularly in automating tasks such as data preprocessing and API evaluation.
- Providing guidance on phrasing, grammar, and style improvements throughout the document.
- Helping to ensure consistency across different sections and chapters, particularly in technical writing and presentation of results.

Parts of this work, where AI was used

The following chapters, sections, and tasks in this thesis involved AI assistance:

- Chapter 1: Introduction – Refining the background, problem statement, and research objectives.
- Chapter 2: Literature Review – AI helped generate references, summarize key studies, and suggest related work.

- Chapter 4: Methodology – Assisted in writing up detailed steps for API evaluation and preprocessing pipeline.
- Chapter 5, 6: Results and Discussion – AI was used to refine the presentation of results and suggest alternative explanations for observed trends.
- Chapter 7: Conclusion – Provided suggestions for concluding remarks and future research directions.
- Code (all scripts used for data preprocessing, API evaluation, and analysis).
- Tables and Figures – AI helped refine the wording in table captions and figure legends for clarity and precision.

Acknowledgement of risks

I hereby acknowledge, that as the author of this work, I am fully responsible for the contents presented in this thesis. This includes the parts that were generated by an AI, in part or in their entirety. I therefore also acknowledge my responsibility in the case, where use of AI has resulted in ethical guidelines being breached.

PREFACE

This thesis was completed as part of my Master's studies at Tampere University, within the field of Computing Sciences. The research focuses on evaluating the performance of cloud-based NLP services in detecting sentiment and emotion in sarcastic text, a topic that combines language understanding with real-world API evaluation.

I would like to express my sincere gratitude to my supervisor and examiner, Zheyang Zhang and Konstantinos Stefanidis, for their valuable guidance, constructive feedback, and consistent support throughout the project. Their insights helped shape the direction of this thesis and challenged me to refine my work.

I am also thankful to the faculty and fellow students in the faculty of Information Technology and Communication Sciences for the collaborative environment and engaging discussions that enriched my learning experience.

Finally, I want to thank my family and friends for their patience, encouragement, and continuous belief in me during this academic journey.

Oulu, 4th May 2025

Nafisa Hossain Nujat

CONTENTS

1.	Introduction	1
1.1	Aim of the Thesis	3
1.2	Research Questions	5
2.	Background and Literature Review	7
2.1	Sentiment and Emotion Detection	7
2.2	Sarcasm in Natural Language Processing	9
2.3	Approaches to Sarcasm Detection	10
2.4	Overview of Cloud NLP APIs	12
2.5	API Benchmarking on Sarcasm Detection	15
3.	Tools and Evaluation Metrics	16
3.1	Cloud NLP APIs	16
3.2	Programming Tools and Libraries	18
3.3	Evaluation Metrics	19
4.	Methodology	21
4.1	Dataset Description	21
4.2	Preprocessing and Formatting	25
4.3	API Testing Setup	27
4.4	API Testing Process	28
4.5	Data Collection and Storage	29
4.6	Error Handling and Logging	29
5.	Result Analysis	31
5.1	Quantitative Metrics	31
5.1.1	Accuracy	31
5.1.2	Latency	35
5.1.3	Cost	36
5.2	Qualitative Metrics	38
5.2.1	Error Patterns and Case Studies	38
5.2.2	Usability	40
6.	Discussion	42
6.1	Key Findings	42
6.2	Challenges in Sarcasm Detection	43
6.3	Implications for NLP in Real-world Applications	44
6.4	Recommendations for API Providers	46

7. Conclusion and Future Work	47
7.1 Findings	47
7.2 Limitations	48
7.3 Future Work	48
References	51

1. INTRODUCTION

Natural Language Processing (NLP) has made rapid advancements over the past decade, emerging as one of the most transformative fields within artificial intelligence. NLP allows machines to understand, interpret, and generate human language with increasing sophistication, bringing automation to tasks once thought to be uniquely humane. Initially dominated by rule-based systems, NLP has evolved significantly with the advent of statistical models, and more recently, deep learning. Key innovations, such as the introduction of Recurrent Neural Networks (RNNs)[33] and Transformer architectures[77], have enabled machines to handle complex linguistic tasks such as machine translation, speech recognition, and sentiment analysis.

The scope of NLP's capabilities has expanded significantly over the years, leading to its widespread adoption across industries. In customer service, NLP-powered chatbots and virtual assistants are now integral parts of customer experience, offering real-time support and personalized interactions. In healthcare, NLP algorithms are employed to analyze patient records, uncover trends in health data, and even assist in diagnosing mental health disorders by analyzing linguistic cues in text[43]. Similarly, sentiment analysis has become a core tool in sectors like marketing and politics, allowing organizations to gauge public opinion, monitor brand health, and even predict political outcomes based on social media conversations[60].

Among the many use cases in NLP, sentiment and emotion analysis stands out as one of the most impressive. By detecting the underlying sentiment in text—whether positive, negative, or neutral—NLP systems offer businesses, governments, and social organizations valuable insights into public perception, emotional states, and social trends. For instance, political analysts use sentiment analysis to track public opinion on candidates or policy proposals[30], while companies mine customer feedback to improve products and services[64]. Social media platforms like Twitter, Facebook, and Instagram generate a massive volume of user-generated content, making sentiment analysis an essential tool for parsing and understanding this data at scale.

However, despite significant advances in sentiment analysis, a persistent challenge remains: the detection and interpretation of sarcasm[38, 3, 10]. Sarcasm represents a unique linguistic phenomenon where the literal meaning of a statement contrasts sharply

with its intended meaning. For example, the sentence “Oh, great! Another meeting!” appears positive on the surface but conveys a negative sentiment in context. Sarcasm can introduce ambiguity that traditional sentiment analysis models often fail to handle. Models trained on surface-level features, such as word frequency or sentiment lexicons, are ill-equipped to understand the nuances of sarcasm[75].

The difficulty of sarcasm detection lies in the need to interpret not just the individual words but also the broader context, the speaker’s intent, and external cultural or situational factors. While humans can easily identify sarcasm using contextual knowledge, tone of voice, and shared experiences, machines struggle with these subtleties. NLP systems that focus solely on word-level analysis or syntactic structures are blind to the subtleties of irony, which are often conveyed through word choice, punctuation, or even the order of words[41].

Sarcasm is prevalent in many real-world domains, particularly in informal communication on platforms like social media, where people often express emotions through hyperbole, irony, and humor. For example, a sarcastic tweet about a product or service might be misclassified as positive, skewing the results of a sentiment analysis. This misinterpretation can have serious consequences for decision-making, such as misleading brands about customer satisfaction or causing political campaigns to misread voter sentiments[62]. Therefore, improving sarcasm detection in NLP is crucial for enhancing the reliability of sentiment analysis tools.

In response to the growing demand for scalable and efficient NLP solutions, many organizations have turned to cloud-based APIs offered by major technology companies. These services, including Google Cloud Natural Language API[28], Amazon Comprehend[2], Microsoft Azure Text Analytics[52], and IBM Watson NLU[35], provide pre-built NLP capabilities such as sentiment analysis, entity recognition, and syntax parsing. These APIs are marketed as accessible, cost-effective, and easy to integrate, offering solutions for businesses that need NLP without developing in-house models. They are designed to handle large volumes of data and provide quick results, making them ideal for enterprises looking to scale their operations.

Despite their widespread adoption, these cloud-based APIs have not been systematically tested for their ability to handle sarcasm—a critical component of sentiment analysis. Given that these APIs are used by organizations across industries for critical decision-making, it is essential to evaluate their performance in sarcasm-heavy environments. Do these systems misinterpret sarcastic comments? If so, what are the potential consequences for organizations that rely on these systems for sentiment analysis?

1.1 Aim of the Thesis

The primary goal of this thesis is to evaluate and compare the performance of commercial cloud-based NLP APIs in accurately detecting sentiment and emotion within sarcastic texts. While significant progress has been made in the development of specialized algorithms for sentiment analysis and sarcasm detection, much of the existing research has focused on novel algorithmic approaches, machine learning models, and academic benchmarks. However, there has been limited attention given to the practical application of these tools in real-world settings. Specifically, many businesses and organizations rely on off-the-shelf, cloud-based solutions for sentiment analysis, which often do not have the capability to understand complex linguistic constructs such as sarcasm. This thesis, therefore, shifts the focus from developing new methods to evaluating existing, commercially available tools in real-world contexts.

By testing widely used, cloud-based APIs, this study aims to provide valuable, actionable insights for businesses, developers, and researchers interested in deploying sentiment analysis tools in sarcasm-heavy domains. Sarcasm detection in particular poses significant challenges to sentiment analysis systems due to its reliance on contextual, tone-based, and often cultural cues, which are difficult for models to process effectively. Understanding how well these cloud-based solutions can cope with sarcastic texts in practical applications will help guide organizations in selecting the right tools for their needs, as well as highlight areas for improvement in future iterations of these services.

To systematically evaluate the performance of these APIs, the study employs the News Headlines Dataset for Sarcasm Detection[57]—a well-established dataset that contains real-world examples of sarcastic text. This dataset, commonly used in sarcasm detection research, is ideal for evaluating the ability of sentiment analysis models to identify sarcasm in news headlines. The dataset includes both sarcastic and non-sarcastic headlines, providing a diverse set of examples that will challenge the tested systems.

The evaluation covers both quantitative and qualitative aspects of each API's performance, with a focus on the following key dimensions:

- **Accuracy:** The core focus of this study is to assess the ability of each API to correctly classify sentiment and emotion in sarcastic and non-sarcastic samples. This aspect involves comparing the output of the APIs to human-annotated ground truth labels, ensuring that the systems are capable of not only identifying the correct sentiment in straightforward cases but also handling the nuanced nature of sarcasm. Accurately detecting sarcasm is a crucial component of this task, as sarcastic expressions often subvert the expected sentiment, making them particularly challenging for traditional sentiment analysis models.
- **Latency:** In addition to accuracy, the response time of each API is a critical

factor, especially for applications where real-time sentiment analysis is required. This metric evaluates the speed at which each API processes large volumes of data, which is vital for industries like customer service, social media monitoring, and marketing analytics. High latency can hinder the effectiveness of an API in real-time environments, so assessing the responsiveness of each platform under realistic usage scenarios will provide valuable insights for businesses looking to scale their operations.

- **Cost:** The financial feasibility of large-scale sentiment analysis implementation is another important consideration. This part of the evaluation focuses on a cost analysis based on the publicly available pricing models for each cloud API. By assessing the cost per request and any additional charges associated with scaling, the thesis provides organizations with a clearer understanding of the long-term financial implications of adopting these services. This analysis will help decision-makers weigh the balance between performance and cost, ensuring that they select a solution that meets both their operational and budgetary requirements.
- **Usability:** The ease of integration and user-friendliness of each API is a crucial aspect for developers and businesses looking to implement NLP tools without extensive technical expertise. This evaluation examines the quality of the documentation provided by each API, the ease of integrating the tool into existing workflows, and the system's ability to handle errors gracefully. A well-documented and user-friendly API can significantly reduce the development time and effort required to deploy sentiment analysis solutions. Additionally, understanding the limitations and common pitfalls of each API can help developers optimize their use in real-world applications.

The findings of this thesis aim to provide a comprehensive comparison of the strengths and weaknesses of commercial cloud-based NLP APIs in sarcasm detection. By evaluating these platforms in real-world conditions, this research offers practical recommendations for businesses, developers, and researchers who need to deploy sentiment analysis tools in environments where sarcasm and nuanced emotions are prevalent. The insights gained from this study will help identify which APIs perform best in sarcasm-heavy contexts, as well as highlight areas where improvements could be made to enhance the effectiveness and reliability of sentiment analysis tools in the future. Furthermore, the research will contribute to a growing body of knowledge on the practical challenges and opportunities associated with using cloud-based NLP services in complex, real-world applications.

1.2 Research Questions

This thesis is structured around three central research questions, each addressing a critical aspect of evaluating commercial cloud-based NLP APIs in the context of sarcasm detection. These questions are designed to guide the investigation and frame the analysis in a comprehensive manner. By examining the performance, operational efficiency, and limitations of these platforms, this research seeks to provide valuable insights that are both theoretically informative and practically actionable.

1. How accurately do cloud-based NLP APIs detect sentiment and emotion in sarcasm-heavy datasets?
2. How do these APIs compare in terms of performance metrics such as accuracy, latency, cost, and usability?
3. What are the common challenges and misinterpretations observed in sarcasm detection across these platforms?

The first research question is fundamental to the core objective of this thesis: evaluating the accuracy of cloud-based NLP APIs in detecting sentiment and emotion in sarcastic texts. Sarcasm, with its unique interplay between the literal meaning of words and the speaker's true intent, often creates ambiguity that traditional sentiment analysis systems struggle to resolve. This question seeks to measure the capacity of each API to identify the correct sentiment, particularly in texts that contain sarcastic expressions. By comparing the API-generated sentiment labels to human-annotated ground truth labels, this question provides a direct evaluation of how well these systems can handle the complexities of sarcastic language. The findings will reveal whether these commercial platforms can reliably detect not only overt sentiment but also the nuanced emotional cues that accompany sarcasm, which is a key challenge in NLP.

The second research question takes a holistic view of the APIs, evaluating their performance not only in terms of accuracy but also considering other critical factors that influence their practical use in real-world applications. While accuracy is undoubtedly important, the deployment of NLP tools in real-world scenarios often involves balancing multiple operational requirements. For example, how quickly can the APIs process large volumes of text (latency)? What are the costs associated with scaling these services for enterprise-level applications (cost)? Finally, how easy is it to integrate and use these APIs, and how well do they handle potential errors or limitations (usability)? By addressing this question, the thesis aims to provide a comprehensive evaluation that considers both the technical performance and the operational feasibility of each cloud NLP API, helping decision-makers assess which platform is best suited to their specific needs and constraints.

The third research question aims to delve deeper into the recurring patterns of failure

and technical limitations that may arise during sarcasm detection. Despite the growing sophistication of NLP models, sarcasm remains a significant challenge due to its dependence on context, tone, and cultural references—elements that many systems struggle to interpret. This question explores common misinterpretations or errors observed in sarcasm-heavy datasets, such as the misclassification of sarcastic remarks as positive or neutral when they are actually negative. By identifying these patterns, the thesis seeks to uncover underlying weaknesses in the APIs' ability to understand sarcasm and emotional complexity. Furthermore, this analysis will help identify the root causes of such failures—whether they are due to inadequate training data, limitations in model architecture, or challenges in detecting contextual cues. Understanding these common issues is essential for guiding future research and improvements in sarcasm detection systems.

Together, these three research questions provide a structured framework for evaluating the capabilities and limitations of cloud-based NLP platforms in handling sarcasm. They address the most pressing concerns for businesses, developers, and researchers seeking to deploy sentiment analysis tools in sarcasm-heavy environments, offering insights not only into the performance of these tools but also into the operational, financial, and technical challenges they face.

The structure of the following chapters builds upon the research questions outlined above. Chapter 2 reviews the theoretical foundations of sentiment analysis and sarcasm detection, providing an overview of key approaches, models, and challenges in the field as well as introduces the four commercial NLP APIs under evaluation—Google Cloud Natural Language, Amazon Comprehend, Microsoft Azure Text Analytics, and IBM Watson NLU. This chapter sets the stage for understanding the context in which cloud-based NLP APIs have been developed and deployed. Chapter 3 details the tools and metrics used to assess the APIs. Chapter 4 describes the experimental methodology, detailing the dataset preparation, the testing setup, and the evaluation protocols. Chapter 5 presents and analyzes the results, providing both quantitative performance metrics and qualitative insights into how each API handles sarcasm in text. Chapter 6 discusses the broader implications of the findings, offering recommendations for both researchers and practitioners. Finally, Chapter 7 concludes with a summary of the thesis and suggests potential directions for future research in sarcasm detection and NLP.

2. BACKGROUND AND LITERATURE REVIEW

This chapter surveys the foundational concepts and technologies relevant to sentiment, emotion, and sarcasm detection in natural language processing (NLP). It begins by introducing the distinction between sentiment and emotion analysis and explores their methodological evolution. This is followed by an in-depth discussion of sarcasm as a linguistic and computational challenge, its common types, and its impact on sentiment analysis. The chapter also reviews key approaches to sarcasm detection and highlights the growing use of commercial NLP APIs for these tasks. Finally, it examines benchmarking efforts and usability considerations that motivate the need for evaluating commercial systems in sarcasm-prone environments.

2.1 Sentiment and Emotion Detection

Sentiment analysis refers to the automatic identification of the polarity of opinions—positive, negative, or neutral—within a text[63]. Emotion detection extends this task by identifying specific affective states such as joy, sadness, anger, or fear[58]. Together, these tasks are critical in domains such as customer experience monitoring, political opinion tracking, and social media analytics[45, 58, 6].

While sentiment and emotion detection share overlapping objectives, they differ in terms of granularity and psychological modeling[59]. Sentiment analysis typically categorizes text into binary (positive/negative) or ternary (positive/negative/neutral) classes, whereas emotion detection deals with a broader and more nuanced taxonomy. These taxonomies are often based on psychological theories, such as Ekman’s six basic emotions[18] or Plutchik’s wheel of emotions[65], which represent discrete emotional states and their intensities[59]. In real-world deployments—like detecting dissatisfaction in customer reviews or gauging public mood during political campaigns—both types of analysis are often used in tandem[62].

Sentiment and Emotion Detection Methods

The evolution of sentiment and emotion detection mirrors broader shifts in natural language processing (NLP)—from rule-based systems to data-driven learning. Early rule-

based approaches relied on manually curated sentiment lexicons and heuristic rules to infer polarity or emotional tone. Tools like SentiWordNet[20] and VADER (Valence Aware Dictionary and sEntiment Reasoner)[34] assign scores to predefined lexical entries and combine them using syntactic rules. These systems are interpretable and fast, making them suitable for real-time applications. However, their reliance on static word lists and limited contextual understanding make them brittle in the face of domain-specific usage and figurative language such as sarcasm and negation[45].

With the emergence of large annotated datasets, supervised machine learning approaches became the dominant paradigm. Classifiers like Support Vector Machines (SVMs)[37], Naïve Bayes[51], and logistic regression[23] learn from labeled examples using engineered features such as n-grams, part-of-speech tags, and sentiment lexicon scores. While more adaptable than rule-based methods, these models struggle to capture deep semantic relationships and often perform poorly on informal or ambiguous text, particularly where meaning is shaped by context[6].

The advent of deep learning marked a major shift by enabling automatic feature extraction and modeling of complex language patterns. Convolutional Neural Networks (CNNs)[12] are effective for identifying local sentiment-bearing phrases, while Long Short-Term Memory (LSTM)[32] and Gated Recurrent Unit (GRU)[13] architectures better capture long-range dependencies in sequential data. These models have shown strong performance on unstructured, user-generated content such as tweets and chat transcripts[79]. Attention mechanisms further enhanced their capabilities by helping models selectively focus on sentiment-relevant parts of the input.

More recently, transformer-based models such as BERT[17] and RoBERTa[48] have revolutionized sentiment and emotion detection. These models use self-attention mechanisms to capture bidirectional context, allowing them to encode complex semantic and syntactic relationships. Pre-trained on large general corpora and fine-tuned for specific tasks, transformers achieve state-of-the-art results on benchmarks like SST-2 and GoEmotions[16]. They are particularly adept at handling subtle forms of sentiment, including implicit opinion, sarcasm, and irony[68]. However, despite their strong performance, these models remain computationally intensive and continue to face challenges in understanding pragmatic intent or sarcasm that depends on speaker context and discourse[38].

As sentiment and emotion detection tools mature, their application across languages has become increasingly important. Most early models and resources were designed for English, limiting their utility in multilingual or cross-cultural contexts[77]. Recent efforts have aimed at creating multilingual datasets and models, such as mBERT[17] and XLM-RoBERTa[14], which extend sentiment capabilities to a wide array of languages. However, linguistic diversity, cultural differences in emotion expression, and resource

scarcity in low-resource languages continue to pose significant challenges[9, 49].

2.2 Sarcasm in Natural Language Processing

Cambridge Dictionary defines Sarcasm as the use of remarks that clearly mean the opposite of what they say, made in order to hurt someone’s feelings or to criticize something in a humorous way[7]. Given that their literal meaning diverges significantly from the speaker’s true intention, sarcastic statements are often difficult to understand[41]. For example, the sentence “I just love getting ignored by customer support” expresses negative sentiment through seemingly positive wording. Detecting sarcasm in text is difficult because, more often than not, it requires extralinguistic knowledge, speaker context, and familiarity with social norms etc[71, 8].

A wide range of linguistic and contextual factors has been identified as contributing to sarcastic expression. A central role has been attributed to incongruity—a mismatch between expectations and reality—in the generation of sarcasm[24]. Sarcasm has been described as arising from disparities between what is desired, believed, or expected and what actually occurs. Incongruity between textual content and contextual information has also been recognized as a critical factor[78] and has been explored extensively, particularly in relation to sarcasm detection tasks[8]. This has been operationalized in computational models through the pairing of positive sentiment verbs with negative situation phrases, illustrating a common sarcastic construction (e.g., “I love waiting in long lines”)[71].

A framework outlining the necessary features of sarcastic utterances has also been proposed, in which sarcasm is characterized by one or more of the following five features [5]:

1. The utterance is evaluative in nature.
2. A reversal of valence exists between the literal and intended meanings.
3. Semantic incongruity is present with contextual knowledge, including common-sense or shared background information.
4. The utterance is directed at a specific target.
5. The utterance maintains relevance to the surrounding communicative scenario.

These characteristics highlight that sarcasm is both pragmatic and context-dependent, relying on shared knowledge and cultural norms between speaker and listener.

Sarcasm can also manifest in multiple linguistic forms, each of which poses unique detection challenges:

- **Hyperbole:** Exaggeration used to emphasize absurdity or frustration (e.g., “That meeting changed my life”)[8].

- **Deadpan sarcasm:** Delivered in a flat tone, relying on emotionless presentation that is difficult to infer from text (e.g., “Just fantastic,” stated plainly in response to failure)[5].
- **Rhetorical sarcasm:** Questions that imply the opposite of what they literally ask (e.g., “Isn’t this just perfect?” during a mishap)[71].
- **Contextual irony:** Sarcasm that depends on external context or prior discourse for interpretation (e.g., commenting “Great weather today!” during a storm)[78].

The diversity of sarcastic expression, combined with the reliance on nuanced contextual and pragmatic information, makes sarcasm a major obstacle for text analysis systems. In particular, sarcasm complicates sentiment analysis by introducing polarity flips, where seemingly positive language is used to express negative sentiment, or vice versa. This misalignment between literal and intended sentiment causes traditional sentiment classifiers to misinterpret the text, leading to flawed outputs across applications[3, 38].

2.3 Approaches to Sarcasm Detection

Early approaches to sarcasm detection primarily relied on manually defined rules and sentiment-incongruity patterns. For instance, methods were introduced for identifying sarcastic expressions that juxtapose positive sentiment words with negative context—an implementation of the “context incongruity” theory[71]. Although interpretable and easy to implement, these rule-based models lacked adaptability and failed to generalize to more ambiguous or context-rich cases.

Subsequent methods introduced traditional machine learning algorithms such as Support Vector Machines (SVMs), decision trees, and logistic regression. These systems relied on handcrafted features such as n-grams, punctuation patterns, part-of-speech tags, emoticons, and interjections [15, 26]. While these techniques improved upon rule-based approaches by learning from labeled data, they were still limited in their ability to capture nuanced semantics and struggled with out-of-domain generalization. The reliance on shallow features hindered their effectiveness in real-world, diverse contexts [40].

The emergence of deep learning introduced more sophisticated models capable of learning semantic and syntactic patterns automatically. A notable example is the CNN-LSTM hybrid model, which leveraged convolutional layers to capture local word patterns and long short-term memory (LSTM) layers to model sequential dependencies[25]. These models showed significant performance improvements, particularly in processing informal and noisy data from social media platforms.

Further advancements were made with the incorporation of attention mechanisms and memory networks. These architectures enabled models to dynamically focus on salient parts of an input, such as contrastive phrases or sentiment reversals, which are often

characteristic of sarcasm[76]. Attention-based models demonstrated enhanced ability to recognize ironic intent by identifying key contextual cues[55].

Transformer-based architectures have since become the dominant paradigm in sarcasm detection. Pre-trained language models such as BERT, RoBERTa, and XLNet have achieved state-of-the-art results on multiple benchmark datasets, including SARC[42] and iSarcasmEval[22]. These models generate contextualized embeddings that allow for deeper interpretation of word meanings within their surrounding context. Some approaches further enhance performance by incorporating auxiliary information such as conversation history, speaker identity, or commonsense knowledge[31, 44].

Recognizing the limitations of text-only systems, recent research has explored multimodal sarcasm detection. The MUSTARD dataset[10], for example, provides aligned audio-visual and textual data, enabling models to leverage prosodic and facial cues alongside text. Multimodal architectures have been especially effective for detecting forms like deadpan sarcasm, which are difficult to capture using text alone[69]. However, these systems face practical barriers, including high annotation costs, increased computational demands, and challenges related to real-time audio-visual input availability.

Table 2.1 summarizes the major sarcasm detection approaches discussed above, highlighting their respective strengths and limitations.

Despite significant progress, sarcasm detection remains an open problem due to several key challenges:

- **Contextual Incompleteness:** Most models operate on isolated sentences, lacking access to preceding discourse or speaker history[38].
- **Cross-Domain Transferability:** Models trained on Twitter data often perform poorly on Reddit, forums, or news media due to domain-specific language and tone[42].
- **Annotation Ambiguity:** Sarcasm is inherently subjective and context-sensitive, making it difficult to annotate consistently, especially with crowd-sourced data[50].
- **Multilingual Limitations:** Existing models are predominantly English-centric, despite sarcasm’s widespread cultural and linguistic variability[21].

To address these limitations, current research is exploring hybrid models that combine symbolic reasoning, deep learning, and multimodal inputs. There is also a push toward developing better annotation frameworks, including guidelines that reduce subjectivity and improve inter-annotator agreement[67]. Techniques such as few-shot sarcasm learning, cross-lingual transfer, and commonsense-augmented models using external knowledge bases such as ConceptNet[46] and ATOMIC[72] are being investigated, which aim to improve inference about implicit meaning. Furthermore, integration of external knowledge bases and commonsense reasoning engines is being pursued to support deeper

Approach	Pros	Cons
Rule-Based Models[71, 4]	Simple and interpretable; easy to implement with domain knowledge	Poor generalization; fails with ambiguous or context-rich expressions
Traditional Machine Learning Models (SVM, Decision Trees, etc.)[26, 15, 73]	Learns from labeled data; effective with well-engineered features	Relies on shallow, handcrafted features; poor cross-domain generalization
Deep Learning Models (CNN, LSTM)[66, 25]	Learns complex semantic and syntactic patterns; handles noisy text well	Needs large labeled data; struggles with subtle or nuanced sarcasm
Attention-Based Models[76, 54]	Focus on salient parts of input; better at modeling contrastive phrases	Computationally expensive; may still require external context
Transformer-Based Models (BERT, RoBERTa, XLNet)[29, 68]	Achieves state-of-the-art accuracy; captures deep contextual meaning	Limited commonsense reasoning; domain and cultural variance issues remain
Multimodal Models[11, 74]	Utilize prosody, facial cues, and text; strong performance on audiovisual sarcasm	High resource requirements; limited scalability in real-time settings
Hybrid and Knowledge-Augmented Models[47, 56]	Combine symbolic reasoning, external knowledge, and neural models; improved robustness	Still emerging; complex integration; no standard evaluation framework yet

Table 2.1. Comparison of Sarcasm Detection Approaches

contextual understanding[44].

2.4 Overview of Cloud NLP APIs

Cloud-based Natural Language Processing (NLP) platforms have become increasingly integral to both commercial products and academic research, especially for tasks such as sentiment and emotion detection. Major technology vendors—Google Cloud, Amazon Web Services (AWS), IBM, and Microsoft—offer turnkey NLP solutions through RESTful APIs that abstract away the complexity of building, training, and deploying machine learning models[28, 2, 35, 52]. These services provide pre-trained language models, scalable infrastructure, and support for multiple programming languages and data formats, significantly lowering the barrier to entry for advanced text analytics[19].

Despite their accessibility and widespread use, general-purpose NLP APIs demonstrate considerable limitations when applied to nuanced linguistic phenomena such as sarcasm—an issue that even some vendors explicitly acknowledge in their documentation.

For example, Microsoft Azure's Text Analytics service states that its sentiment analysis model may not reliably detect sarcasm due to the absence of contextual, tonal, and social cues typically required to understand such expressions [53]. Similarly, Google Cloud's Natural Language API documentation cautions against overreliance on confidence scores, especially in moderation and sensitive applications, indirectly reflecting the API's challenges in handling figurative or context-dependent language [27].

Detecting sarcasm involves more than analyzing the **lexical content** (i.e., the words themselves); it also requires consideration of **contextual features** (such as preceding discourse or shared background knowledge), **pragmatic elements** (including speaker intent, tone, and social cues), and **discursive structures** (how language is organized within a broader conversation or narrative)[8, 1]. However, most commercial NLP APIs fall short of capturing these nuanced layers of meaning. Consequently, sarcastic expressions are often misinterpreted as genuine sentiment, limiting the reliability of these tools in contexts where irony and sarcasm are common—such as social media, political discourse, and satirical content.

Google Cloud Natural Language API

Google's Natural Language API offers sentiment analysis by returning a sentiment **score** (ranging from -1.0 to $+1.0$) and a **magnitude** that quantifies the overall emotional intensity in a document or sentence. Internally, the API uses machine learning models trained on a mixture of web, news, and conversation-based datasets[28].

Despite its strong multilingual support and enterprise adoption, the API does not account for sarcasm or figurative language. Google Cloud benefits from a mature developer ecosystem with comprehensive SDKs, detailed tutorials, and active Stack Overflow discussions. It is also cited in many benchmarking literature, particularly in emotion or sentiment-focused comparisons [19, 61].

Amazon Comprehend

Amazon Comprehend offers sentiment detection by classifying input text into one of four categories: **positive**, **negative**, **neutral**, or **mixed**. While AWS does not disclose the architecture, patent filings and indirect documentation indicate that the models are based on deep neural networks using token-level embeddings, attention mechanisms, and potentially RNN or transformer-based classifiers[2]. Comprehend was originally designed to process documents in business and customer feedback contexts.

The API does not support figurative language recognition or sarcasm modeling. Its predictions rely heavily on lexical polarity and syntactic patterns. However, Amazon provides a feature called ***Comprehend Custom***, which allows developers to train

domain-specific classifiers using AutoML-style interfaces. It is deeply integrated into the AWS ecosystem, enabling seamless deployment alongside Lambda, S3, and SageMaker. Documentation is thorough and developer tooling is extensive.

IBM Watson Natural Language Understanding

IBM Watson NLU is among the most feature-rich commercial NLP services, offering sentiment analysis, emotion detection (across five classes: joy, anger, sadness, fear, and disgust), entity linking, and concept extraction. According to IBM's whitepapers and public documentation, Watson NLU uses a hybrid architecture combining rule-based heuristics, deep learning models, and statistical co-occurrence mechanisms [35]. For sentiment tasks, it employs bidirectional RNNs with attention layers and LSTM variants, augmented with lexical databases for affective polarity.

Watson NLU distinguishes itself by supporting aspect-based sentiment analysis at the entity level and offering sentence-wise breakdowns. Despite these strengths, it does not explicitly model figurative or ironic expressions. IBM does, however, provide a strong customization toolkit via Watson Studio, allowing developers to retrain models with user-supplied labeled data.

Although Watson is less prominent than AWS or Azure in developer communities, it is highly regarded in academic evaluations for its rich emotion classification, particularly in multilingual contexts.

Microsoft Azure Text Analytics

Microsoft Azure's Text Analytics API provides sentiment analysis at document, sentence, and aspect level. It classifies sentiment as **positive**, **neutral**, or **negative**, with confidence scores. Microsoft has publicly stated that the underlying models are based on transformer architectures fine-tuned for various domains, including social media, product reviews, and customer service interactions [52]. It is likely that Azure's stack builds upon variants of BERT or XLM-R, consistent with Microsoft's open-source work on MT-DNN and DeBERTa.

Azure provides detailed transparency notes confirming that its sentiment model does not recognize figurative speech, sarcasm, or irony. However, it is deeply embedded into enterprise ecosystems and benefits from strong documentation, Azure DevOps integration, and support for over 20 programming languages. It has been included in multiple benchmark studies for its robust performance on generic tasks but not yet verified in sarcasm detection contexts[19, 36].

Customization and Fine-Tuning

Among the four providers, only Amazon Comprehend and Google Cloud offer limited customization via AutoML or custom classification models. However, these still require labeled datasets and lack access to low-level architectural tuning.

In contrast, sarcasm detection research increasingly favors fine-tuned open-source transformer models such as BERT, RoBERTa, or DeBERTa. These models have been shown to outperform commercial APIs when adapted to sarcasm detection tasks using domain-adaptive pretraining, multi-task learning, or sarcastic intent modeling[11, 70].

2.5 API Benchmarking on Sarcasm Detection

Despite their widespread adoption, commercial NLP APIs have not been systematically evaluated for sarcasm detection. Most existing evaluations focus on sentiment classification accuracy in generic domains such as product reviews or news articles, often neglecting linguistically complex categories like sarcasm[39].

Moreover, proprietary performance benchmarks provided by vendors lack methodological transparency, and academic studies evaluating these APIs on sarcasm-specific datasets are virtually absent. As a result, practitioners often deploy these tools unaware of their limitations in irony-prone content such as tweets, Reddit threads, or conversational transcripts.

This gap is critical in domains where accurate sentiment interpretation is essential, such as customer support, political discourse monitoring, and content moderation. Misclassification of sarcastic expressions as genuine sentiment can lead to biased analyses, inappropriate responses, or failure to detect harmful content[26, 31].

Future benchmarking efforts should evaluate these APIs on well-established sarcasm detection datasets such as SARC[42], MUsTARD[10], and iSarcasmEval[22]. Metrics like precision, recall, and F1-score for sarcasm-specific classes would provide a more realistic view of their capabilities and limitations. Comparative analyses with fine-tuned transformer baselines could further guide deployment decisions in sarcasm-sensitive applications. This study aims to step in that direction by evaluating the aforementioned cloud NLP APIs on a news headline dataset and benchmarking on metrics such as: Accuracy, Precision, Recall and F1-score.

3. TOOLS AND EVALUATION METRICS

This chapter outlines the technical infrastructure, cloud-based NLP services, and evaluation methodologies used to benchmark sentiment and emotion detection capabilities. The study adopts a black-box evaluation approach, focusing exclusively on the output quality from each API without delving into model internals, architectures, or training data.

3.1 Cloud NLP APIs

This evaluation compares four major commercial NLP services: **Google Cloud Natural Language API**, **AWS Comprehend**, **IBM Watson Natural Language Understanding**, and **Azure Text Analytics**. Each service was accessed via RESTful HTTP endpoints, with consistent input data formats and preprocessing steps applied to ensure a fair and unbiased evaluation.

Integration with Google Cloud NLP was relatively straightforward, leveraging API key-based authentication and detailed documentation. However, a known issue involving out-of-range float values caused JSON serialization failures after processing approximately 26,500 samples. To maintain consistency with the other platforms and avoid this problem, the dataset was truncated to 15,000 samples. Sample outputs from Google Cloud NLP, including ground truth sentiment labels against API generated magnitudes, are shown in Table 3.1. In this API, the score ranges from -1.0 (most negative/sarcastic) to 1.0 (most positive/non-sarcastic), representing the emotional leaning of the text, while the magnitude indicates the overall emotional intensity regardless of polarity.

Index	Text	Label	Score	Magnitude
2929	man nods knowingly mechanic	1	0.0	0.0
14874	new polls increase fears midterm elections wave politicians	1	-0.6	0.6
3477	innocent man unrepentant	1	-0.3	0.3
2042	hillary clinton holds infant grandson upside ankle front convention crowd	1	0.3	0.3
625	remembering lynn walker huntley	0	0.2	0.2

Table 3.1. Sample responses from Google Cloud NLP

AWS Comprehend required a simple setup using AWS access and secret keys, with the additional need to explicitly specify a region in each request. The evaluation was also capped at 15,000 samples for consistency. Table 3.2 presents representative outputs from AWS Comprehend against ground truth labels, detailing sentiment and confidence scores. The API categorizes sentiment into three discrete classes: POSITIVE, NEGATIVE, and NEUTRAL. Each sentiment label is accompanied by confidence scores (ranging from 0 to 1) that reflect the model's certainty in its classification across all categories. The final sentiment is chosen based on the highest confidence score.

Index	Text	Label	Sentiment	Pos	Neg	Neutral
10302	austria germany open borders migrants offloaded hungary	0	NEUTRAL	0.003	0.403	0.594
4386	thing distracting healthy self-actualized lifestyle garners 240 emmy nominations	1	NEUTRAL	0.147	0.165	0.642
174	annoyed movers werent expecting client belongings	1	NEGATIVE	0.004	0.962	0.034
823	fox news guest blames liberals innercity violence	0	NEGATIVE	0.002	0.628	0.369
7753	freakonomist keeps close eye ge stock versus height mexican weightlifters	1	NEUTRAL	0.327	0.092	0.564

Table 3.2. Sample responses from AWS Comprehend

IBM Watson offered a quick integration experience using its API key and endpoint. The service returned detailed JSON responses that enabled richer downstream analysis, particularly useful for psychological or affective studies. Representative results are listed in Table 3.3. Here, sentiment analysis provides a sentiment polarity label—positive,

negative, or neutral—along with a confidence score from -1.0 to 1.0 against the actual labels from the dataset. A score closer to +1.0 reflects strong positive sentiment, while values closer to -1.0 indicate strong negative sentiment. Neutral sentiment is typically scored near 0.

Index	Text	Label	Sentiment	Score
8020	pretty obvious friend look like old	1	positive	0.678
1826	dog finds absolutely perfect place shit	1	positive	0.446
13828	students get suspended school	0	neutral	0.000
4219	frank gehry music liquid architecture	0	neutral	0.000
5890	daniel craig takes home pretty good actor award	1	positive	0.912

Table 3.3. Sample responses from IBM Watson

Azure Text Analytics required input data to be structured with fields for ID, language, and text. While the API returned nested JSON responses, enabling expressive outputs, it also introduced challenges, such as more frequent formatting and validation errors during batch submission. Sample responses from Azure’s service compared to the ground truth labels are included in Table 3.4. Azure assigns one of three sentiment labels—positive, negative, or neutral—based on three probability scores. These scores indicate the likelihood of the text being perceived as expressing each sentiment, and the label with the highest score is selected as the overall sentiment classification.

Index	Text	Label	Sentiment	Pos	Neg	Neutral
3498	high school student whines way 40 gpa	1	neutral	0.02	0.39	0.59
8868	fun fall recipes joy bauer video	0	positive	1.00	0.00	0.00
6110	bittersweet goodbye pregnancy	0	neutral	0.27	0.32	0.41
6072	local neurotic undergo invasive 32000hourlong therapy procedure	1	neutral	0.00	0.01	0.99
8383	course hair stylist remembers gina	1	neutral	0.17	0.01	0.81

Table 3.4. Sample responses from Azure Text Analytics

3.2 Programming Tools and Libraries

The evaluation pipeline was developed using Python 3.9 and relied on a variety of widely adopted libraries for data handling, API communication, and performance measurement. The tools and their primary purposes are as follows:

- **pandas**: For data wrangling, filtering, and tabular manipulation.
- **requests**: To interface with the cloud APIs via HTTPS requests.
- **json, csv**: For serialization and conversion between structured formats.
- **nlTK**: Used for basic natural language preprocessing tasks, such as tokenization and stopword removal.
- **logging**: Facilitated structured logging of runtime events, warnings, and exceptions.

To ensure consistency across APIs and minimize noise from non-linguistic artifacts, input data was cleaned and standardized prior to submission.

3.3 Evaluation Metrics

The evaluation was guided by four primary metrics:

Accuracy

Accuracy quantifies the proportion of predictions that matched the ground truth labels. For APIs returning continuous scores, decision thresholds were applied to map them into discrete sentiment categories.

$$\text{Accuracy} = \frac{\# \text{ Correct Predictions}}{\text{Total Samples}}$$

Accuracy results are presented and analyzed in Section 5.1.1.

Latency

Latency measures the time taken by each API to process a single request. The average latency was computed as follows, where n is the number of API requests:

$$\text{Avg Latency} = \frac{1}{n} \sum_{i=1}^n (t_{\text{response},i} - t_{\text{request},i})$$

Latency comparisons are discussed in Section 5.1.2.

Cost

Cost evaluation was based on public pricing data for each API, assuming 15,000 processed samples and accounting for any free-tier limits.

Detailed cost findings are presented in Section 5.1.3.

Usability

Usability was assessed qualitatively, based on:

- Ease of setup and integration
- Clarity and completeness of documentation
- Quality of error messages and debugging support

Comparative usability assessments appear in Section 5.2.2.

4. METHODOLOGY

This chapter describes the experimental design used to benchmark the sentiment and emotion detection capabilities of the four cloud-based NLP APIs in the presence of sarcastic text. The methodology includes dataset preparation, preprocessing routines, API integration, testing logic, and data handling strategies. Emphasis is placed on ensuring consistency, reproducibility, and fairness across platforms.

4.1 Dataset Description

The dataset used in this study is the **News Headlines Dataset for Sarcasm Detection**[57], curated by Rishabh Misra. It contains a total of 28,619 English news headlines sourced from both satirical outlets (primarily *The Onion*) and non-satirical sources (such as *The Huffington Post*). Each entry is annotated with a binary label indicating whether the headline is sarcastic (label 1) or non-sarcastic (label 0).

The dataset is particularly suitable for evaluating sentiment detection tools in cases where literal sentiment may differ from implied sentiment—a common characteristic of sarcastic text. This makes it valuable for probing the robustness and nuance sensitivity of commercial NLP APIs.

Distribution Overview

The dataset exhibits a near-balanced distribution between the two classes, as summarized below:

- **Sarcastic (Label 1):** 13,663 headlines (47.7%)
- **Non-Sarcastic (Label 0):** 14,956 headlines (52.3%)

Figure 4.1 illustrates this distribution graphically.

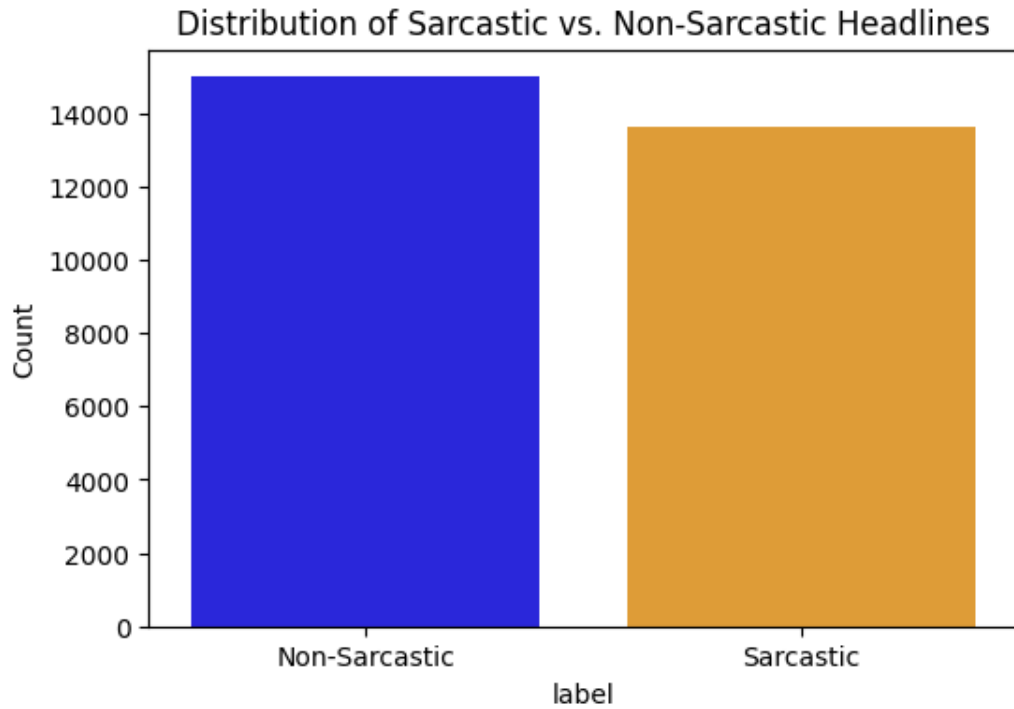


Figure 4.1. Distribution of sarcastic vs non-sarcastic headlines in the dataset.

This balance supports fair and unbiased evaluation across both sarcasm-aware and standard sentiment analysis models.

Word and Character Statistics

To further understand the structure of the headlines, summary statistics for word and character counts are presented in Table 4.1. These indicate that the average headline contains approximately 10 words and 62 characters, with a maximum of 151 words and 926 characters in a single headline. Figure 4.2 also shows a histogram of word counts across the dataset. Most headlines contain between 5 and 15 words, reflecting the concise and attention-grabbing nature of typical news headlines. A kernel density estimate (KDE) overlay provides a smoothed view of the distribution.

Metric	Word Count	Character Count
Count	28,619	28,619
Mean	10.05	62.31
Standard Deviation	3.39	20.73
Minimum	2	7
25th Percentile	8	49
Median (50th)	10	62
75th Percentile	12	75
Maximum	151	926

Table 4.1. Summary statistics for word and character counts across all headlines.

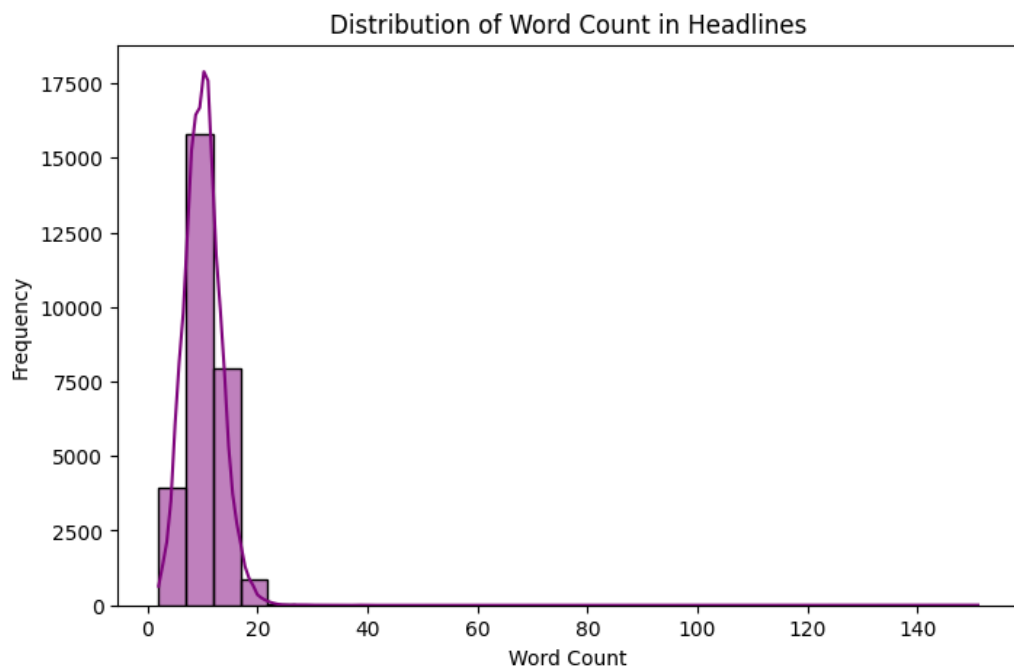


Figure 4.2. Distribution of Word Count in Headlines

Word Cloud Visualization

To gain qualitative insights into the vocabulary distribution, word clouds were generated separately for sarcastic and non-sarcastic headlines. The most frequently occurring words in each class are visually represented in Figures 4.3 and 4.4.

label	text	article_link
1	thirtysomething scientists unveil doomsday clock of hair loss	The Onion article link
0	dem rep. totally nails why congress is falling short on gender, racial equality	The Huffingtonpost article link
0	eat your veggies: 9 deliciously different recipes	The Huffingtonpost article link
1	inclement weather prevents liar from getting to work	The Onion article link
1	mother comes pretty close to using word 'streaming' correctly	The Onion article link

Table 4.2. Sample entries from the News Headlines Dataset for Sarcasm Detection

4.2 Preprocessing and Formatting

To ensure consistency and minimize noise across all inputs to the sentiment APIs, a unified preprocessing pipeline was implemented. This step was crucial to reduce variability in text length, casing, and extraneous tokens—while retaining enough linguistic signal to preserve sarcasm-related features. The overall structure of this pipeline is illustrated in Figure 4.5.

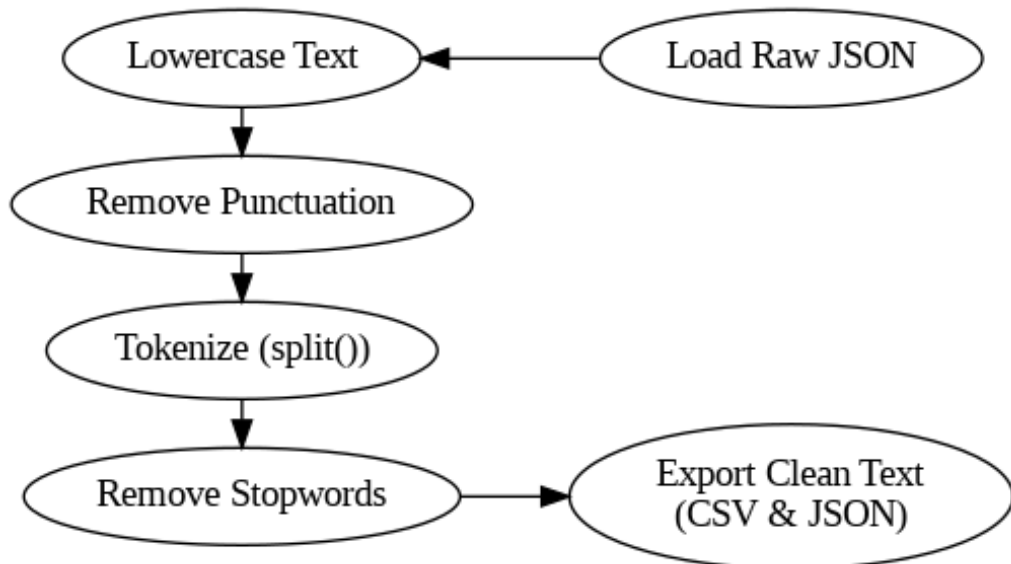


Figure 4.5. Preprocessing Pipeline

Preprocessing Steps

The following transformations were applied uniformly to all headlines:

- **Lowercasing:** All text was converted to lowercase to standardize word forms and reduce vocabulary sparsity.
- **Punctuation Removal:** All non-alphanumeric characters (including exclamation marks, commas, periods, etc.) were removed using regular expressions. While this may strip some potential sarcasm cues, it was deemed necessary for consistent API processing.
- **Tokenization:** Headlines were tokenized using Python's built-in `split()` method, which splits on whitespace. Although more advanced tokenizers (e.g., NLTK's `word_tokenize`) exist, a simpler approach was used due to the short length of headlines.
- **Stopword Removal:** Common English stopwords (from NLTK's corpus) were removed to focus on semantically meaningful words. This also helped reduce input length for APIs with character or token limits.

Example:

"Oh fantastic, another traffic jam!" → "fantastic another traffic jam"

Preserving Sarcasm Cues

While the preprocessing aimed to simplify inputs, minimal linguistic transformation was applied to preserve tone and word order. However, punctuation—such as exclamation marks or ellipses—was not retained in the current pipeline. Since sarcasm often relies on subtle cues like punctuation and function words (e.g., "oh," "really," "sure"), some nuance may be lost during stopword removal and basic tokenization. This trade-off was accepted to ensure compatibility and fairness across APIs, though future iterations may explore preprocessing variants that better retain sarcasm-related linguistic signals.

Format Output

The cleaned data was exported into two formats for use in downstream tasks and API testing:

- **CSV:** For manual inspection, analysis, and debugging.
- **JSON Lines:** A line-delimited JSON format required by most commercial NLP APIs.

Table 4.3 presents a sample of the cleaned sarcasm-labeled dataset used in the evaluation pipeline. This preprocessed dataset served as the primary input across all four NLP APIs, ensuring uniformity in performance benchmarking.

Index	Clean Text	Label
9138	aweism could soulful humanists answer religious transcendence	0
18840	dont spend cent bitcoin see john olivers cryptocurrency warning	0
11802	5 reasons need boundaries relationships life	0
17248	naacp bed donald sterling	0
5284	sat found biased favor nonhungover	1

Table 4.3. Sample headlines from sarcasm dataset used for evaluation

4.3 API Testing Setup

Each API was configured according to its unique requirements, including obtaining authentication credentials via platform dashboards and ensuring proper API setup. The primary goal was to provide uniform access and streamline testing processes across different platforms.

Authentication

The authentication process for each API was handled using specific credentials that were securely stored for each platform. The authentication requirements for each API are as follows:

- **Google Cloud:** API key + billing account
- **AWS:** IAM credentials and session token
- **IBM Watson:** API key and endpoint URL
- **Azure:** Subscription key and region endpoint

These authentication credentials were critical for securely making requests to the respective APIs and ensuring proper usage tracking and access control.

Sample Request Format

Below is an example of the JSON format used to structure the input for Azure Text Analytics API. Each request includes a list of documents with their respective language and text content for sentiment analysis. This format was used across all platforms, with slight variations in syntax depending on each API's requirements.

```
{
  "documents": [
    {
      "language": "en",
```

```

        "id": "123",
        "text": "Lovely weather for a hurricane!"
    }
]
}

```

4.4 API Testing Process

The API testing process was designed to ensure consistent interaction with each platform while capturing performance metrics like latency and request success rates. The process was structured to handle both individual and batched requests, depending on API-specific configurations.

Batching and Requests

The API requests were handled in batches of news headline samples to optimize throughput and reduce the number of API calls. The batch sizes for each platform, along with relevant constraints, are summarized in Table 4.4.

API Platform	Batch Size	Notes
Google Cloud Natural Language	100	Maximum batch size for optimal throughput
AWS Comprehend	25	Limited by API rate thresholds
IBM Watson NLU	1	Processed individually due to emotion scoring overhead
Azure Text Analytics	10	JSON array with a cap of 10 documents per request

Table 4.4. Batch sizes and request constraints for each NLP API

Each API call was executed via Python's `requests` library, ensuring that all requests followed a consistent structure. The scripts also incorporated custom logging to track metrics such as request timestamps, batch IDs, and latency.

Timing Example

Below is an example of how latency was tracked during API requests. The code snippet captures the start time, sends the request, and calculates the elapsed time for each API call:

```
start = time.time()
```

```
response = requests.post(api_url, headers=headers, json=batch)
latency = time.time() - start
```

This method provided insights into the time it took for each API to process batch requests in total, which was vital for performance evaluation.

API Testing Architecture

The testing architecture was standardized across all APIs to ensure consistency in how requests and responses were handled. This structure enabled seamless integration with different platforms and facilitated accurate comparisons.

4.5 Data Collection and Storage

Volume and Deduplication

Each API processed 15,000 unique samples. Google exceeded this number during testing but was filtered post-hoc. Each response was stored in:

- **Raw JSON:** Entire payload from API
- **Flat CSV:** Prediction, ID, latency
- **Merged Summary:** Includes ground truth and API score

4.6 Error Handling and Logging

Observed Issues

- **Google:** Serialization errors with float sentiment magnitude.
- **AWS:** “ThrottlingException” during batch spikes.
- **Azure:** Schema failures from malformed batch JSON.
- **IBM:** Timeout for long emotional responses.

Mitigations

- Retry logic with exponential backoff.
- JSON format validation using `jsonschema`.
- Filtering high-magnitude values to fix float serialization.
- Caching partial batch responses to resume failed runs.

All errors were tracked via a custom logging module. Logs were written to structured text files and included timestamps, error types, and affected batch IDs, enabling post-mortem debugging and recovery automation.

5. RESULT ANALYSIS

This chapter presents a comprehensive analysis of the experimental results obtained from the evaluation of the four cloud-based NLP APIs: Google Cloud Natural Language API, AWS Comprehend, IBM Watson Natural Language Understanding, and Microsoft Azure Text Analytics. The evaluation was performed on sarcasm-rich text data, with both quantitative and qualitative metrics assessed to evaluate each service's effectiveness in detecting sentiment and emotion under challenging linguistic conditions.

Quantitative metrics such as accuracy, latency, and cost provide measurable insights into API performance, while qualitative analysis, including error patterns and usability assessments, offers a more nuanced view of system behavior in real-world conditions.

5.1 Quantitative Metrics

Quantitative evaluation focused on three key performance dimensions: accuracy, latency, and cost. Each dimension reveals different aspects of system effectiveness and operational feasibility when dealing with sarcasm detection challenges.

5.1.1 Accuracy

Accuracy was calculated by comparing each API's predicted sentiment (after standardization) with the human-annotated ground truth labels. To account for differences in output formats:

- Continuous sentiment scores (from Google and IBM) were thresholded into binary classes: scores below 0 were treated as *Negative* (sarcastic), and scores above or equal to 0 were treated as *Non-Negative* (non-sarcastic).
- Discrete outputs (from AWS and Azure) were directly mapped: *Negative* predictions were mapped to 1 (sarcastic), and other categories were mapped to 0 (non-sarcastic).

The results are summarized in Table 5.1.

IBM Watson NLU achieved the highest accuracy (55.30%), while Google Cloud NLP recorded the lowest (52.57%). However, the absolute differences across all platforms

were relatively small, highlighting the challenge sarcasm presents even for state-of-the-art commercial NLP services.

API	Accuracy (%)
Google Cloud NLP	52.57%
AWS Comprehend	54.43%
IBM Watson NLU	55.30%
Azure Text Analytics	53.59%

Table 5.1. Accuracy results across APIs on sarcasm detection ($n = 14,857$ samples)

To support these observations, Figure 5.1 visually compares API accuracies:

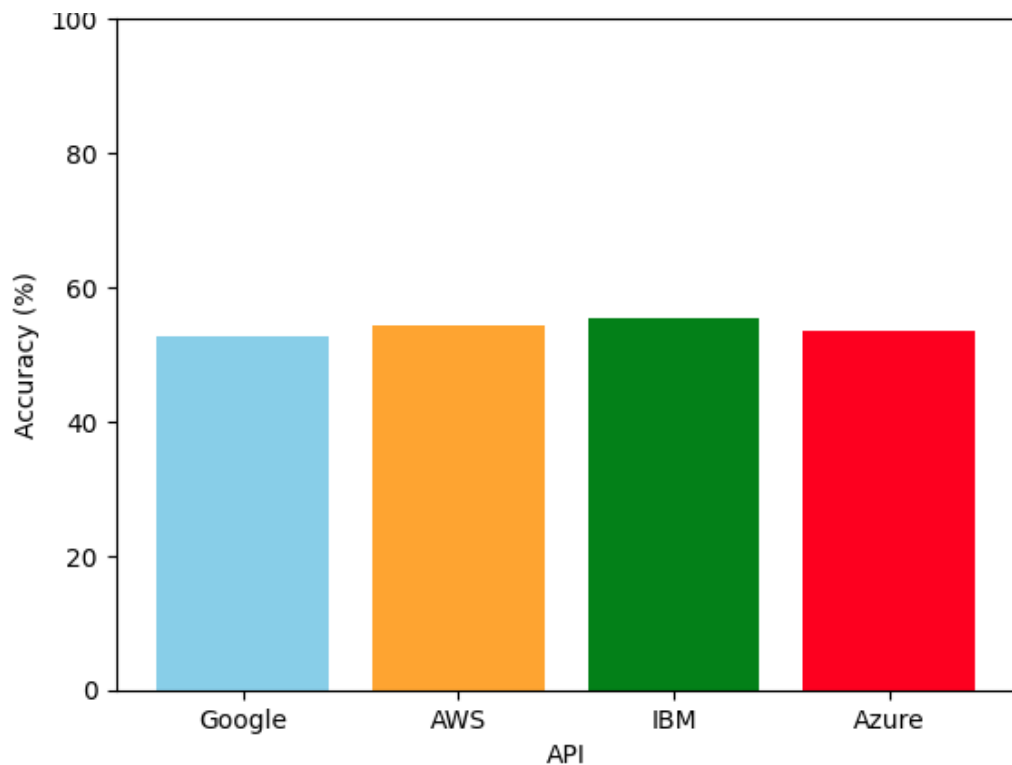


Figure 5.1. Accuracy comparison across NLP APIs

Confusion Matrices

To better understand prediction behavior, confusion matrices were generated for each API (Figures 5.2–5.5). These matrices provide a detailed look at the true positive, false positive, true negative, and false negative classifications for sarcastic and non-sarcastic instances. Each confusion matrix reveals how each API handles the challenge of distinguishing sarcasm from non-sarcasm.

In Google Cloud NLP's case (Figure 5.2), the predictions are relatively balanced between true positives and true negatives, but a notable number of sarcastic inputs are still misclassified, suggesting moderate difficulty in capturing nuanced sarcastic tone.

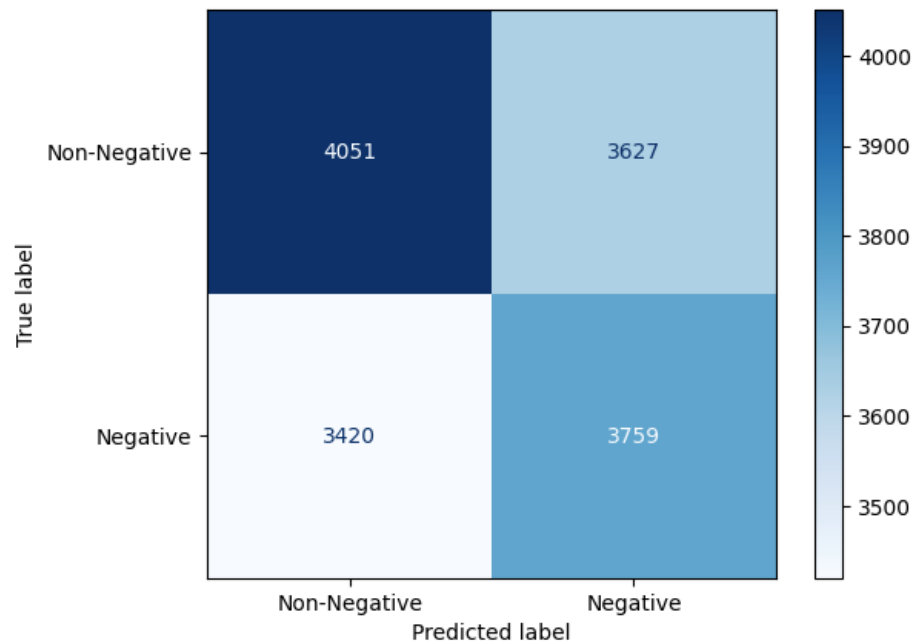


Figure 5.2. Google Cloud NLP: Distribution of sarcastic vs non-sarcastic headlines.

AWS Comprehend (Figure 5.3) shows a strong bias toward predicting non-sarcastic labels, reflected in a high true negative rate but a low recall for sarcasm. This is consistent with the classification report, where sarcastic recall is only 30%.

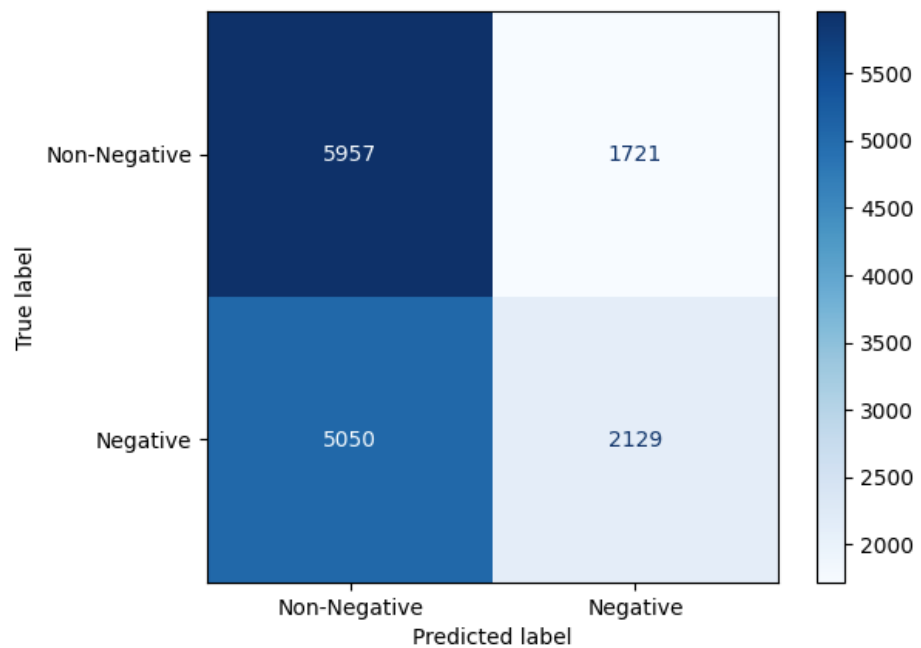


Figure 5.3. AWS Comprehend: Distribution of sarcastic vs non-sarcastic headlines.

IBM Watson NLU (Figure 5.4) performs somewhat better with sarcastic recall, resulting in a more populated true positive region in the matrix. However, misclassifications are still frequent.

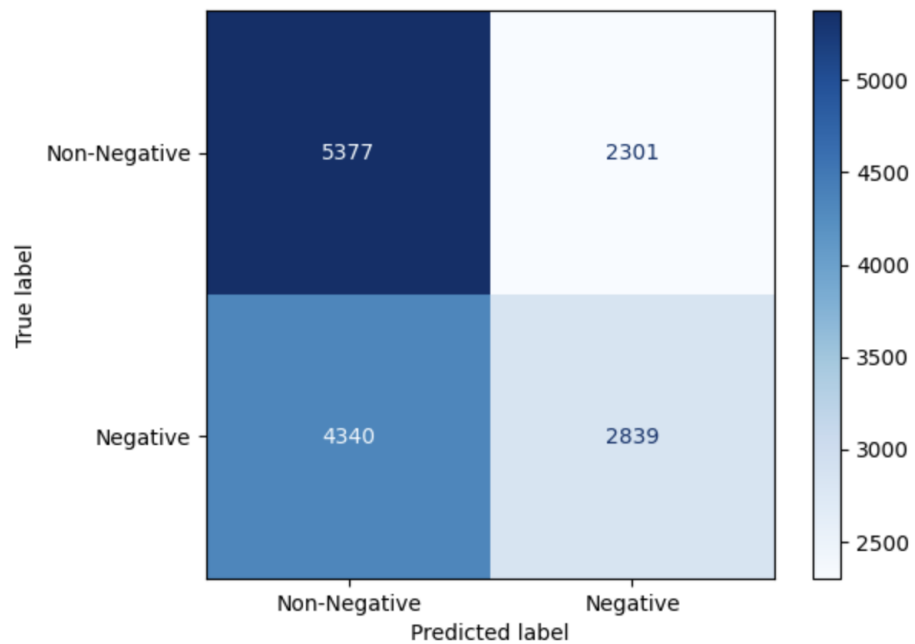


Figure 5.4. IBM Watson NLU: Distribution of sarcastic vs non-sarcastic headlines.

Azure Text Analytics (Figure 5.5) offers a similar profile, with slightly improved sarcastic detection compared to AWS.

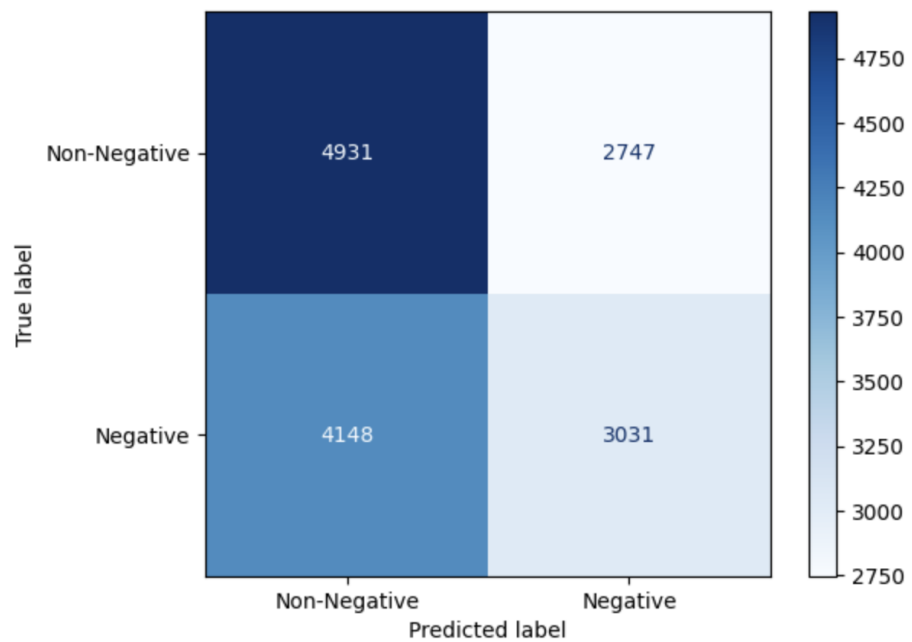


Figure 5.5. Azure Text Analytics: Distribution of sarcastic vs non-sarcastic headlines.

A general pattern across all matrices is the elevated number of false negatives—sarcastic

inputs labeled as non-sarcastic—highlighting the APIs’ collective struggle with detecting sarcasm. These matrices provide concrete visual evidence of the systemic bias toward non-sarcastic classifications in ambiguous or ironic text.

Classification Reports

The full classification reports for each API, which include precision, recall, and F1-score, are provided in Table 5.2. These reports quantify the performance in distinguishing between sarcastic and non-sarcastic cases.

While the precision and recall values for non-sarcastic classes are consistently higher, sarcastic recall remains relatively low across all APIs—with the exception of Google Cloud NLP, which maintains a more balanced performance between the two classes. AWS, in particular, exhibits the weakest recall for sarcasm (30%), leading to a lower F1-score. Azure achieves the best F1-score for sarcastic detection, but IBM and Google are close behind in terms of balance. These patterns reinforce the insights from the confusion matrices, where false negatives—misclassified sarcastic inputs—were the dominant error type.

API	Class	Precision	Recall	F1-score
Google Cloud NLP	0 (Non-sarcastic)	0.54	0.53	0.53
Google Cloud NLP	1 (Sarcastic)	0.51	0.52	0.52
AWS Comprehend	0 (Non-sarcastic)	0.54	0.78	0.64
AWS Comprehend	1 (Sarcastic)	0.55	0.30	0.39
IBM Watson NLU	0 (Non-sarcastic)	0.55	0.70	0.62
IBM Watson NLU	1 (Sarcastic)	0.55	0.40	0.46
Azure Text Analytics	0 (Non-sarcastic)	0.54	0.64	0.59
Azure Text Analytics	1 (Sarcastic)	0.52	0.42	0.47

Table 5.2. *Classification Reports for Sarcasm Detection*

5.1.2 Latency

The average latency results, shown in Table 5.3 reveal considerable variation in processing speed across the four NLP APIs. Azure Text Analytics emerged as the fastest service, completing the analysis of 15,000 samples in just 17 minutes. This performance indicates an efficient backend infrastructure optimized for high-throughput batch processing, making Azure particularly suitable for time-sensitive applications.

Google Cloud NLP followed with a latency of 50 minutes—significantly faster than

AWS Comprehend (70 minutes) and IBM Watson NLU (120 minutes). While Google's performance was moderate, it represents a good balance between speed and usability, especially for medium-scale workloads.

AWS Comprehend's latency was notably higher than Azure and Google, possibly due to overhead in processing and internal response buffering. However, the most time-consuming was IBM Watson NLU, which took two hours on average to complete the dataset. This extended latency can likely be attributed to its more granular emotional analysis pipeline and configurable output layers, which, while rich in detail, incur additional computational costs.

API	Average Latency (minutes for 15,000 samples)
Google Cloud NLP	50 minutes
AWS Comprehend	70 minutes
IBM Watson NLU	120 minutes
Azure Text Analytics	17 minutes

Table 5.3. Average Latency for Full Dataset Processing

Overall, latency differences highlight critical trade-offs between speed and processing depth. Azure offers the best option for low-latency, large-scale deployment, whereas IBM may be more appropriate for offline or research contexts where depth of analysis outweighs response time.

5.1.3 Cost

Estimated costs for processing 15,000 text samples were derived based on official pricing documentation available as of early 2025. Table 5.4 detail the estimated costs associated with each API usage over 15,000 samples.

API	Estimated Cost (€)
Google Cloud NLP	€9.30
AWS Comprehend	€1.40
IBM Watson NLU	€34.88
Azure Text Analytics	€18.60

Table 5.4. Estimated cost for processing 15,000 samples based on public pricing

Google Cloud Natural Language API pricing was structured based on the number of characters processed per request. Charges were applied incrementally, with decreasing

per-character costs at higher usage tiers. For the sample size considered, approximately €9.30 was estimated based on typical request lengths.

Amazon Comprehend employed a pricing model based on 100-character units, with a minimum charge of three units per request. This model resulted in an estimated cost of €1.40 for the dataset, making it the most cost-effective among the evaluated services.

IBM Watson NLU offered tiered plans based on the number of API calls. The estimated cost of €34.88 reflected the use of the standard embedded NLP offering at the base plan, which provided rich analytical features but incurred the highest cost per sample.

Azure Text Analytics followed a record-based pricing model, with one text document treated as one record. At €1.00 per 1,000 records, the estimated cost for 15,000 records was approximately €18.60. This positioned Azure between Google and IBM in terms of overall expense.

In summary, AWS Comprehend was found to be the most cost-efficient option under default settings, while IBM Watson NLU incurred the highest cost due to its pricing structure and advanced feature set. Google Cloud NLP and Azure Text Analytics were observed to offer moderate cost profiles, balancing capability and affordability for medium-scale deployments. Figure 5.6 visualizes these comparisons.

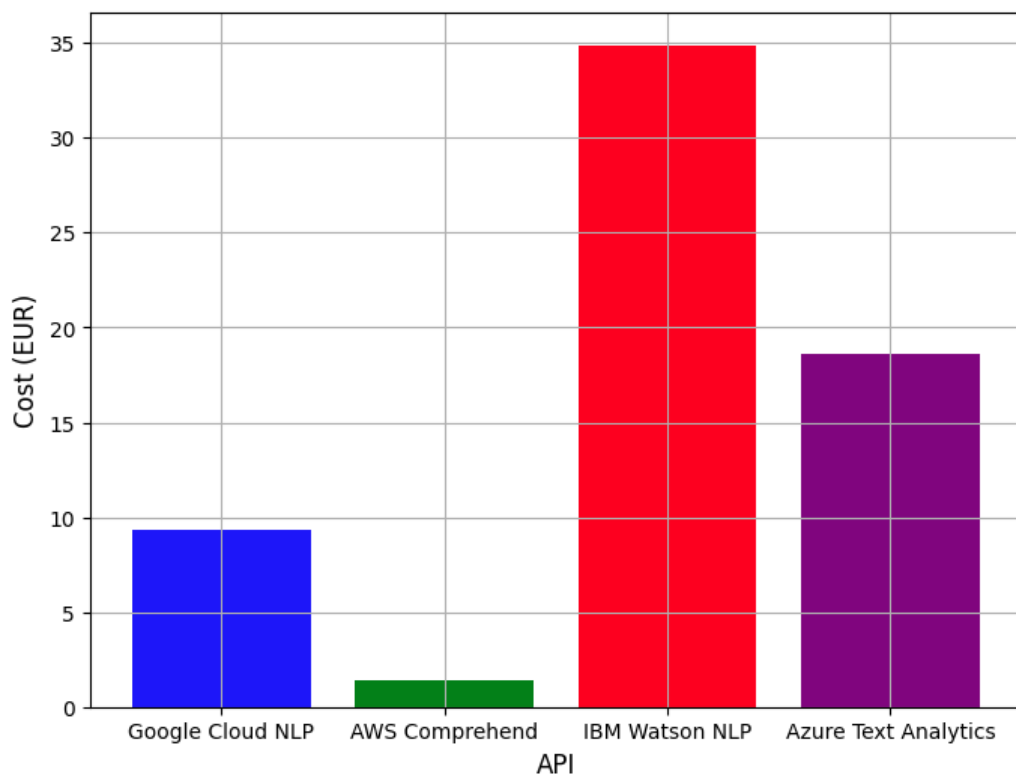


Figure 5.6. Estimated cost comparison across APIs

5.2 Qualitative Metrics

In addition to quantitative metrics, several qualitative error patterns emerged during analysis. Additionally, the usability metric was also assessed.

5.2.1 Error Patterns and Case Studies

Qualitative analysis was performed of the 14,857 test samples to uncover recurring failure modes across APIs. Misclassifications were identified by comparing each API's binary sentiment prediction against the ground truth sarcasm label. This comparison used custom scripts to isolate disagreement cases (e.g., when an API labeled a sarcastic input as non-sarcastic or vice versa). Further, ground truth mismatches were flagged in cases where *all four APIs* agreed on a prediction that conflicted with the actual label—indicating deeper, model-agnostic blind spots. Representative samples were manually reviewed to extract qualitative patterns, and error counts were compiled as shown in Table 5.5.

API	Misclassifications (n = 14,857)
Google Cloud NLP	7047
AWS Comprehend	6771
IBM Watson NLU	6641
Azure Text Analytics	6895

Table 5.5. Total misclassified samples by API

Figure 5.7 contrasts the ground truth sarcasm distribution with predictions derived via majority voting (threshold ≥ 3 APIs predicting sarcasm), highlighting a consistent underprediction of sarcasm due to ambiguity and context insensitivity.

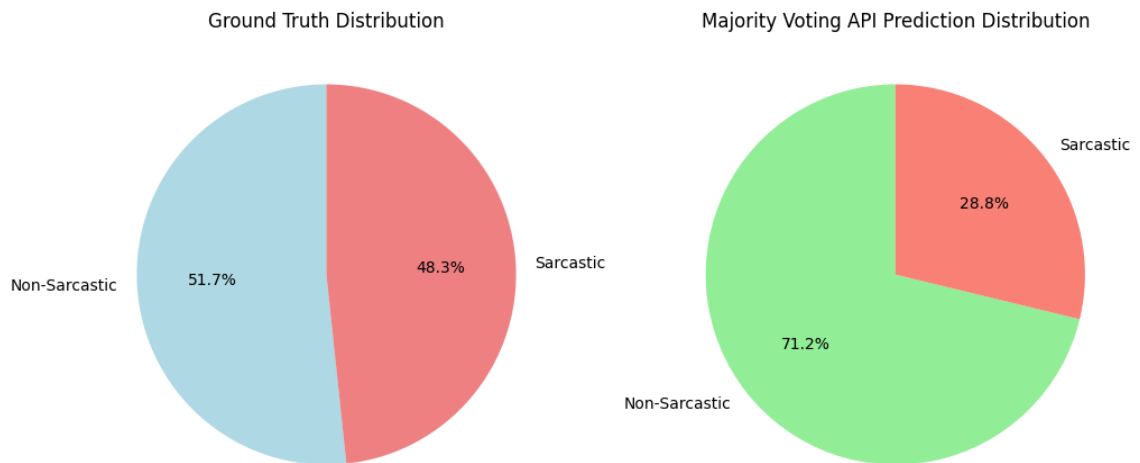


Figure 5.7. Left: Ground truth sarcasm vs non-sarcasm; Right: Majority voting prediction distribution

Common Misinterpretation Patterns

Five core failure modes were consistently observed across platforms, often contributing to misclassifications:

1. **Literal Interpretation of Sarcasm:** Sentences using overtly positive language were mistaken as genuine sentiment.

“Absolutely thrilled to get stuck in traffic again!” (Sarcastic → Non-sarcastic)

2. **Contextual Blindness:** All APIs operated on isolated inputs, failing to leverage discourse or pragmatic context.

“Dem rep totally nails congress falling short gender racial equality”
(Non-Sarcastic → Sarcastic)

3. **Negation Misfires:** Subtle negations or irony were missed, particularly in mixed-tone inputs.

“Inclement weather prevents liar getting work” (Sarcastic → Non-sarcastic)

4. **Entity Misdirection:** Emotionally charged or politically salient named entities biased sentiment inference.

“Chris Christie suggests Hillary Clinton blame Boko Haram...” (Non-Sarcastic → Sarcastic)

5. **Universal Failures:** Some samples were misclassified by all four systems, reflecting shared model blind spots tied to cultural or satirical knowledge gaps.

Text (truncated)	Label
Shadow government getting large meet Marriott conference room B	Sarcastic
Ford develops new SUV runs purely gasoline	Sarcastic
Area boy enters jumpingandtouchingtopsofdoorways phase	Sarcastic
Area man traveling gurney	Sarcastic
Secret Service agent secret David Alan Grier fan	Sarcastic

Table 5.6. Examples misclassified by all APIs

Edge Cases with Over-Prediction

Some non-sarcastic examples were falsely flagged as sarcastic across all APIs, indicating a tendency to overinterpret satire-like cues:

“Cat scared shelter won’t even look” (Non-Sarcastic → Sarcastic by all APIs)

These overpredictions highlight a shared limitation in distinguishing subtle satire from sarcasm, particularly in emotionally neutral or ambiguous headlines.

API-Specific Behaviors

While patterns overlapped, each API exhibited unique tendencies:

- **Google Cloud NLP:** Frequently misclassified emotionally loaded sarcasm due to surface-level sentiment cues.
- **AWS Comprehend:** Tended to overuse the “Neutral” label when faced with ambiguity, impairing sarcastic recall.
- **IBM Watson NLU:** Provided nuanced sentiment analysis but suffered from inconsistencies in edge cases and slower responses.
- **Azure Text Analytics:** Struggled with rigid input formatting; malformed requests led to silent failures, reducing robustness.

5.2.2 Usability

Beyond raw performance, usability and developer experience strongly influence the practical adoption of NLP APIs. Table 5.7 summarizes the comparative evaluation across platforms.

API	Ease of Setup	Documentation Quality	Operational Stability
Google Cloud NLP	High	Excellent	Stable
AWS Comprehend	High	Moderate	Stable
IBM Watson NLU	Very High	Excellent	Stable
Azure Text Analytics	Moderate	Good	Schema-sensitive

Table 5.7. Qualitative Evaluation of API Usability

- **Google Cloud NLP** offered a seamless integration experience with high-quality documentation and consistent API responses. Its strong SDK support across multiple languages made it ideal for rapid development. For general sarcasm detection, it serves as a reliable plug-and-play option.
- **AWS Comprehend** was similarly easy to set up and stable in operation, though its documentation was less detailed on advanced topics such as confidence scores and handling ambiguous sentiment. It tended to default to “Neutral” in unclear cases, which reduced informativeness in nuanced contexts. Nevertheless, it remains suitable for general-purpose use with minimal configuration.
- **IBM Watson NLU** provided the richest and most configurable feature set, supporting fine-grained sentiment and emotion analysis. This flexibility benefits advanced users and long-term maintainability but introduces a steeper learning curve and longer response times. Its modular design is best suited for sophisticated use cases.
- **Azure Text Analytics** was fast and functional but less forgiving in practice. Input schema rigidity led to silent failures when payloads were improperly formatted, complicating debugging. While its documentation was generally clear, the lack of explicit error messaging made development more error-prone compared to other services.

In summary, even with high-level language models, sarcasm detection remains elusive due to the absence of pragmatics, tone, and world knowledge. These misclassifications underline the need for context-aware architectures and multimodal training data.

6. DISCUSSION

This chapter synthesizes the results presented in the preceding chapter and evaluates them in the context of the research questions. The discussion is structured around three main areas: key findings from the experimental evaluation, challenges that underline the difficulty of sarcasm detection in NLP, and broader implications for deploying these systems in real-world environments such as customer support, social media monitoring, and mental health assessment.

6.1 Key Findings

The results of this thesis demonstrate that while commercial NLP APIs offer robust baseline sentiment detection capabilities, their performance on sarcasm-heavy texts remains limited. Several critical observations were made during the evaluation:

- **IBM Watson NLU** achieved the highest overall accuracy (55.30%) and the best F1-score for detecting sarcasm (0.55). Its support for multi-emotion classification (e.g., joy, anger, sadness) provided richer interpretability, which may have aided its relative success on ironic expressions. Watson was also the only system to provide multi-label outputs, helping retain emotional complexity in some sarcastic cases.
- **AWS Comprehend** was the most cost-effective option (€1.40 per 15,000 samples) and offered fast average latency. However, it frequently defaulted to “Neutral” in ambiguous or sarcastic inputs, likely due to its conservative scoring design. This strategy reduces false positives but comes at the expense of true positive sarcasm detection.
- **Azure Text Analytics** achieved the fastest total processing time (17 minutes), making it a strong candidate for time-sensitive tasks. However, its rigid input schema and less nuanced handling of sarcasm limited its effectiveness. Azure’s scoring tended to cluster around the “Neutral” midpoint unless sentiment was overt.
- **Google Cloud NLP** had the lowest sarcasm detection accuracy (52.57%) and occasionally failed due to floating-point serialization errors in high-magnitude outputs. Its reliance on scalar sentiment values offered flexibility but introduced errors in extreme cases.

- **General Performance:** All APIs showed a marked decline in accuracy on sarcasm-rich data compared to typical sentiment corpora. Confusion matrices revealed consistently high false-negative rates, indicating a systematic difficulty in recognizing sarcastic negativity cloaked in positive language. Notably, sarcastic samples containing exaggeration or hyperbole were more likely to be misclassified than those involving understatement or deadpan irony.
- **Class Imbalance Handling:** APIs performed better on non-sarcastic samples, which reflects a likely imbalance in training datasets. Commercial sentiment models are often tuned on review or survey data, which contain relatively little sarcastic language, leading to a lack of generalization on complex expressions.

These findings highlight that the optimal tool depends on specific use-case priorities, such as real-time performance, cost, or interpretive nuance.

6.2 Challenges in Sarcasm Detection

Despite recent advances in deep learning and large-scale NLP models, sarcasm remains one of the most persistent and complex challenges in computational linguistics. This study reinforces four major obstacles that limit current API performance:

Literal Bias

All four APIs exhibited a strong tendency to infer sentiment based solely on lexical polarity. Positive words such as *“fantastic,” “great,”* or *“just what I needed”* were frequently misclassified as positive sentiment, even in obviously sarcastic contexts.

Example: *“Oh fantastic, another traffic jam.”* — This sentence was consistently interpreted as genuinely positive by Google, AWS, and Azure.

This literal bias is particularly problematic in environments where sarcasm and satire are common, such as social media, political commentary, and youth-oriented domains like memes or gaming forums.

Contextual Blindness

Sarcasm is often highly contextual. Without access to conversational history, speaker identity, or real-world background knowledge, even humans can struggle to detect sarcasm. Cloud APIs process each input sentence in isolation, treating it as a self-contained unit.

In longer documents or dialogue sequences, sarcasm may only become apparent after multiple sentences. For example, a sarcastic review may begin earnestly and then shift

tone—something difficult to catch without paragraph- or document-level context modeling.

Rigid Sentiment Labels

Most APIs operate on rigid categorical schemes: *Positive*, *Neutral*, and *Negative*. This structure is poorly suited to handle sarcasm, which often blends contradictory emotional signals.

Only IBM Watson partially mitigates this limitation by assigning probabilistic scores to multiple emotion categories simultaneously. However, even this fails to resolve sarcasm when underlying emotions (e.g., anger, humor, and resignation) are presented simultaneously or metaphorically.

Lexical Over-reliance

In subtle or syntactically complex cases, sarcastic intent is conveyed not through polarity words, but through phrasing, punctuation, or tone. For instance, sentences like *“Exactly what I needed today”* or *“That went well... not!”* require understanding of irony and syntactic cues.

Such expressions are especially common in irony-heavy domains like British journalism, internet humor, or satirical editorials. In these contexts, APIs relying on keyword detection or shallow syntax models often perform poorly.

6.3 Implications for NLP in Real-world Applications

The consequences of misinterpreting sarcasm vary significantly depending on the domain, but can often lead to serious misunderstandings, poor user experience, or even safety risks. The following sectors illustrate the practical importance of sarcasm-aware sentiment detection:

Customer Service and Brand Monitoring

Sarcastic complaints are common in product reviews and social media mentions. If such inputs are misclassified as positive or neutral, companies may misjudge customer satisfaction, potentially ignoring key pain points.

Example: *“Best hotel stay ever — loved the moldy bathroom!”*

Recommendation: Supplement general-purpose APIs with fine-tuned sarcasm classifiers or rule-based sarcasm pre-filters trained on domain-specific datasets.

Social Media and Trend Analysis

Public sentiment tracking for political campaigns, brand launches, or crisis response may be skewed if sarcastic commentary is interpreted literally. A trending hashtag may appear supportive when it's actually critical in tone (e.g., #ThanksForNothing). Misinterpreting such trends could lead to public relations blunders.

Recommendation: Use sarcasm-aware models for platforms like Twitter or Reddit, where irony is common, and validate sentiment pipelines using sarcastic sample benchmarks.

Mental Health and Behavioral Monitoring

In sensitive domains like mental health, sarcastic remarks may mask distress or suicidal ideation. For example, statements like *"Just another day wishing I didn't exist — yay me!"* may appear neutral or positive to a literal parser.

Recommendation: In high-stakes applications, use transformer-based or human-in-the-loop systems that can flag sarcasm as high-risk or ambiguous.

Compliance and Legal Risk

Sarcastic threats or jokes in email or internal communications could escape detection by compliance software if sentiment analysis fails to recognize ironic intent. In regulatory domains such as finance, law, or healthcare, even one missed sarcastic warning could lead to compliance violations.

Recommendation: Flag ambiguous statements for manual review and audit high-risk domains with sarcasm-trained filters.

Operational Best Practices

Based on the findings, the following general practices are recommended:

- **Use Hybrid Systems:** Combine API sentiment output with sarcasm classifiers, either rule-based or ML-based.
- **Validate on Real Data:** Always test APIs using in-domain data that includes sarcasm and ambiguous cases.
- **Favor Flexible Models:** Select APIs offering emotion distributions or confidence scores over hard categorical labels.
- **Maintain Human Oversight:** In critical applications, ensure human review of flagged or sentiment-ambiguous cases.

6.4 Recommendations for API Providers

In addition to consumer recommendations, there are important improvements API developers should consider in future iterations:

- **Integrate Contextual Transformers:** Models like BERT or GPT should be embedded to better handle multi-sentence context and discourse markers that signal irony.
- **Offer Sarcasm Flags:** APIs should include optional sarcasm probability scores or irony detectors to supplement standard sentiment analysis.
- **Enable Adaptive Thresholding:** Developers should expose control over classification thresholds and confidence cutoffs to allow tuning for sarcasm-heavy environments.
- **Document Known Limitations:** Providers should clearly communicate known issues in sarcasm detection and recommend fallback procedures or workarounds.

Such upgrades would not only enhance API utility but also promote transparency and responsible AI deployment.

These observations collectively respond to the central research questions posed at the outset of this study. First, the accuracy of sentiment detection in sarcasm-rich contexts was found to be limited across all APIs, with IBM Watson performing slightly better than others. Second, a comparative analysis revealed clear trade-offs between latency, cost, accuracy, and interpretability. Finally, the investigation surfaced recurring challenges—such as literal bias, context blindness, and rigid sentiment schemas—that underpin the difficulties in sarcasm detection across all platforms.

7. CONCLUSION AND FUTURE WORK

This final chapter summarizes the key findings from the experimental evaluation of commercial sentiment analysis APIs in the presence of sarcasm. It reflects on the implications of the observed performance gaps, outlines the limitations of the current study, and proposes directions for future research. By synthesizing what has been learned, the chapter aims to contextualize the results within the broader goals of sentiment analysis and natural language understanding.

7.1 Findings

This thesis undertook a critical evaluation of four leading commercial sentiment analysis APIs—IBM Watson NLU, AWS Comprehend, Azure Text Analytics, and Google Cloud NLP—using a balanced, sarcasm-rich benchmark dataset. The study was motivated by practical use cases in which accurate sentiment interpretation—especially of sarcasm—is crucial, such as customer feedback analysis, social media monitoring, and digital mental health tools.

The findings confirmed that while these APIs offer robust baseline sentiment capabilities, their effectiveness deteriorates significantly when applied to sarcastic inputs. The highest sarcasm detection accuracy observed was just 55.30%, underscoring the challenge sarcasm presents for systems trained primarily on literal, polarity-based sentiment data. Across all APIs, misclassifications were common in ironic or deadpan statements, and confusion matrices revealed a consistent skew toward false negatives.

Rather than identifying a single best-performing API, the results point to a broader systemic issue: sarcasm detection remains a poorly addressed capability in mainstream NLP tools. Despite differences in architecture, scoring granularity, and response format, all four systems struggled to handle context, contradiction, and pragmatic intent—key components of sarcastic expression.

These insights not only address the core research questions, but also reinforce the need for hybrid architectures, transformer-based models, and context-enriched pipelines to advance sentiment analysis beyond surface-level affect.

7.2 Limitations

Several limitations must be acknowledged when interpreting the results of this research:

- **Opaque model architectures:** The proprietary nature of commercial APIs precludes transparency regarding the underlying model architectures and training datasets, limiting interpretability.
- **Restricted input scope:** The evaluation was limited to sentence-level inputs, as dictated by API constraints. This limitation is noteworthy given that sarcasm often emerges from broader discourse or conversational context.
- **Monolingual analysis:** All experiments were conducted in English, which restricts the generalizability of the results to other languages and cultural contexts where sarcasm may manifest differently.
- **Dataset constraints:** While the dataset used was balanced and contextually diverse, sarcasm is inherently subjective. Incorporating additional sources, such as conversational threads, audio transcripts, or social media interactions, could yield richer insights.

These limitations serve not only as caveats to the current study but also as opportunities for refinement in subsequent research.

7.3 Future Work

Advancing the field of sarcasm detection—particularly in the context of sentiment analysis—necessitates focused and interdisciplinary efforts. Building on the findings of this thesis, several promising directions for future work are outlined below:

Transformer-based Sarcasm Classifiers

Large language models such as BERT, RoBERTa, and GPT-4 have demonstrated strong performance on sarcasm-related benchmarks when fine-tuned on specialized corpora. Future research should explore the integration of these models as dedicated sarcasm classifiers, either standalone or in tandem with commercial APIs. These architectures are capable of leveraging richer contextual cues, such as speaker history or discourse structure, which are often critical for sarcasm interpretation.

Context-enriched Sentiment Systems

To overcome the limitations of sentence-level analysis, future sentiment systems should support document-level, conversational, and even multimodal inputs. Sarcasm often arises through contrast, irony, or timing—factors that elude models restricted to decon-

textualized text. Expanding APIs to accept contextual metadata, such as dialogue history or user intent, could significantly improve sarcasm recognition in dynamic environments.

Explainable Sarcasm Detection

Interpretability remains an underexplored but vital aspect of sentiment analysis. Future models should be designed to provide transparent justifications for their predictions, especially in cases of sarcasm. Highlighting linguistic features such as contrastive cues, hyperbole, punctuation patterns, or syntactic irregularities can improve both user trust and system accountability.

Cross-cultural and Multilingual Sarcasm

Sarcasm is a culturally embedded phenomenon, with significant variation in how it is expressed across different languages and societies. British irony, American deadpan humor, and South Asian sarcasm each pose unique challenges. Future work should examine how commercial and open-source models handle these cultural differences, and evaluate performance on multilingual sarcasm datasets, including those for low-resource languages.

Human-in-the-Loop and Real-Time Systems

In high-stakes domains such as mental health screening or customer grievance management, the misclassification of sarcasm can lead to serious misinterpretations. Future systems should incorporate human-in-the-loop mechanisms, such as active learning frameworks or fallback escalation protocols, to mitigate error in ambiguous cases. Real-time sarcasm detection pipelines that combine automated classifiers with human oversight could represent a practical middle ground in sensitive applications.

Sarcasm, by its nature, encapsulates some of the most human dimensions of language: subtlety, contradiction, shared context, and humor. As sentiment analysis becomes increasingly embedded in decision-making systems, the ability to correctly interpret ironic or sarcastic statements will become more than a linguistic curiosity; it will be a functional necessity. In addressing the research questions, this thesis demonstrated that commercial NLP APIs struggle to accurately detect sarcasm, often defaulting to literal interpretations. A detailed comparison revealed performance trade-offs across accuracy,

latency, cost, and usability, affirming that no single API excels in all areas. Common failure patterns—such as lexical over-reliance and a lack of contextual modeling—further highlight the shared limitations across these platforms. These insights directly align with the study's original inquiry and reinforce the need for sarcasm-aware improvements in both academic and commercial sentiment systems.

REFERENCES

- [1] Salvatore Attardo. *A primer for the linguistics of humor*. Vol. 2008. na, 2008.
- [2] AWS Documentation. *Amazon Comprehend Developer Guide*. 2023. URL: <https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html>.
- [3] David Bamman and Noah Smith. “Contextualized sarcasm detection on twitter”. In: *proceedings of the international AAAI conference on web and social media*. Vol. 9. 1. 2015, pp. 574–577.
- [4] Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. “Parsing-based sarcasm sentiment recognition in twitter data”. In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. 2015, pp. 1373–1380.
- [5] Christian Burgers, Margot Van Mulken, and Peter Jan Schellens. “Verbal irony: Differences in usage across written genres”. In: *Journal of Language and Social Psychology* 31.3 (2012), pp. 290–310.
- [6] Erik Cambria et al. “Sentiment analysis is a big suitcase”. In: *IEEE Intelligent Systems* 32.6 (2017), pp. 74–80.
- [7] Cambridge Dictionary. *Sarcasm*. Accessed: 2025-05-13. 2025. URL: <https://dictionary.cambridge.org/dictionary/english/sarcasm?q=Sarcasm>.
- [8] John D Campbell and Albert N Katz. “Are there necessary conditions for inducing a sense of sarcastic irony?” In: *Discourse Processes* 49.6 (2012), pp. 459–480.
- [9] Ethem F Can, Aysu Ezen-Can, and Fazli Can. “Multilingual sentiment analysis: An RNN-based framework for limited data”. In: *arXiv preprint arXiv:1806.04511* (2018).
- [10] Santiago Castro et al. “Towards multimodal sarcasm detection (an _obviously_ perfect paper)”. In: *arXiv preprint arXiv:1906.01815* (2019).
- [11] Dushyant Singh Chauhan et al. “Sentiment and emotion help sarcasm? A multi-task learning framework for multi-modal sarcasm, sentiment and emotion analysis”. In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, pp. 4351–4360.
- [12] Yahui Chen. “Convolutional neural network for sentence classification”. MA thesis. University of Waterloo, 2015.
- [13] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).
- [14] Alexis Conneau et al. “Unsupervised cross-lingual representation learning at scale”. In: *arXiv preprint arXiv:1911.02116* (2019).

- [15] Dmitry Davidov, Oren Tsur, and Ari Rappoport. "Semi-supervised recognition of sarcasm in Twitter and Amazon". In: *Proceedings of the fourteenth conference on computational natural language learning*. 2010, pp. 107–116.
- [16] Dorottya Demszky et al. "GoEmotions: A dataset of fine-grained emotions". In: *arXiv preprint arXiv:2005.00547* (2020).
- [17] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.
- [18] Paul Ekman. "Are there basic emotions?" In: (1992).
- [19] Tatiana Ermakova et al. "A comparison of commercial sentiment analysis services". In: *SN Computer Science* 4.5 (2023), p. 477.
- [20] Andrea Esuli and Fabrizio Sebastiani. "SentiWordNet: a high-coverage lexical resource for opinion mining". In: *Evaluation* 17.1 (2007), p. 26.
- [21] Ibrahim Abu Farha and Walid Magdy. "From arabic sentiment analysis to sarcasm detection: The arsarcasm dataset". In: *The 4th Workshop on Open-Source Arabic Corpora and Processing Tools*. European Language Resources Association (ELRA). 2020, pp. 32–39.
- [22] Ibrahim Abu Farha et al. "SemEval-2022 task 6: iSarcasmEval, intended sarcasm detection in English and Arabic". In: *The 16th International Workshop on Semantic Evaluation 2022*. Association for Computational Linguistics. 2022, pp. 802–814.
- [23] Alexander Genkin, David D Lewis, and David Madigan. "Large-scale Bayesian logistic regression for text categorization". In: *technometrics* 49.3 (2007), pp. 291–304.
- [24] Richard J Gerrig and Yevgeniya Goldvarg. "Additive effects in the perception of sarcasm: Situational disparity and echoic mention". In: *Metaphor and Symbol* 15.4 (2000), pp. 197–208.
- [25] Aniruddha Ghosh and Tony Veale. "Fracking sarcasm using neural network". In: *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*. 2016, pp. 161–169.
- [26] Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. "Identifying sarcasm in twitter: a closer look". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 2011, pp. 581–586.
- [27] Google Cloud. *Moderating Text with the Natural Language API*. Accessed May 12, 2025. 2023. URL: <https://cloud.google.com/natural-language/docs/moderating-text>.
- [28] Google Cloud. *Cloud Natural Language API Documentation*. 2024. URL: <https://cloud.google.com/natural-language/docs>.

- [29] Hunter Gregory et al. "A transformer approach to contextual sarcasm detection in twitter". In: *Proceedings of the second workshop on figurative language processing*. 2020, pp. 270–275.
- [30] Jiatao Gu et al. "Learning to translate in real-time with neural machine translation". In: *arXiv preprint arXiv:1610.00388* (2016).
- [31] Devamanyu Hazarika et al. "Cascade: Contextual sarcasm detection in online discussion forums". In: *arXiv preprint arXiv:1805.06413* (2018).
- [32] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [33] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.
- [34] Clayton Hutto and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 8. 1. 2014, pp. 216–225.
- [35] IBM Cloud. *IBM Watson Natural Language Understanding Documentation*. 2023. URL: <https://www.ibm.com/cloud/watson-natural-language-understanding>.
- [36] Sergiu C Ivan, Robert Ş Györödi, and Cornelia A Györödi. "Sentiment Analysis Using Amazon Web Services and Microsoft Azure". In: *Big Data and Cognitive Computing* 8.12 (2024), p. 166.
- [37] Thorsten Joachims. "Text categorization with support vector machines: Learning with many relevant features". In: *European conference on machine learning*. Springer. 1998, pp. 137–142.
- [38] Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. "Automatic sarcasm detection: A survey". In: *ACM Computing Surveys (CSUR)* 50.5 (2017), pp. 1–22.
- [39] Aditya Joshi et al. "Sarcasm target identification: Dataset and an introductory approach". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.
- [40] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).
- [41] Faria Binte Kader et al. "'When Words Fail, Emojis Prevail': A Novel Architecture for Generating Sarcastic Sentences With Emoji Using Valence Reversal and Semantic Incongruity". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*. 2023, pp. 334–351.
- [42] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. "A large self-annotated corpus for sarcasm". In: *arXiv preprint arXiv:1704.05579* (2017).
- [43] Aziliz Le Glaz et al. "Machine learning and natural language processing in mental health: systematic review". In: *Journal of medical Internet research* 23.5 (2021), e15708.

- [44] Jiangnan Li et al. "Sarcasm detection with commonsense knowledge". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3192–3201.
- [45] Bing Liu. *Sentiment analysis and opinion mining*. Springer Nature, 2022.
- [46] Hugo Liu and Push Singh. "ConceptNet—a practical commonsense reasoning toolkit". In: *BT technology journal* 22.4 (2004), pp. 211–226.
- [47] Hui Liu, Wenya Wang, and Haoliang Li. "Towards multi-modal sarcasm detection via hierarchical congruity modeling with knowledge enhancement". In: *arXiv preprint arXiv:2210.03501* (2022).
- [48] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).
- [49] George Manias et al. "Multilingual text categorization and sentiment analysis: a comparative analysis of the utilization of multilingual approaches for classifying twitter data". In: *Neural Computing and Applications* 35.29 (2023), pp. 21415–21431.
- [50] Diana G Maynard and Mark A Greenwood. "Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis". In: *Lrec 2014 proceedings*. ELRA. 2014.
- [51] Andrew McCallum, Kamal Nigam, et al. "A comparison of event models for naive bayes text classification". In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Madison, WI. 1998, pp. 41–48.
- [52] Microsoft Azure. *Azure Text Analytics Documentation*. 2023. URL: <https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/overview>.
- [53] Microsoft Corporation. *Transparency Note for Sentiment Analysis*. Accessed May 12, 2025. 2023. URL: <https://learn.microsoft.com/en-us/legal/cognitive-services/language-service/transparency-note-sentiment-analysis>.
- [54] Abhijit Mishra, Tarun Tater, and Karthik Sankaranarayanan. "A modular architecture for unsupervised sarcasm generation". In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2019, pp. 6144–6154.
- [55] Prakamya Mishra, Saroj Kaushik, and Kuntal Dey. "Bi-ISCA: Bidirectional inter-sentence contextual attention mechanism for detecting sarcasm in user generated noisy short text". In: *arXiv preprint arXiv:2011.11465* (2020).
- [56] Rishabh Misra and Prahal Arora. "Sarcasm detection using hybrid neural network". In: *arXiv preprint arXiv:1908.07414* (2019).
- [57] Rishabh Misra and Prahal Arora. "Sarcasm Detection using News Headlines Dataset". In: *AI Open* 4 (2023), pp. 13–18. ISSN: 2666-6510. DOI: <https://doi.org/10.1016/j.aiopen.2023.01.001>. URL: <https://www.sciencedirect.com/science/article/pii/S2666651023000013>.

- [58] Saif M Mohammad and Peter D Turney. "Nrc emotion lexicon". In: *National Research Council, Canada 2* (2013), p. 234.
- [59] Pansy Nandwani and Rupali Verma. "A review on sentiment analysis and emotion detection from text". In: *Social network analysis and mining* 11.1 (2021), p. 81.
- [60] Peter Adebowale Olujimi and Abejide Ade-Ibijola. "NLP techniques for automating responses to customer queries: a systematic review". In: *Discover Artificial Intelligence* 3.1 (2023), p. 20.
- [61] Sebastião Pais, João Cordeiro, and M Luqman Jamil. "NLP-based platform as a service: a brief review". In: *Journal of Big Data* 9.1 (2022), p. 54.
- [62] Bo Pang, Lillian Lee, et al. "Opinion mining and sentiment analysis". In: *Foundations and Trends® in information retrieval* 2.1–2 (2008), pp. 1–135.
- [63] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques". In: *arXiv preprint cs/0205070* (2002).
- [64] Yolande Piris and Anne-Cécile Gay. "Customer satisfaction and natural language processing". In: *Journal of Business Research* 124 (2021), pp. 264–271.
- [65] Robert Plutchik. "A general psychoevolutionary theory of emotion". In: *Theories of emotion*. Elsevier, 1980, pp. 3–33.
- [66] Soujanya Poria et al. "A deeper look into sarcastic tweets using deep convolutional neural networks". In: *arXiv preprint arXiv:1610.08815* (2016).
- [67] Soujanya Poria et al. "Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research". In: *IEEE transactions on affective computing* 14.1 (2020), pp. 108–132.
- [68] Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. "A transformer-based approach to irony and sarcasm detection". In: *Neural Computing and Applications* 32.23 (2020), pp. 17309–17320.
- [69] Shraman Pramanick, Aniket Roy, and Vishal M Patel. "Multimodal learning using optimal transport for sarcasm and humor detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 3930–3940.
- [70] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. "Sarcasm detection on twitter: A behavioral modeling approach". In: *Proceedings of the eighth ACM international conference on web search and data mining*. 2015, pp. 97–106.
- [71] Ellen Riloff et al. "Sarcasm as contrast between a positive sentiment and negative situation". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 704–714.
- [72] Maarten Sap et al. "Atomic: An atlas of machine commonsense for if-then reasoning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3027–3035.

- [73] Samer Muthana Sarsam et al. "Sarcasm detection using machine learning algorithms in Twitter: A systematic review". In: *International Journal of Market Research* 62.5 (2020), pp. 578–598.
- [74] Rossano Schifanella et al. "Detecting sarcasm in multimodal social platforms". In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 1136–1145.
- [75] Martin Sykora, Suzanne Elayan, and Thomas W Jackson. "A qualitative analysis of sarcasm, irony and related# hashtags on Twitter". In: *Big Data & Society* 7.2 (2020), p. 2053951720972735.
- [76] Yi Tay et al. "Reasoning with sarcasm by reading in-between". In: *arXiv preprint arXiv:1805.02856* (2018).
- [77] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).
- [78] Deirdre Wilson. "The pragmatics of verbal irony: Echo or pretence?" In: *Lingua* 116.10 (2006), pp. 1722–1743.
- [79] Lei Zhang, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey". In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 8.4 (2018), e1253.