INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

in collaboration with
The University of Westminster (UK)

6COSC012C Final Year Project 2021/22

Thesis

Sarcastically

Sarcasm detection using a Multi-Modal Approach
via Text and Audio based Sentiment Analysis

Written By:
Ryan Joshua Kuruppu
2018351

Supervised by:
Mr. Banu Athuraliya

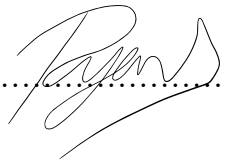
Submitted in fulfillment of the requirements for the BSc (Hons) Computer Science Course, Department of Computing

October 2021

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where it states otherwise by reference or acknowledgment, the work presented is entirely my own. The experimental work itself is almost entirely my own work and due references have been provided on all supporting literature and tooling.

Full Name: Ryan Joshua Kuruppu
IIT Registration No.: 2018351
University of Westminster ID No.: w1742302

Signature:  Date: 19/05/2022

Abstract

Sarcasm is one of the most well-known but easily misunderstood parts of the English language. While sarcasm is understood well to a vast majority, the uniqueness of sarcasm is that no one thinks about it the same way, when it is understood by the majority, it does its job but when it isn't is when questions start to arise in terms of meaning, understanding and so on. Most research on sarcasm has come in the form of a textual understanding of it using context from previous statements or blocks of statements and more recent research has led to the discovery of supplemental forms of associative data such as video and images which help to understand a machine's understanding of sarcasm. Sound plays just as much a part as other social and visual cues which indicate sarcasm and will be the focus of the current research project.

Sarcastically attempts to make use of this area of sarcasm to better classify sarcasm by making using of vocal cues accompanied by basic text sarcasm detection to better evaluate the results of machines understanding sarcasm. It will make use of a multi-model sarcasm algorithm which contains not only a text evaluator but also an audio and hybrid evaluation model that will be cross-compared and extracted to get the highest possible result. For this research, the core content has been based on acted-sarcasm for the purpose of generating generic sarcasm rather than advanced sarcasm which might take considerably longer time to prepare for. As such, Sarcastically's job is to evaluate the benefits or the losses of using audio in accompaniment with text to justify that audio is more or less important to understanding sarcasm as other visual or social cues.

Keywords: Multi-Modality, Sarcasm Detection, Natural Language Processing, Audio Processing, Ensemble Learning, Shared Representation

Acknowledgement

Firstly, I want to thank my mentor and supervisor, Banu Athuraliya. Without his constant enthusiasm and support in encouraging me to complete this dissertation, I wouldn't have made it this far along without his much-needed support.

Secondly, I want to thank all my co-workers from Ascentic (Pvt) Ltd. For supporting me both technically and mentally throughout the last year during my research and for helping me both catch up my work as well as giving me the right space and time to think and invest on the things I needed to. All of this has been a huge help in me completing whatever I have put up.

Third, my friends who have been nothing but supportive throughout the entire process, sharing the entire journey with me and giving me a group of people who I could talk to about anything and everything. My true mental peacekeepers.

Special thanks to all the domain and technical experts who took time out of their incredibly busy schedule to help evaluate my application and my research despite how new the field was even when they themselves didn't have the greatest of ideas about parts of my research. They still critically evaluated my project, and I am truly grateful for that.

Another special thanks goes to Gihan Liyanage and Thiloshon Nagarajah who assisted me in understanding the concepts needed work with both Text, multi-modality and ensemble learning which were detrimental in my research attempts.

Finally, my parents and my extended family. Without them I would have never even come this far in my life. They helped me not just financially but by providing the constant push and drive to keep being better than I am was, something that was inculcated into me from the very start and for that I am truly, undeniably grateful for. It is what made me into the person I am today, writing this dissertation and is something I will never forget.

Table of Contents

List of Figures

List of Tables

List of Glossaries

Abbreviation	Definition
PSDP	Project Specification, Design and Prototype
NLP	Natural Language Processing
AI	Artificial Intelligence
SVM	Support Vector Machines
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
SDLC	Software Development Life Cycle
LSTM	Long Short-Term Memory
GIF	Graphical Interchange Format
API	Application Programming Interface
NPM	Node Package Manager
MUStARD	Multi-Model Sarcastic Dataset
SRS	System Requirement Specification
GUI	Graphical User Interface
CLI	Command Line Interface
RO	Research Objective
LO	Learning Objective
SSADM	Structured Systems Analysis and Design Methodology
OOAD	Object Oriented Analysis and Design
DNN	Deep Neural Networks
MFCC	Mel-Frequency Cepstral Coefficient
ANN	Artificial Neural Network
CPU	Central Processing Unit
GPU	Graphical Processing Unit
CUE-CNN	Content User Embedding Convolutional Neural Network
KNN	K-Nearest Neighbor
SDLC	Software Development Life Cycle
JSON	JavaScript Object Notation

1 INTRODUCTION

1.1 Overview

The chapter focuses and expands on the project background, the domain of it and the reason for its implementation. It also tackles some of the challenges which existing pieces of work face and the solution the project comes up with. Later in the chapter, the research objectives and challenges faced when researching will be discussed and evaluated. Finally, a discussion over the projects scope and the feature set to be developed with it will be done and a detailed list of system requirements will be gone over to determine how powerful or sophisticated of a machine is required to run the project.

1.2 Problem Domain

1.2.1 Natural Language Processing

NLP is a division of artificial intelligence which focuses on training computers to understand free-form text and human speech. In its simplest form, natural language processing "is concerned with theories and techniques that address the problem of natural language communication between humans and computers" (Lehnert and Ringle, 2014). One goal of NLP is to engage in natural conversion with a human and many techniques over the years have given birth to conversational AI's that utilize NLP such as Google Assistant and Amazon's Alexa. NLP breaks down human speech and text into multiple areas such as Speech Recognition, Sentiment Analysis, Word Sense Disambiguation and Natural Language Generation. It builds on lower-level tasks such as learning to detect sentence boundaries to higher level tasks such as grammar identification or Information Extraction.

1.2.2 Sarcasm Detection

Sarcasm is defined as a remark or statement where the intent is in-fact opposite of what is said. It transforms the polarity of an apparently positive utterance into a negative utterance (González-Ibáñez et al., 2011). Sarcasm Detection is an important part of sentiment analysis in NLP, it is often difficult to interpret using basic methods due to the lack of available data to train such models on as well as machine-related limitations which prevent it from understanding when and where sarcasm is implied. Many approaches have been taken to identify sarcasm from text-based approaches using lexical analysis, SVM's to deeper machine learning techniques like CNNs and RNN's which bring out more accurate results.

Along with the theme of this project, multi-model approaches have also been taken in many projects such as taking video and combining features like text, image and audio (Wang et al., 2017) which relatively support each other in predictions.

1.2.3 Sarcasm Detection using Multi-Model Approaches

A multi-model approach is an approach used in artificial intelligence which uses two or more models to gain joint representations of different modalities. In simple terms, this means combining two machine learning models to improve the prediction. As previously mentioned, understanding the nuances in sarcasm is usually more successful when the context is involved. For example, the phrase "This is fine" sounds positive until the image in figure 1.1 (15 GIFs For When Your Life is a Mess but You're Pretending It's FINE, 2021) is provided alongside it indicating irony, a type of sarcasm which designed to make statements that are often contradictory to the situation. In figure 1.1, the dog in the middle is surrounded by flames while stating he's fine if a form of Ironic Sarcasm. Figure 2.2 shows a type of self-deprecating sarcasm where Ross, the actor depicted in the image is laughing at the fact that all the memes he's scrolling through relate to him.

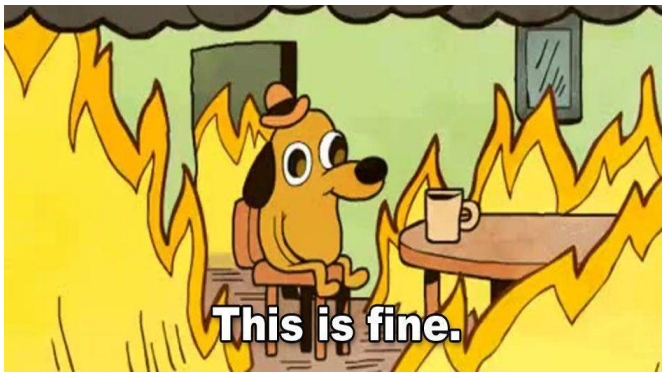


Figure 1.1 - A meme showing irony, a form of sarcasm

Figure 1.2 - A meme showing self-deprecation, another form of sarcasm

Situations like these are where machines often struggle to grasp the situation. Unless trained, machines often think of these situations are normal. So, handling sarcasm through text, while it can be done is still relatively difficult unless there's material to reinforce it, hence the reason why multi-model approaches have become increasingly more popular in sarcasm detection. Whether it be using video, images and so on. Our research will attempt to debunk the use of audio in this regard.

1.3 Problem Definition

As previously shown in section 1.2.3, until the context is provided most machine learning models find it difficult to predict sarcasm. Sarcasm is an important feature of the human language, in some cases, the inability to understand sarcasm can be in certain cases a symptom of autism or another mentally related issue. It is a basic human skill, about 8% of conversational turns between friends is a form of sarcasm called Irony (Gibbs, 2000) which further proves the previous statement. It also has a negative side where those who can't detect sarcasm as fast or as usual as others may take it in a negative sense in the form of an insult or degradation, this is presented by Colston (1997) who argued that in cases where individuals were not able to understand sarcasm, the negative impact of something such as self-degradation or friendly-insult may end up *enhancing* the negative emotion associated with the statement rather than the sarcastic side of it. Natural Language Models have been trying to replicate this unique feature of human languages for a long time and we've come closer than ever before however, effective sarcasm models are hard to implement due to the nuances of human language and the nature of sarcasm is to be ambiguous and doubtful. Due to this pure textual sarcasm models are good enough to implement but not accurate enough to depend upon since they lack fewer features for models to extract. Due to this, text reinforcement in the form of video, images, graphics, audio or even context is used to improve the accuracy of sarcasm detection. In the case of this project, the use of a refined text and audio model has not yet been implemented using proper human speech over a text to speech model, the problem lies here, training a model to understand the audio provided by a text-to-speech may provide more artificial results than natural human language which has its own set of features which help identify sarcasm. These features include the pitch of the user's voice or even or even the use of words such as "of course" or "obviously" in situations where the literal meaning might be different from the reality of the situation.

1.3.1 Problem Statement

Recognizing sarcastic quotes and text using only textual information and glyphs has proven to be effective to a degree. These results have only been boosted since then by incorporating multiple other aspects such as image and video. By focusing on audio tones and way of speech in human speech rather than artificially generated speech, it can be hoped to show more accurate results such as those done in image and video reinforced text.

1.4 Research Motivation

For years, teaching NLP based models to understand the nuances of free form speech and text has been a difficult task. When it comes to text-based sentiment analysis, it is often too difficult to interpret any sort of emotion or intent using short bursts of text alone which led many to test it alongside other indicators such as context, graphics such as emoticons and video. This proved promising as accuracy levels increased when incorporating additional factors into text analysis. However, one area stood out when deciding on this project and that was audio. Many had gone too deep levels using techniques as basic as Naïve Bayes to advanced ones like LSTMs to improve the pure text and textual graphic analysis but never touched as much when it came to combining short text bursts with audio clips of those saying such statements. This is what led to the project targeting the combination of the two to predict sarcasm.

Sarcasm is a challenging topic to predict when it comes to text analysis. So, combining it with audio is the approach being taken to detect if it is possible to better understand the text and if how it was being said was known. The current project aims to achieve this by using a multi-model approach with audio reinforced text using natural human language over text-to-speech to better tell apart sarcastic speech from non-sarcastic ones.

1.5 Existing Work

There have been multiple approaches to multi-model-based sarcasm detection using different mediums such as text and video or text and images.

Citation	Description	Limitations	Improvements
González-Ibáñez, Muresan and Wacholder, 2011	Report on an empirical study on the use of lexical and pragmatic factors to distinguish **sarcasm** from **positive** and **negative** sentiments expressed in Twitter messages. (González-Ibáñez, Muresan and Wacholder, 2011)	● The approach used here was a purely textual approach ● Because of the text-based approach the analysis model under performed in most cases against a human.	Utilize a multi-model approach to sentiment analysis in the form of sarcasm detection to boost the accuracy in prediction.

Gan, no date	Attempts to use a multi-model approach to show enhanced results over a single model based analysis model.	Used multi-model approaches using *purely text* Multi-model approach did not focus on improving a specific part of the text analysis process and focused on a generalized approach	Utilize audio reinforcement with the added use of natural voice as a training set
Bharti, Naidu and Babu, 2017	Aims to use hyperbolic words as a feature to detect sarcasm using Twitter text data	The system yet again uses a text approach but, in this case, targets a specific feature of text i.e., hyperbolic words.	We can utilize audio features human voices to reinforce the twitter feed data by getting users to repeat it both normally and sarcastically to get a valid result.
Wang *et al.*, 2017	A Select-Additive Learning (SAL) procedure that addresses the confounding factor prob- lem, specifically for neural architectures such as convolutional neural networks. (Wang *et al.*, 2017)	Using their SAL model, they are taking a generalized approach on Audio, Video and Text, this may end up with an un-biased weight causing all factors to be weighted in	We utilize a audio reinforced text model that has a greater emphasis on the audio component to distinguish sarcastic tones in human voice
Zadeh *et al.*, 2017	Introduces a new model called Tensor	While the model does show promising	The solution to this would be to take an

	Fusion Network (TFN), which learns both the intra-modality and inter-modality dynamics end-to-end. Inter-modality dynamics are modeled with a new multimodal fusion approach, named Tensor Fusion, which explicitly aggregates uni- modal, bimodal and trimodal interactions. Intra-modality dynamics are modeled through three Modality Embedding Subnetworks, for language, visual and acoustic modalities, respectively.	results, the results themselves are neutral and this is most likely due to their approach which like the previously mentioned paper, takes multiple aspects into account rather than focusing on a single area of improvement first.	approach that focuses on specific features that may help with accuracy and prediction. This may be a relatively simpler approach but doing so helps to first focus on making a specific part of the process better which after doing it iteratively can produce greater results when combined together.

Table 1.1 - Existing Work

1.6 Research Gap

The work in existing solutions have all shown promising results with improved detection across the board. But one area which hasn't been worked on extensively is in audio. While there have been studies that use video which contains audio, the audio clip itself has never been the direct subject of the research. Even in cases where the audio was the main topic, it was often done using text-to-speech. A multi-modal approach has been used in many of these areas but never

with a focus on text and audio especially with human speech, doing so would most likely bring benefit to sarcasm detection. The research carried out here on out will aim at solving this gap.

1.7 Contribution to Body of Knowledge

1.7.1 Domain Contribution

The research is focused on improving the current state of sarcasm detection. Current approaches in the area of work have used mediums such as video, imagery and audio as reinforcement agents. This research also uses audio as a reinforcement mechanism but replaces artificially generated voice cues with human-speech. The size of the contribution will ultimately come down to how accurate the result will be.

1.7.2 Technological Contribution

The contribution of this project is in the form of a GUI/CLI based application that will use a **text and audio multi-modal** approach to better predict sarcasm in natural human voice.

1.8 Research Challenges

1.8.1 Joint Representation

The first challenge faced especially when looking at a multi-model approach is how to get an accurate joint representation of both models and make them work together to provide a reasonable and accurate outcome. There will be a reasonable amount of research put into this since it forms the core grounds on which the application will be developed, without this a well-designed prototype will not be achievable. The biggest challenge that will be faced here is how to get the best of both worlds while keeping it at a level that is manageable given the time span of the project.

1.8.2 Data Gathering and Cleaning

Sarcastically tests the effectiveness of **natural human speech** as opposed to **text-to-speech** derived models, the data used needs to be gathered and checked. The reasoning behind this is due to the lack of datasets available that provide a combination of speech and text together. Most datasets out there have joint audio and video sets, but there are yet to be data sets with audio and text-focused results.

For this reason, the most likely course of action will be to crowdsource the data by asking willing participants to recite sarcastic and normal speech to help train the model to distinguish speech. Afterwards, cleaning the data and making sure only the required data is available is an important step to getting the data ready for use. An alternative approach albeit the easier one

would be to use sarcastic video data sets and extract the text by using **text-to-speech** with the questionable part being the accuracy of the speech to text model.

1.9 Research Questions

RQ1. How do previous models help accelerate the process of utilizing a multi-model approach?
RQ2. What is the degree of improvement, if at all any when human speech is added in place of text-to-speech which fails to recite sarcasm as a normal person would?

1.10 Research Aim

This project aims to design, develop and test a machine learning model to improve sarcasm detection by utilizing better audio models made with natural speech over text-to-speech.

This project will provide a machine learning model and optimistically a minor application that will test the previously mentioned questions in section 2.4 as well as conclude as to whether natural language models paired with another text model can provide better outcomes compared to a text-to-speech based project.

The application, if doable in the timespan, provides a web interface which allows a user to speak a statement and get feedback as to whether the user was speaking sarcastically or not.

The process of developing the models will also utilize one of the existing available methods of sentiment analysis for both modalities since the objective is to get a joint representation of the best possible models in both cases. The project will attempt to use simplified models at the start to test the hypothesis but then move on to test other types of algorithms that may provide better results.

1.11 Research Objectives

At the end of the research, the following objectives must have been covered or touched upon.

Research Objectives (RO)	Explanations
Problem Identification	RO1. Go through multiple interested domains and figure out a problem domain RO2. Go through previous solutions and attempts in those different domains RO3. Find a valid and feasible gap within those domains.
Literature Review	RO4. Better understand approaches taken in sarcasm detection

	RO5. Go through issues related to sarcasm detection. RO6. Evaluate other attempts at solving or improving sarcasm detection. RO7. Find texts to reinforce the identified problem and better understand the possible solutions which can be taken to solve it. RO8. Find any improvements that could be made to existing solutions that could be incorporated into research
Data Gathering and Analysis	RO9. Find way to efficiently gather the data required for training and testing the model. RO10. Select a targeted area of sarcasm to focus the research on
Research Design	RO11. Design and develop an approach to utilize a multi-model approach to the best effect.
Implementation	RO12. Build a model that utilizes the multi-model via text and human speech-based audio approach to gain better accuracy results in sarcasm detection.
Testing and Evaluation	RO13. Put the model into a user acceptance test where the plan is to make the model predict some sarcastic and normal statements and compare them to a real user and how they interpret it to understand the variation between how the model predicts sarcasm and how a person does the same. RO14. To build a suitable test model that places the model against real world scenarios.

Table 1.2 - Research Objectives

Objectives 1 and 2 will involve going through research literature mentioned in section 3.2 which goes through existing research. This will better help understand the overall domain background in terms of the projects which have now been implemented as an attempt to solve the issues mentioned in sections 1.1 and 1.2

1.12 Project Scope

Based on related projects and the aims of this project the scope of this project is defined below. Since the purpose of this application/model is essentially an attempt to improve the accuracy of sarcasm detection rather than solving sarcasm detection itself, a workable application is out of the scope. Like this, the features that are more desirable rather than required are given less attention to.

1.12.1 In-Scope

The features that will need to be covered in this project are as follows:

1. Sarcasm Detection from speech and text – A system that takes in an audio file and it's text representation and provides feedback as to whether the speech was sarcastic or not.
2. A web application that demos the above application
3. An API that will allow other users to tap into the model and train it by providing their voice models.

1.12.2 Out-Scope

1. A GUI application that lets the user record, submit the recording, and get feedback within the same app.

1.13 Resource Requirements

1.13.1 Hardware Resource Requirements

Requirement	Recommendations	Reasoning
Processor	7th Gen Intel i5 (64-bit) or higher AMD Ryzen 5 (64-bit) or higher Apple M1 or better	Required to run the operating system. A better processor also means better computational performance for CPU intensive tasks
RAM	Windows – Min. 8GB MacOS – Min. 8GB Linux (Debian) – Min. 4GB	To ensure that the application doesn't cause the system to crash or slow down
Storage	20GB or more	To store all necessary applications and data
GPU	On Windows ● Min. Integrated Graphics ● Recommended – Nvidia GTX series or higher On Mac ● Min. Intel Integrated Graphics On M1 Mac ● M1 Pro or M1 Pro Max	The system will use the CPU in absence of a dedicated GPU which is why the Integrated graphics is set as a minimum. However, it should be noted that running the model on integrated graphics may deliver **significantly slower results**

Table 1.3 – Hardware Requirements

1.13.2 Software Requirements

Requirement	Recommendations	Reasoning
Operating System	MacOS (Catalina or higher) Windows (8 or higher) Linux (Debian v10 or higher)	The operating system required to run the model should have support for large computations as well as be able to run the application built for the model by allowing microphone access.
ReactJS	-	To build the frontend for the application. Even though it's not highly necessary to use something as big as React JS, it is used due to the already developed packages available to build interfaces which will make it easier to improve and develop the application down the line.
Python	Python 3.7 or higher and Flask library	Offers a massive set of APIs, tools and libraries for Machine Learning and specifically both audio processing and NLP. Provided the frontend is not built, a CLI tool that takes a recorded file will be implemented which will require python to run it. Python will also be used alongside the Flask library

		for building an API for the project. This is a desirable so it may not be required but is mentioned here beforehand.
Documentation Tools	Zotero Microsoft Word	Tools required for referencing and project document writing and management.
Planning Tools	OmniPlan GitHub Projects	Used to prepare Gantt charts and task managment
Repository Management	GitHub	Managing the application repositories
Browser	Chromium browsers (v92 or higher) Firefox (v92 or higher) Safari (v13 or higher)	The application GUI will be built for the web.

Table 1.4 – Software Requirements

1.13.3 Data Requirements

Dataset for Learning

- MUStARD (via GitHub at https://github.com/soujanyaporia/MUStARD)

Algorithms

- Sci-Kit Learn
- TensorFlow V2
- TensorFlow V2 Keras

1.14 Document Structure

Chapter 1 is the introduction to the project, it's domain and the problem it aims to solve. The author then discusses the gap the project aims to target along with the contribution that it provides. It also talks about the objectives and aims that the author hopes to achieve with the completion of the project and finally goes about the different requirements needed to complete the project.

Chapter 2 is the Literature Review. It discusses the systems and the work of existing model that relate to Sarcastically and attempts to analyze, review and evaluate their domains, technologies and applications/ products in relation to Sarcastically.

Chapter 3 talks about the different methodologies followed by Sarcastically. It documents the timeline of the project alongside the work breakdowns, development, research and project management methodologies.

Chapter 4 is the Software Requirement Specification. This outlines and defines the way the system is expected to perform and behave. It will define the working environment of the system alongside the different techniques, findings and statistics backing the software and its creation. It will also cover the functional and non-functional requirements of the project.

Chapter 5 covers the Social, Legal and Ethical issues that the project will face and how the author has gone about either handling or mitigating them

Chapter 6 covers the system design and architecture which includes design paradigms and diagrams indicating the architecture of the system.

Chapter 7 covers the implementation decisions of the system. It covers the technology selection, tooling and functionalities of the application.

Chapter 8 covers all testing related topics including ones such as the testing criteria, model testing, benchmarking and functional/non-functional testing

Chapter 9 evaluates the project, its effectiveness, how close it came to completing the required functional and non-functional goals and limitations.

Chapter 10 is the Conclusion to the project and covers learning results of the project and how different skills were applied to it. It finally covers the limitations of the project, future enhancements, deviations and concluding remarks.

2 LITERATURE REVIEW

2.1 Chapter Overview

Sarcasm Detection is a very common topic amongst NLP researchers and has slowly been progressing overtime with more and more advanced techniques. This chapter addresses all the recent techniques and types of analysis which are used for Sarcasm Detection in general and the existing work on the field of audio and text detection in sarcasm and evaluate them against our project. Doing so will reveal pros and cons of different approaches to this project.

2.2 Concept Map

The concept graph encompassing the structure of the literature review is attached in **Appendix II.** It ensures that all avenues in the Literature Review are covered and that nothing is omitted from the process.

2.3 Problem Domain

2.3.1 Sarcasm in Text

Sarcasm is a subtype of verbal and written irony, and its usage and interpretation may vary between individuals (Suhaimin et al., 2017). Because of this variation in usage and interpretation, sarcasm as common as it maybe, is fundamentally difficult to comprehend unless one's comprehension of it is well versed. In addition to this, sarcasm also has many variations as explained by Spacey (2018). This includes types such as Irony, flattery, passive aggression and satire among others. While it is easy for us to identify this when additional context such as graphics or vocal tone is provided, it is not the same when written and spoken using nothing but raw text. Machine learning models attempt to solve this by using proven algorithms and techniques to train computers to understand these written modes.

Sarcasm is often used in the context of humor but often than not instigates conflict due to miscommunication and misinterpretation. It is usually intended to pass negative emotion in a more superficially positive way.

As per Filik et al. (2016), manual analysis of sarcasm was done by researchers with experience in linguistics and discourse analysis such as in Sykora, Elayan and Jackson (2020) through systematic approaches containing closed coding content analysis approaches listed in Thelwall (2018) such as grounded theory and sentiment analysis. This was purely focused at obtaining the surface level understanding of statements since the analysis required an understanding of how the humor or irony was used within these messages and is usually required in more

automated approaches to sarcasm detection which makes use of these features. However, many research attempts at conducting sarcasm detection using purely text with no additional supplementary features tend to deliver mixed results as it is unclear as to whether the text alone delivers (Derks et al., 2008) or does not (Walther & D'Addario, 2001) deliver sarcastic comprehension. These research attempt also aim to question whether other punctuation marks or types of emoticons deliver better results (e.g., Hancock, 2004).

2.3.2 Sarcasm Detection

For years, sentiment analysis has been one of the forefront research fields in natural language processing. From text summarization to conversational AI, the NLP research sector has been booming. Detecting sarcasm in NLP has had its own strides as shown by Patro, Bansal and Mukherjee (2019) who attempted to detect sarcasm using text-based statements by analyzing irony in their tests. To analyze something like sarcasm, we must not only consider syntactic and lexical level information but also information regarding topics like pragmatics, semantics and discourse which are highly influential when deciding whether something is sarcastic or not. Most of the recent progress, however, has been done with a focus on only the lexical and syntactic level approach. There are many techniques to detect sarcasm in text-based sarcasm such as the utilization of Naïve Bayes, SVM's and LSTMs ordered here in terms of least complexity and performance. In essence, these algorithms and techniques attempt to gain a more in depth understanding at a sentence and word level to derive sarcastic intent. The latter has shown to perform at a higher much level as per Ghosh and Veale (2016) who made use of a combination DNNs, CNNs and LSTMs which eventually outperformed recurrent SVMs. Some of the common pitfalls in Sarcasm Detection in NLP as per (Eremyan, no date) are negation, word ambiguity and multipolarity, all of which heavily impact sarcasm and are still problems in general sentiment analysis to date.

2.3.3 Multi-Modality in Sarcasm Detection

Multi-Modality in sentiment analysis is no new feat and is used in much research over the past few years. The importance of using multi-model approaches, however, has increased significantly over these years as the need for more performant and in-depth models increase. Multi-model approaches have 5 main problems to address as per Baltrušaitis et al., (2019). Representation, Translation, Alignment, Fusion and Co-Learning. These are central to multi model learning must be tackled in order to progress in this field. In the case of sarcastically, the biggest challenges would be representation and fusion as it is essential that we provide a shared representation that does not degrade or reduce the value of the dataset being used. Multi-

Modality approaches attempt to use various features such as images, video, graphics such as emoji and kaomoji in order to supplement the detection capabilities of their models. This is evident in Cai et al., (2019) and Castro et al., (2019) among others multi-model attempts where Castro uses video data to complement his text and Cai uses images to complement his model. Another such case can also be seen in Suryawanshi et al., (2020) who utilizes images and text together in the form of a meme to detect sarcasm, he was able to obtain measurable results between 0.44 and 0.67 in precision without suffering in the recall aspect of his model which further elaborates on the potential performance of multi-model detection approaches.

2.4 Existing Work

2.4.1 Existing Research in Sarcasm Detection

Sarcasm detection in general has been a relatively researched topic under sentiment analysis with most research utilizing more than one feature to determine it. However, there have also been successful attempts at using singular features complimented by other linguistic features. Whether this be background text explaining the context (e.g., Bamman and Smith, 2015), video, images (e.g., Wang et al., 2020), emojis (e.g., Subramanian et al., 2019) or anything along those lines, sarcasm detection has come a long way with so many ways of being improved. As mentioned in Chapter 2.3.2, in order to detect if a statement is sarcastic or not, many factors such as the semantic and pragmatic features need to be taken into account as they are key indicators of sarcastic intent. Kreuz and Caucci (2007) carried out a study on lexical influence on sarcasm detection and found out that interjections and punctuation were strong indicators of sarcasm in text (Ren et al., 2018). This proves that minute details in the way statements are written, said or read have a massive impact on an individual's perceived notion of sarcasm and can heavily influence decision making in this regard. Another study focused on the acoustic cues of sarcasm stated that the most obvious cue to sarcasm is the tendency to raise or lower vocal pitch (Banse and Scherer, 1996) this emphasizes the significance of audio-based cues in sarcasm detection in a non-technical standpoint.

2.4.2 Existing Research in Text-Based Sarcasm Detection

One of the popular lines of research is done using user writing styles as outlined in Hazarika et al., (2018) who made use of a fusion technique combining stylometric and personality features into an embedding for each user using a technique called Canonical Correlation Analysis (CCA) introduced in (Hotelling, 1936), which allowed them to achieve major improvement results against models such as Bag-of-Words (Dave and Desai, 2016), CNNs (Poria et al., 2016) CNN-SVMs and CUE-CNNs with the exception being CUE-CNNs without personality

features which dropped the accuracy rating of the CASCADE model. Like this, many research attempts have been made to use existing language features using purely text based analytical approaches to improve overall sarcasm detection. Similar work to this was done by Tsur et al., (2010) by utilizing 6.6k manually annotated amazon reviews using kNN-classifiers over punctuation-based and pattern-based features (Poria et al., 2016) and again by (Davidov et al., 2010) when they attempted to utilize n-grams and unsupervised learning with sentiment features. As such there were many research attempts using purely text-based methods, however they were limited to the lack of context provided in most cases as shown by (Poria et al., 2016) who even though gained a high level of performance in comparison to other baseline models in their project, were limited by the size of their corpus among other missing pieces. This is a major drawback. However, chapter 2.3.4 explains how a single feature model focused purely on text can reap lesser results than that of a "Multi-model" approach utilizing more than just text. It must be noted however that the aims of these research aren't to compete but rather to evaluate how much can be derived from each mode of research.

2.4.3 Existing Research in Multi-Model Sarcasm Detection

Multi-Model research attempts to make use of multiple stand-alone models which provide additional context cues for a statement. This enhances or even replaces general lexicon-based approaches to sarcasm detection by making use of more advanced individually focused models based on features such as images, video or audio. Even though this specific research utilizes only 2 models (text and audio), multi-model approaches can utilize more than 2 models to further improve results by combining more advanced features. Castro et al., (2019) utilized multi-modality-based approaches in their paper which aimed at improving sarcasm detection whilst utilizing models created to detect text and a combination of audio and video. They compared this approach to typical sentiment analysis techniques utilizing SVMs and Random Forest among others as a baseline. Their results showed a dramatic increase in both error rate reduction across the board as well as higher results in prediction, recall and the f-score as shown below using a speaker-dependent setup

Algorithm	Modality	Precision	Recall	F-Score
Majority	-	25.0	50.0	33.3
Random	-	49.5	49.5	49.8
SVM	T	65.1	64.6	64.6
	A	65.9	64.6	64.6
	V	68.1	67.4	67.4
	T+A	66.6	66.2	66.2
	T+V	**72.0**	**71.6**	**71.6**
	A+V	66.2	65.7	65.7
	T+A+V	71.9	71.4	71.5
$\Delta_{multi-unimodal}$		↑ 3.9%	↑ 4.2%	↑ 4.2%
Error rate reduction		↑ 12.2%	↑ 12.9%	↑ 12.9%

Figure 2.1 - Results of Castro et al., multi-model approach

As shown above, the model improves in all areas against baseline readings for other algorithms and techniques used in the project which proves the effectiveness of multi-model approaches in this scenario. Similarly, Cai et al., (2019) also conducts a similar study utilizing twitter posts to improve sarcasm detection utilizing a combination of text and images. Her models utilize attribute feature representation and text feature representation for her audio and text models respectively, both of which utilize some form of CNN to deliver results. The research performed on sarcastically will utilize an approach used by Cai et al., (2019) who also was inspired by Gu et al., (2018a) which used a technique known as "Modality Fusion" to combine the results of the text and audio models instead of adding them together as part of a much longer vector. This would result in a single fixed length vector that can be used to train the accumulator model which gives the final output.

2.5 Approach

The approach for Sarcastically's implementation must be broken down into multiple steps due to the use of a multi-model approach. For each step, the author will discuss the different approaches available but also the approach that was taken for the specific step and why.

2.5.1 Approach for Text Model

The text model in Sarcastically is required to understand the text portion of detection, this is the text that can be voluntarily supplied by the user or extracted directly from the audio itself. For implementing the text model there are various approaches available from different research which focus primarily on text-based input, some of which will be discussed below. The research will be focused on using various machine learning algorithms rather than other available alternatives to bring out the best performance and accuracy to test against other baseline models which also use some form of machine learning.

Sarcasm Detection can be categorized as a classification problem where the output delivers either a positive or negative result and is considered a form of supervised learning. Classification models are trained to understand the polarity of this text to detect the positive or negative result mentioned prior. Classifiers such as Naïve Bayes (John and Langley, 2013), Logistic Regression (le Cessie and van Houwelingen, 1992) and Scalar Vector Machines (SVM) (Keerthi et al., 2001) are all examples of classifiers that are famously used across many NLP related research projects. Supervised learning in general requires labelled data to train and output the most suitable result, when this is not provided the machine learning model would have to carry out some form of clustering or manual annotations beforehand for it to be trained properly. In the case of Sarcastically we would use a type of classification known as Binary Classification since the data required to carry out the training and obtain a result both consist of only 2 possible values, positive or negative. While the dataset used to train the model does contain around 5 classes enabling multi-class classification, we will only make use of 2 values, the label (the statement which is said) and its result (sarcastic or non-sarcastic). This opens the following algorithms for use.

Algorithm	Benefit	Drawback
Naïve Bayes	• Doesn't require much training data • Simple to implement • Fast and capable of real-time prediction	• Assumes that all features are independent which is rarely the case in real life • Faces the zero-frequency problem where it assigns zero probability to variables whose data wasn't provided in the training set.
SVM	• High Accuracy predictions • More memory efficient • Works well when there's a clear margin of separation between features	• Not suitable for large datasets • Inefficient when data contains noise

	• Works better in high-dimensional spaced data	
K-Nearest Neighbor	• No Training Step, only produces tagged data • Simple and intuitive to understand and implement • No Assumptions need to be made when using it.	• Does not work well with Large datasets • Extremely sensitive to noisy data • Does not work well with data having high dimensions
Neural Networks	• Easy to implement and use • Fault tolerant • Can work with limited data • Ability to parallel process data	• High hardware dependencies • No assured topology/structure • Does not explain inner reasoning for results.

Table 2.1 – Algorithm choices for text model

The author will attempt to use Neural Networks for this project since it is important that we get the best result with the limited data that we have and at the same time be easy to implement with little difference between the tooling required to implement the audio model as well. The other classifiers mentioned here would be used as benchmarks for the final model (after applying the audio and accumulator models)

2.5.2 Approach for Audio Model

The purpose of the audio model is to extract the sentiment from statements when spoken to be used as a supplement to the text model to both improve accuracy as well as to accommodate for the lack of data. Audio classification can be subclassed into the following,

1. Acoustic Data Classification
2. Music Classification
3. Natural Language Classification
4. Environment Sound Classification

For this research, the most appropriate classification falls under natural language classification which is generally used for projects aimed at natural language processing. The same models used above in 2.5.1 Table 2.1 can also be used to classify data in the audio model. We will

make use of a CNN which is a form of Neural Networks which has a "deep-forward" architecture (Indolia et al., 2018) simply put, it is "a network mapping which defines **y = f (x; θ)** and learns the value of parameters **θ** that results in the best function approximation" (Goodfellow et al., 2016). The CNN will train on the output of another model which will extract the MFCC of the audio files which will be split into 1 second chunks which will allow us to read and analyze each frame of the audio. This will then be converted an "average vector" which will take the above MFCC values and convert them into a vector representative of the average of all accumulated vectors, these will be fed to the audio model to be trained on. Before all this however, the audio will be converted to a .wav format rather than .mp4 mainly because of a few reasons as listed below,

1. .wav uses a lossless format compared to .mp4 which uses a lossy format. The shift to .wav results in less audio loss since the audio itself isn't compressed.
2. .wav uses LPCM which shows audio waveforms as a sequence amplitude values via a sample based on a linear scale which for this research will provide a structure we can use for pre-processing.

2.5.3 Approach for Accumulator Model

The purpose of the accumulator model is to combine the resulting vectors of the audio and text model. This will be the final output of the model and the output here will determine whether the statement will be sarcastic. The accumulator uses a technique called shared representation which combines the output vectors of the audio and text model (Ngiam et al., 2011) and outputs a vector that is representative of the joint vectors. How this joint vector is represented can be argued but one approach which will be used is called Modality Fusion as used in Gu et al., (2018a). Modality Fusion comes 3 distinct forms Early Fusion (a.k.a. Data Level Fusion), Late Fusion (a.k.a. Decision Level Fusion) and Intermediate Fusion. (Khaleghi et al., 2013; Lahat et al., 2015)

Early Fusion is the more traditional approach to data fusing before attempting to use it in an analysis (Khaleghi et al., 2013). There are 2 ways of approaching this, one is to remove the correlation between two features to make them independent. The second approach is to fuse data at a lower dimension where the data in relation to the other feature is common. This can be done via statistical techniques such as canonical correlation analysis or independent component analysis. This is applicable in situations where raw or pre-processed data is present.

A challenge of this approach is combining data sources into a singular vector representation since it's important that the data must be synchronous.

Late Fusion, also known as decision level fusion, uses independent data sources followed by fusion at the decision-making stage. This often results in better performance since everything up to error handling is handled independently and thus are uncorrelated. Ramachandram and Taylor (2017) however, argue that there is no conclusive proof that it performs better than early fusion. There are multiple projects however, that do make use of this approach. (Simonyan and Zisserman, 2014; Wu et al., 2016). The best-case scenario for using late fusion is when the modalities or sampling rates of the input data are extremely different in nature.

Intermediate fusion is built on top of deep neural networks. It allows data fusing at multiple stages in the process and thus makes it the most flexible out of all three fusion methods. It converts changes in data into a level of higher representation via multiple sub-layers. Each of these sublayers runs functions which skew, scale, shift or swing the data around, providing a new representation of the input data. This allows deep learning models to fuse multiple layers of data into a single fused layer so that they can learn the different shared representations of each set of data. Because of how intermediate fusion works, it inherently gains a risk of quick overfitting or the case where the network fails to understand the relationship between each modality in the data. A way to improve this would be using Principal Component Analysis (PCA) as per Yi et al., (2015) which is a tried and tested technique or reducing the dimensionality of the input data like demonstrated in Ding and Tao's (2015) multi-model approach for image recognition.

2.6 Chapter Summary

This chapter went over the different literature which discuss both the domain and implementation of Sarcastically and elaborates the need for a project such as this. Additionally, the author discussed about the different approaches available for the implementation of each model in Sarcastically and showed the preferred and most-suitable choice for the prototype. Finally, the author goes through the different tools and techniques used to build both Sarcastically and its current "competitors" while also critically evaluating the algorithms which will be used for detecting the state of sarcasm.

3 METHODOLOGY

3.1 Overview

The methodology chapter covers the methodologies followed for the research project in terms of research, development and project management. Comparisons to those methodologies are presented alongside diagrammatic representation of the work plan and schedule of the project with some deliverables given at the end

3.2 Research Methodology

The best projects start with a great plan and this section helps us systematically and carefully solve the research problems encountered. With that, the following table gives a list of methodologies that were selected to suit the current project.

Research Methodology	Philosophy	Research philosophies consider 3 core concepts. Ontology, which is the view of the nature of reality, Epistemology, which is the view of the nature of knowledge and Axiology, which is the view of roles and ethics in the research methodology. And within these there are the views of Realism, Positivism, Pragmatism and Interpretivism (GuhaThakurta and Chetty, 2015) out of which, the **Pragmatist Approach** was chosen The reason for the pragmatist approach is due to the subjectivity and nature of sarcasm and the way it is interpreted among different groups of people. The research results will be **subjective to the writer's interpretation of sarcasm** and in addition to this, the method used to score the degree of sarcasm predicted by the model will be **highly quantifiable.** Ultimately however the actual results of the sarcasm detection model, will be tested amongst others as well to gain a better understanding of how well the sarcasm was understood.

	Approach	The research approach focuses on the relevance of the hypothesis in comparison to the study and the steps taken for collecting data and interpreting it. For this, a deductive approach was selected with a **combination of both qualitative and quantitative analysis.** Reason being that in terms of something as vague and subjective as sarcasm, a range of evaluations are required to determine its validity such as a community survey which compares the models answer to that of a real person which will help with the subjectivity problem. While the "F-Score" among other popular statistical measurements could be used to detect accuracy at a text level and avoid false positives
	Strategy	The research strategy outlines the plan for conducting a research study (Johannesson and Perjons, 2014). Out of the available research strategies such as Surveys, Ethnography, Interviews, Case Studies etc. the strategy being used will be **Phenomenology.** The reason for this choice is again due to the subjectivity and other nuances involved with sarcasm. To get a more comparatively accurate result against real-world assumptions of sarcasm. Ultimately the results of the model would have to be compared to the results of real people guessing whether a statement is sarcastic or not based on their experiences. This, while still bringing in a level of subjectivity will still bring a valid comparison point

		to place against our model which may help in bringing about a more accurate response model.

Table 3.1 - Research Methodology

3.3 Development Methodology

3.3.1 Life Cycle Model

The project in question will require a lot of changes going forward in terms of both finding solutions as well as data gathering. This also requires multiple iterations of trial and error to find and mitigate any possible issues that may arise during development, data gathering or even testing down the line.

Because of these requirements, the Agile lifecycle approach was picked over other available SDLC approaches. One of the many benefits in agile is continuous and adaptive planning where issues are resolved on the fly as opposed to the very start like in waterfall methodologies among others. This helps us with the rapidly changing environment of this project. Ultimately however, Agile will allow us to deliver a value-first working and tested prototype quicker than in other situations. However, one disadvantage of agile is that it is less predictable since not all issues are addressed at the start which may lead to either a project that fails to meet some requirements or additional time added towards the end of the project, another disadvantage is that the project can likely fall off track due to the very little planning required at the start to get going. This improper delivery of major or minor details may eventually show its form down the line in development.

3.3.2 Design Methodology

For this, an Object-Oriented Design & Analysis Design Methodology (OOAD) was chosen for a few reasons,

1. It follows a divide and conquer strategy to handle large complex systems. In the case of this project. The research project requires us to do multiple iterations of trial and error, so breaking down the application into small manageable deliveries will help speed up the process as well as help deliver a quality application (Object Oriented Analysis & Design Tutorial, 2021)
2. Object Identification and representation; This allows us to take the overall system and put it into the scenarios of real-world object comparisons to better understand and design the system. (Object Oriented Analysis & Design Tutorial, 2021)

3. Optimization; This design approach allows for the best level of optimization in the application since the structure of the approach allows us to arrange computational tasks in their most efficient order and removes dead or unused parts of the application when not in use allowing for greater performance. (Object Oriented Analysis & Design Tutorial, 2021)

3.3.3 Evaluation Methodology

The system will have 2 main methods of evaluation. One will be statistic based by using measures such as the "F-Score" for accuracy in terms of how sarcastic a statement is. The second will be a subjective real-world test where another real-life user would evaluate the system, this will be done over multiple test iterations to make sure that the outcome is representative of multiple "definitions" of sarcasm.

3.4 Project Management Methodology

The project management methodology is simply a set of principles and guides to plan, manage and execute a given project, below is a breakdown of what methodology we will be using and further sub-sections will discuss the steps and other related information regarding the methodology.

3.4.1 Schedule

The GANTT chart in **Appendix III** will show the schedule of the entire project timeline until completion.

3.4.2 Deliverables

Deliverable	**Date**
Submission of Final Project Proposal	November 4, 2021
Submission of ethics form	November 4, 2021
SRS Submission	November 25, 2021
Proof of Concept V1	December 6, 2021
Test and Evaluation Report	March 17, 2022
Draft Project Report	March 31, 2022
Final Project Report	April 28, 2022

Table 3.2 – Deliverables

3.4.3 Risks and Mitigation

3.4.3.1 Data Gathering

The primary source of data will be already gathered video datasets which will be passed through a speech to text tool to get the relevant text messages out of the video and through an

audio extractor to get only the audio feature from the videos. On the chance that the video datasets don't fit the bill then a crowd sourcing method would have to be used where sarcastic text datasets will provide the data for sarcastic text and an app will be built to crowdsource audio clips from individuals to build the audio model.

3.4.3.2 Audio Cleansing

The audio cleaning will be an essential part of making sure the audio provided can be clearly heard for feature extraction to be less plagued with noise. Since the crowdsourcing will use mobile microphones a large amount of time will go into cleaning this audio before it is used for processing.

3.5 Chapter Summary

The chapter covered all the methodologies of the project and their comparisons. Alongside this the work plan of the project was covered and the deliverables and risk mitigation methods were presented towards the end.

4 SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Overview

The following section will outline and define the way the system is expected to perform and behave. It will define the working environment of the system alongside the different techniques, findings and statistics backing the software and its creation. The last few sections will include the essential and non-essential requirements of the system to perform at its best and finally some descriptions on the flow of system operations.

4.2 Rich Picture

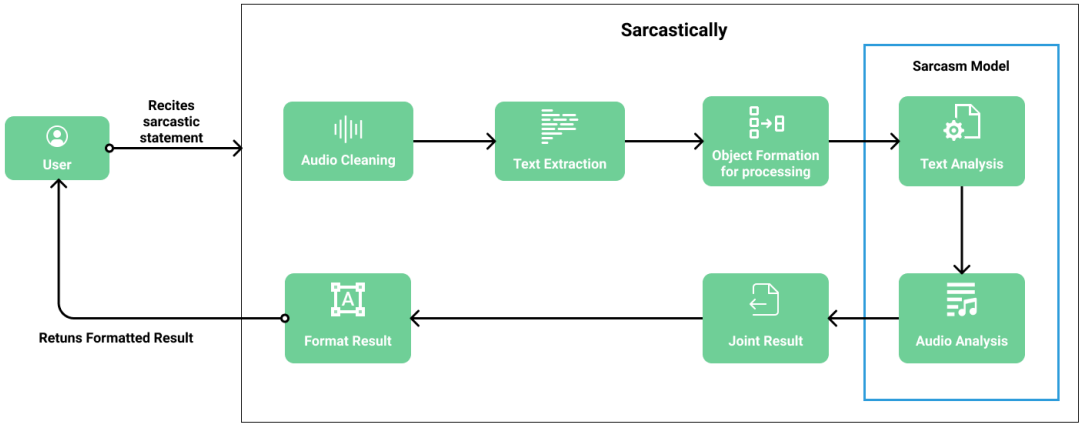


Figure 4.1- Rich Picture Diagram

4.3 Stakeholder Analysis

The stakeholder analysis shows the different identified stakeholders within the system, their roles and a small description of their involvement in the system.

4.3.1 Onion Model

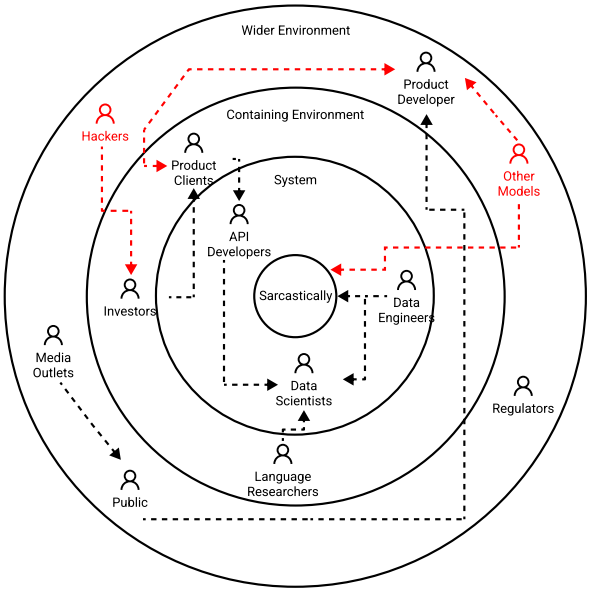


Figure 4.2 - Onion Model of Stakeholders in the system

4.3.2 Stakeholder Viewpoints

Stakeholder	Role		Benefits
Data Engineers	Operational	Maintenance	Develops better models using tool
Data Scientists			
API Developers	Functional	Development	Uses model to build API's for better using the model's functions
Investors	Financial	Beneficiary	Invests on tool with the hope of making financial benefits out of it
Product Clients			Uses model/tool to enhance or create products
Product Developer			
Language Researchers	Research	Expert	Will aid in research attempts to better understand Sarcasm and help to provide better opinion
Other Models	Negative	Competitor	Analyzes flaws in the current model to make a better or more efficient model
Hackers		Security Risk	Misuse data used by the model or misuse the model itself.
Media Outlets	Social	Beneficiary	Can use the model to help determine levels of sarcasm when interviewing individuals or even verify and validate external stories
Public			
Regulators	Privacy & Security	-	To validate whether any security or privacy laws are

			broken by the model and to validate its safe use.

Table 4.1 - Stakeholder Viewpoints

4.4 Selection of Requirement Elicitation Techniques

Requirement Elicitations are about the numerous ways of collecting and collating requirements for the project. The following sections will run through the different pros and cons of the different methods and run through their results and facts in as much detail as possible.

4.4.1 Questionnaire Distribution

To better understand the requirements for the project, the author needed to better understand the limitations and issues with a real-life audience when it came to their perspective of sarcasm. To do this, a form collecting various information regarding different users and their various perspectives was released via Google Forms

Advantages	Disadvantages
• Ability to reach a wider audience in a shorter period. • Anonymity • Ready-made statistics • Comparability	• Dishonest answers • Misunderstandings and misinterpretation of questions being asked • Accessibility issues for users who are unable to read or hear or maybe even physically limited. • Questionnaire Fatigue where users may get exhausted mid-way through the survey.

Table 4.2 - Pros vs Cons of Requirement Elicitation via Questionnaires

4.4.2 Comparison Against Existing Systems

Comparing the idea of the project against existing systems in the same area of work can help by providing us with feature parity and potential issues in existing systems which can help us identity possible requirements for the project.

Advantages	Disadvantages

• Ability to identify possible gaps or limitations in Sarcasm Detection systems • Benchmarks can be obtained for system features	• Requires a large investment of time to analyze the existing systems and find their gaps and limitations. • May require experience in other areas of expertise which may end up leading to additional learning time

Table 4.3 - Pros vs Cons of Requirement Elicitation via comparisons with existing systems

4.4.3 Literature Reviews

Literature Reviews allow us to look not only at the overall idea of the system but the thought process and gain an in-depth understanding of how the inner workings of the system perform.

Advantages	Disadvantages
• It is possible to confirm and validate details by using references to expert opinions, studies and reviews. • Gain an in-depth understanding into the thought process and inner-workings of the system.	• Requires a large amount of time to read and understand documentation. • Often requires additional checks to validate the popularity and validity of the documents in question.

Table 4.4 - Pros vs Cons of Requirement Elicitation via literature reviews

4.4.4 Brainstorming and Self-Evaluation

Brainstorming can often be a creative and quick way of getting new feature ideas and possible requirements for the system. It can be done alone or even with groups of people known and unknown.

Advantages	Disadvantages
• Can be effective when creative ideas are explored • Can be done alone or with many people	• Easy to over or under-estimate • Easy to get distracted with desirables rather than necessities • Can take an unnecessarily longer time than other methods

Table 4.5 - Pros vs Cons of Requirement Elicitation via brainstorming and self-evaluation

4.5 Discussion of Results

4.5.1 Questionnaire

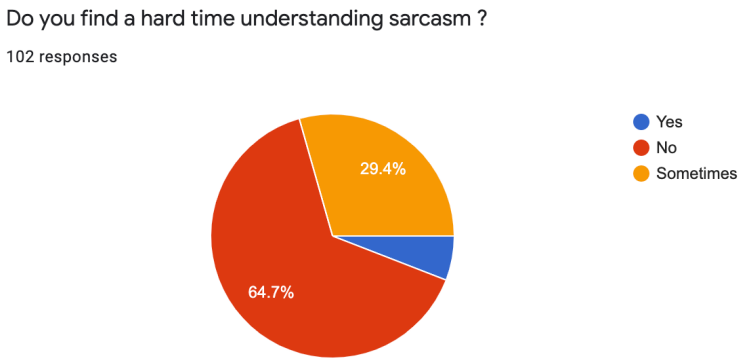
Question	Do you find a hard time understanding sarcasm?
Aim	To understand whether an individual considers him/herself well versed in sarcasm
Response	
	
Observation	
This question was used to get a ground understanding of the audience's understanding of sarcasm before diving into the sarcasm test that was done later in the questionnaire. Very minor 6% claimed they found it hard to understand sarcasm this is but a small substratum of the overall audience but individuals like this enforce the need for AI's that can understand and aid these individuals.	

Table 4.6 - Results of Questionnaire – Question 1

Question	Do you feel uneasy when someone uses sarcasm, and you can't really understand what their intent is?
Aim	To understand whether understanding sarcasm can cause frustration or other emotions when not properly understood

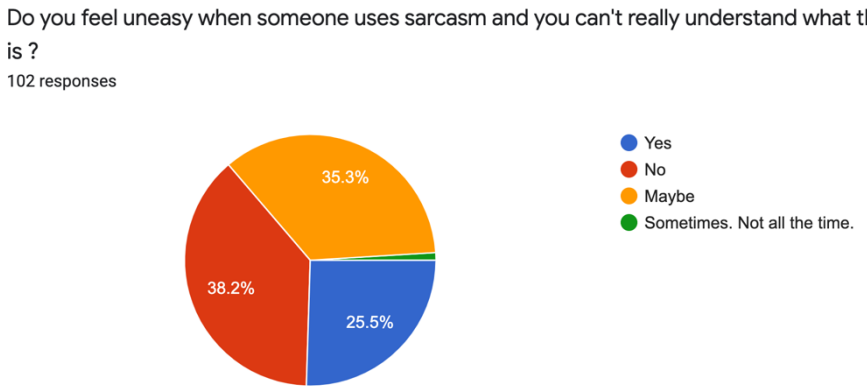
Response

Observation
This question was asked to find out whether there were any negative emotions involved with sarcasm when it wasn't understood properly. Firstly, the response "Sometimes, not all the time" was erroneous due to a questionnaire mistake and this falls under "Maybe". But the results here are interesting that a total of 60.8% were either conditionally uneasy when they couldn't understand the sarcasm being used and 25.5% of that total were confirmed to be uneasy when they didn't understand. This could be due to a various number of reasons such as misinterpretation, misunderstanding and other related reasonings. The author reinforced this question with question 3 mentioned below to understand other features that help to evaluate sarcasm.

Table 4.7 - Results of Questionnaire – Question 2

Question	Does the tone of someone's voice help you tell when that person is being sarcastic?
Aim	To understand whether vocal tone can help alleviate confusion with sarcasm

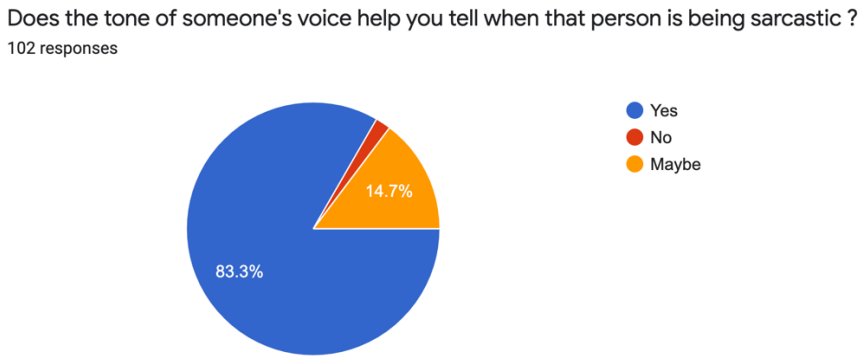
Response

Observation
The above question yielded a good result which showed that more than 80% of participants admitted that hearing someone's vocal tone helps them to better understand sarcasm without context. This is backed up by a companion question which gave the participants a choice of more indicators which may have helped them better evaluate sarcasm.

Table 4.8 - Results of Questionnaire – Question 3

4.5.2 Sarcasm Test – Without Audio

A test of a user's ability to understand sarcasm through a series of questions containing sarcastic and non-sarcastic statements with pre-defined answers and comparisons of how users analyzed their answers.

Question	I'm not saying I hate you, what I'm saying is that you are literally the Monday of my life.
Intended Answer	Sarcastic

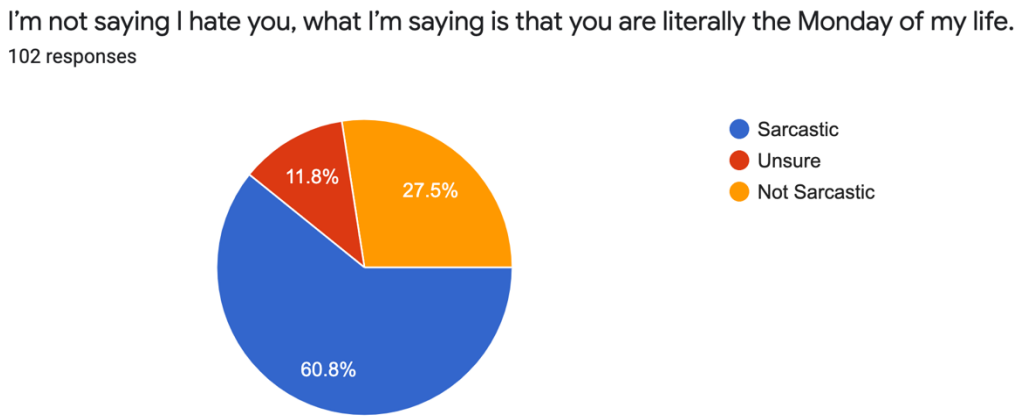
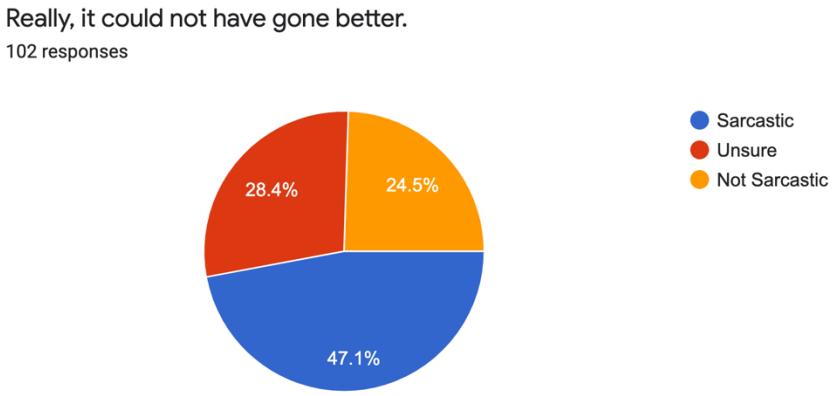
User Responses

Analysis
This question was intentionally made obvious in order to give users a base level understanding of what might be asked in the following questions. Unsurprisingly, a majority of participants were able to understand that it was a form of sarcasm known as Passive Aggression. However, a surprising 39.3% were either unsure or guessed incorrectly which may be due to previously mentioned issues such as misinterpretation and misunderstanding.

Table 4.9 – Questionnaire Sarcasm Test Question 1

Question	Really, it could not have gone better.
Intended Answer	Sarcastic
User Responses	
	
Analysis	

This question again was made to be sarcastic but was made slightly more difficult thant he first since the statement is no longer direct and can be taken in multiple ways. This type of sarcasm via text is often misinterepreted and as expected, a majority of participants were either unsure or marked the question are non-sarcastic.

Table 4.10 – Questionnaire Sarcasm Test Question 2

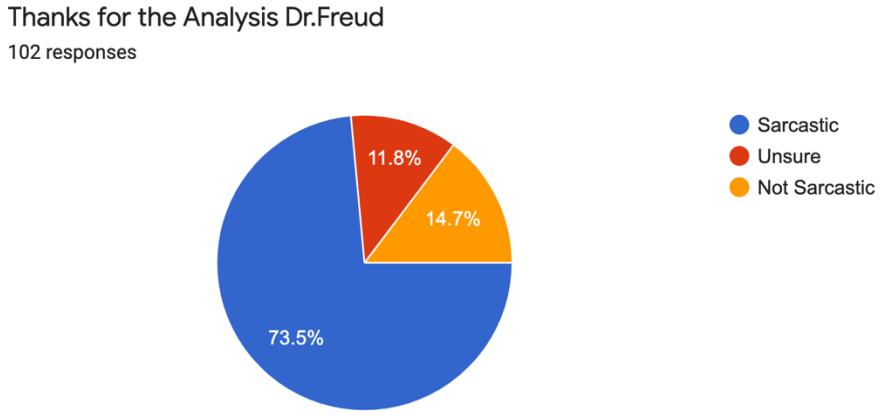
Question	Thanks for the Analysis Dr.Freud
Intended Answer	Sarcastic
User Responses	
	
Analysis	
The question here on one hand. Is Sarcastic, however this is mostly due to the fact that the statement which is the question, is a very famous sarcastic statement used in multiple TV Shows and Movies. Consdiering that more than 70% of all participants were between the ages of 18 to 30. It is highly likely that they have heard and seen this phrase being used in said shows/movies. The fact that the remaining percentage of participants guessed incorrectly or were unsure, means that like explained previously in Table 2.10, the statement can be considered to be 100% seriously if taken in the correct context which means that trying to predict sarcasm in these instances using purely text without context can be a difficult task to achieve human and computer alike.	

Table 4.11 – Questionnaire Sarcasm Test – Question 3

4.5.3 Sarcasm Test – With Audio

This test follows the same pattern as the above with the addition of voice clips that the user would play to guess whether the statement was sarcastic or not. A disclaimer however is that

since the voice clips were never made mandatory to be played, we assume that all users who went through the form, listened to the audio clips before they answered.

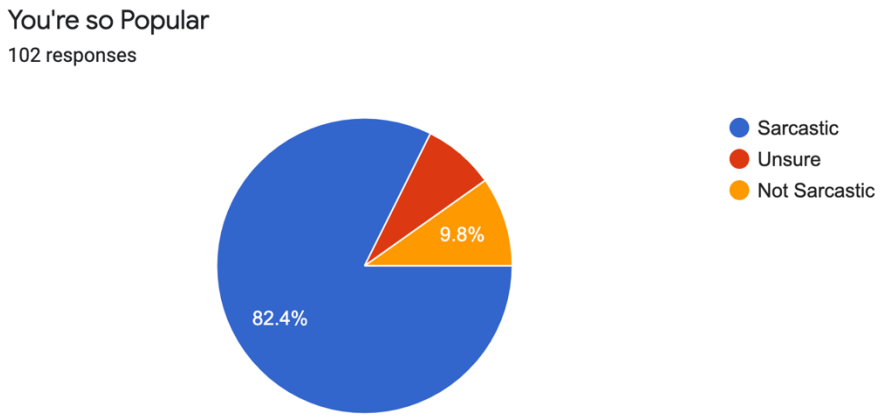
Question	You're so Popular
Intended Answer	Sarcastic
User Responses	
	
Analysis	
The clip here was made as sarcastic as possible to gauge whether participants could better understand the sarcasm using voice as an indicator. As expected a majority of more than 80% was able to correctly identify the sarcasm in the statement. For those who did not answer correctly, a good assumption would be that these individuals may rely on other indicators such as Body Language in order to better understand sarcasm. These are visual learners and most likely do not gain sarcastic cues via audible sources as well as others	

Table 4.12 – Questionnaire Audio Sarcasm Test – Question 1

Question	You're so Popular (Second Variation)
Intended Answer	Not Sarcastic

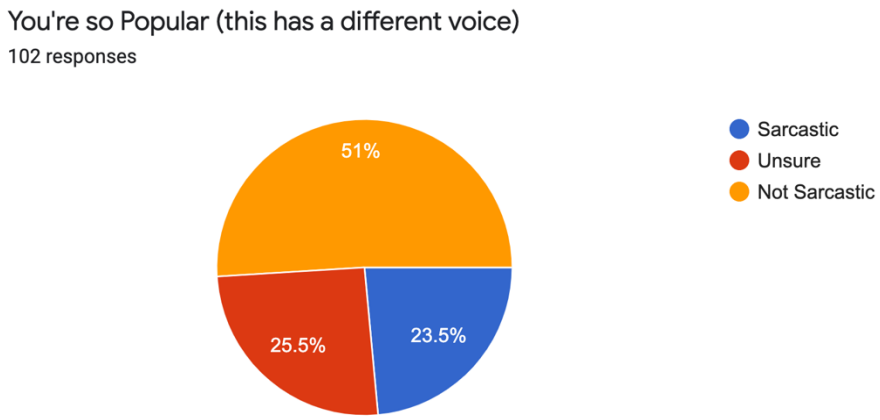
User Responses

Analysis
This clip was intentionally done with the same phrase in order to test whether users would react differently to the same statement if it was said in another way. The audio clip used in this scenario, was less sarcastically toned relative to the first question and as expected 51% of all participants gussed correctly while the other 49% seemed to struggle a bit more. The 49% could simply be due to a misunderstanding in terms of vocal lines. The audio clip makes use of sarcasm interpreted in the voice of an american which may be significantly different to sarcasm used in different reigions. This may have led to the confusion faced by the 49%.

Table 4.13 – Questionnaire Audio Sarcasm Test – Question 2

Question	Are you kidding?
Intended Answer	Not Sarcastic

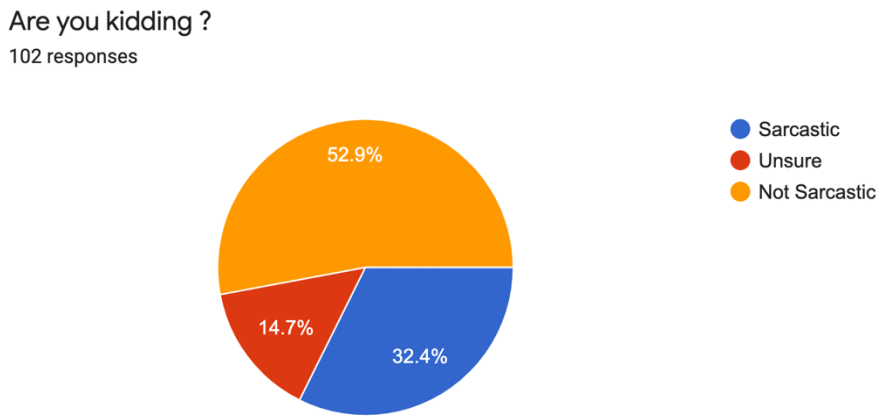
User Responses

Analysis
This clip was nothing out of the ordinary and was kept as plain as possible to coerce users into picking Not Sarcastic, which seems to have worked out for the most part. Most users correctly guessed this statement as a non-sarcastic statement which it was. The users who termed this phrase as sarcastic may have been under the impression that all the statements were sarcastic due to the varying voices being used. However, this could also be due to simply the requirement of more context and is a very common occurance in sarcasm especially in machines when context is everything.

Table 4.14 – Questionnaire Audio Sarcasm Test – Question 3

4.6 Summary of Findings

4.6.1 The Questionnaire

The questionnaire could be summated using the following pie chart which describes the overall feelings of all participants who carried out the survey.

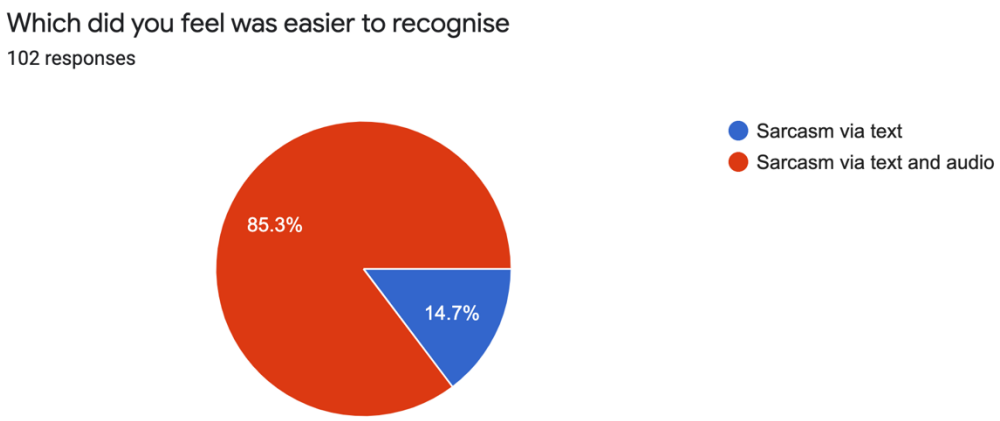


Figure 4.3 - Overall spread of which indicators users found easier to understand sarcasm with

Overall, participants seemed to have both yielded better responses as well as favored questions that were accompanied by an audio clip which helped them better understand the context. Usually this would be done over emojis and other graphical aids via text. But the combination of audio and text is one that when used correctly can gain better outcomes especially in terms of machine understanding of these statements. Most participants said that the audio reinforced statements were easier to understand for some of following reasons:

1. It is easier to understand the tone and get more context
2. In my opinion it's always easier to recognize sarcasm using the way someone sounds. I think it has a lot to do with the way he/she makes the specific statement (tone).
3. Voice generated more emotions with tonality
4. Plain text is just not enough to recognize sarcasm unless it's a common set of text people usually use when they're being sarcastic(which really isnt always the case). Hearing the tone in someones voice really confirms the intention of the words being said(specially when it comes to deciding if it was serious or sarcastic)
5. The tone of voice changes according to the expression and motive of the speaker. Sarcasm has it own style and modulation of tone which varies from person to person.

The above reasons were all extracted via responses generated from the questionnaire. More than 100 responses were provided but the above 5 were selected from them to represent the overall idea which was passed by most of the 100+ responses.

4.7 Context Diagram

The context diagram is a level 0 data flow diagram which shows a high-level overview of the relationships between the system and its external actors. It helps us identify the boundaries of

the system immediately without the need for any technical knowledge which can benefit many stakeholders who require an understanding into the application

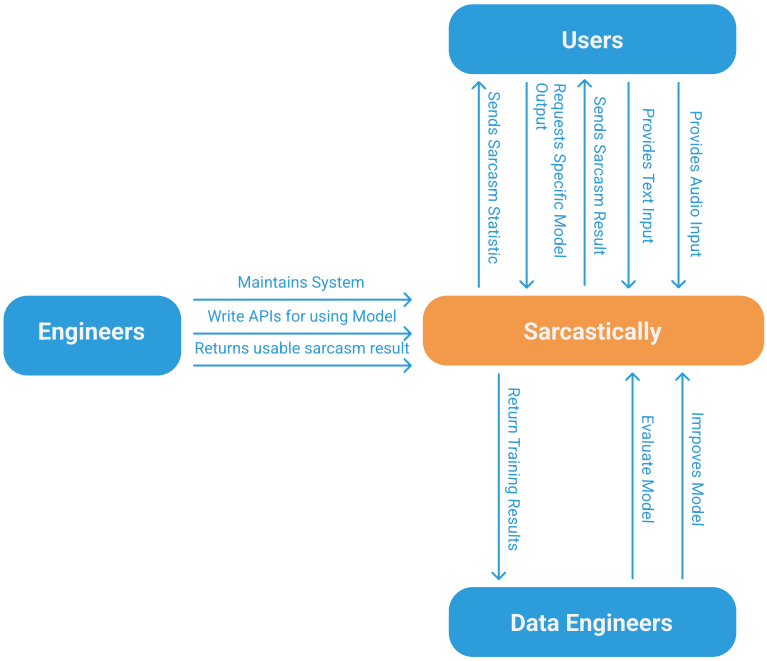


Figure 4.4 - Context Diagram

4.8 Use Case Diagram

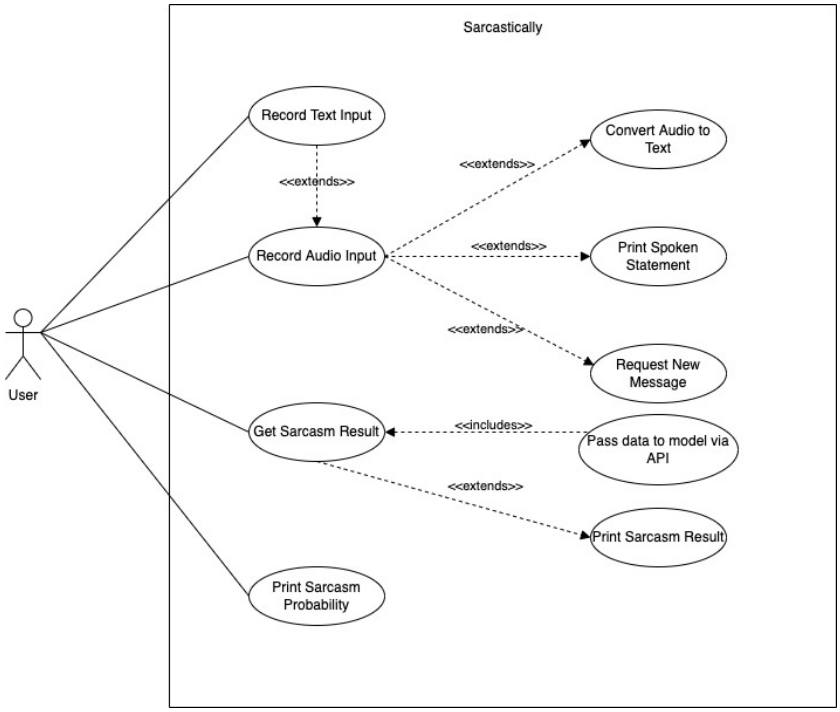


Figure 4.5 - Use Case Diagram

4.9 Use Case Descriptions

Use Case	Record Text Input	
ID	UC-1	
Goal in Context	Users are able type text into an input box for processing	
Actors	Primary Actor	
	User	
Scope	Primary Task	
Pre-Conditions	User must be connected to a network	
Post-Conditions	User can submit text to model for processing	
Trigger	User enters text in text area	
Description	**Step**	**Action**
	1	Submit button is enabled
	2	User types into text area
Variations	Exception Flow	
	Step	**Action**
	1	Submit button is enabled
	2	User types into text area

Table 4.15 - Use Case Description : Record Text Input

Use Case	Record Audio Input
ID	UC-2
Goal in Context	Users can record an audio clip of themselves reciting a sarcastic or non-sarcastic statement for processing
Actors	Primary Actor
	User
Scope	Primary Task
Pre-Conditions	- User must be connected to a network - User must have a working microphone

Post-Conditions	- Spoken text is displayed on the screen for validation by user - Submit button is enabled	
Trigger	User clicks on record audio button	
Main Success Scenario	**Step**	**Action**
	1	User clicks on record audio button
	2	System begins to record
	3	User recites statement
	4	System stops recording
	5	System converts speech to text
	6	System displays converted speech on screen
	7	Submit button is enabled
Variations	Exception Flow	
	Step	**Action**
	1	User loses network after typing
	2	Button is disabled
	3	Audio recording is stopped and deleted
	4	Notification alerting the user that the network was lost appears

Table 4.16 – Use Case Description : Record Audio Input

Use Case	Get Sarcasm Result
ID	UC-3
Goal in Context	User submits text or audio and converted text to API to fetch model result
Actors	Primary Actor
	User
Priority	Critical
Pre-Conditions	- User must be connected to a network - Text or Audio + Text sample must be provided

Post-Conditions	- Spoken text is displayed on the screen - Sarcasm Result is shown - User is prompted for whether the probability should be shown	
Trigger	User clicks submit button	
Main Success Scenario	**Step**	**Action**
	1	User records an Audio sample or provides a text sample
	2	User submits the data via the submit button
	3	Model takes the relevant data and passes it via the relevant model (if text then text model, else combined model)
	4	Model saves weights for prediction
	5	API sends back result
	6	System displays result
	7	System prompts for user to get Sarcasm Probability
Variations	Exception Flow	
	Step	**Action**
	1	User loses network after typing
	2	Button is disabled
	3	Notification alerting the user that the network was lost appears

Table 4.17 – Use Case Description : Get Sarcasm Result

Use Case	Print Sarcasm Statistics
ID	UC-4
Description	User requests for probability and system fetches it based on previous predicted result
Actor	Primary Actor

	User	
Priority	Important	
Pre-Conditions	- User must be connected to a network - User must have requested a sarcasm result	
Post-Conditions	- User gets Sarcasm statistics on screen	
Trigger	User requests probability	
Main Success Scenario	**Step**	**Action**
	1	User records an Audio sample or provides a text sample
	2	User submits the data via the submit button
	3	Model takes the relevant data and passes it via the relevant model (if text then text model, else combined model)
	4	Model saves weights for prediction
	5	API sends back result
	6	System displays result
	7	System prompts for user to get Sarcasm Probability
	8	User requests sarcasm probability
	9	System fetches sarcasm probability from API
	10	Statistics is displayed on screen
Variations	Exception Flow	
	Step	**Action**

	1	User loses network after typing
	2	Button is disabled
	3	Notification alerting the user that the network was lost appears

Table 4.18 - Print Sarcasm Probability

4.10 Requirements

The table below contains the relevant priority levels of functional and non-functional requirements. This will aid in deciding the order of features to be completed and whether they will be highly important for the final system.

Priority	Definition
Critical	These are associated with core functionalities of the system that are required for the system to be fully functional as specified.
Important	These are important features that do not decide the functionality of the system but rather are quality of life improvements that can help with the overall efficiency and performance of the system
Preferable	These are features that can be considered as “nice to have” but do not determine either the functionality or performance of the system.
Desired	These are features that are also “nice to have” but are not considered for the current implementation and are rather considered as future enhancements that may or may not get included into the project.

Table 4.19 – Priority Levels of Requirements

4.10.1 Functional Requirements

ID	Requirement	Priority	Use case
FR01	The system should accept any grammatically correct statement in English via text input	Critical	UC-01
FR02	The system should accept any grammatically correct statement in English via Audio input	Critical	UC-02

FR03	The system should be able to correctly differentiate sarcasm in most common scenarios after a text or audio input	Critical	UC-03
FR04	The system should be able to print out any statistics provided by the model	Important	UC-04

Table 4.20 - Functional Requirements

4.10.2 Non-Functional Requirements

The non-functional requirements for the application are listed below

ID	Requirement	Specification	Priority
NFR01	System should deliver relatively accurate results	Performance	Critical
NFR02	Fetch result under 5 seconds	Performance	Critical
NFR03	Application should contain best practices for development to maintain readability and extensibility of model	Maintainability	Important
NFR04	The application should be able handle many requests if handled by a server	Scalability	Desirable
NFR05	The application should be able to process large statements	Performance	Preferable
NFR05	The application should be able to handle complex statements	Performance	Desirable

Table 4.21 - Non-Functional Requirements

4.11 Chapter Summary

In summary, the chapter identified the system stakeholders and defined their roles and behaviors in the system. The requirement elicitation techniques were discussed and compared to justify their benefits and issues to make informed decisions about which methods would provide accurate and ultimately beneficial information. Finally, an overview of the system was provided using use case diagrams and the main cases were described using the use case

descriptions ultimately providing us with a list of functional and non-functional requirements that will help us identify the features that the application will soon contain.

5 SOCIAL, LEGAL, ETHICAL AND PROFESSIONAL ISSUES

Sarcastically belongs to class 2 research based on the University of Westminster ethical research guidelines. It holds implications regarding possible identification of participants due to the voice recording feature of the application. The following section discusses the mitigations taken to avoid issues under class 2 research.

5.1 SLEP issues and Mitigation

Social	**Legal**
• During the execution of the application, no user data is saved. The audio data provided at record time is only ever used to provide an output in the model. After-which it is disposed of. • Religious, Ethnic, Political and Gender are differences that are considered purely because a topic such as sarcasm is heavily dependent on the person and therefore his/her views on the above topics relate to their tolerance of sarcasm. However, the author as not used this information in a way that exploits these users and is purely for informational and comparison in views. • Questionnaire data is never used verbatim in the report. Additionally, no information provided by the user can be tied to a specific user and therefore their identity remains safe.	• As per the GDPR regulations in place by the EU. Sarcastically does not violate any of the required regulations mentioned. • Data collection was anonymized via surveys. • Any images, data re-use or collective works have been correctly attributed. • The application and it source code shall be made open source under the Mozilla Public License 2.0 (MPL 2.0)

Only statistics are used to demonstrate points in the report.	
Ethical	**Professional**
• The survey was completely optional, and users had the ability to decline filling it from the very start via one of the questions provided. Additionally, they were allowed to provide an email and a request to delete their information towards the end. If they wished for it to be deleted. • The survey/questionnaire fully detailed the use of the data in the project and confirmed the participation of the user in the survey via a dedicated "permission for data" page in the form. • The dataset that was utilized for the project is used under the MIT public license	• The users were informed about the use of their data in the survey passed out initially. • No manipulation of the data results was done to falsify the results of the research. • The system used for development was consistently updated to receive the latest security updates.

Table 5.1 – Socail, Legal, Ethical and Professional Issues & Mitigation

5.2 Chapter Summary

This chapter covered all the issues and the mitigations taken to prevent these issues from a social, legal, ethical and professional perspective.

6 SYSTEM ARCHITECTURE

6.1 Overview

This chapter will touch on the initial system architecture and design for Sarcastically. It will run through some high-level architecture decisions as well as some design paradigms which will be shown directly after. We will also look at the initial goals of the system's design which will essentially shape the overall architecture of our system.

6.2 Design Goals

Goal	Description
Reusability	The system should be modular, such that it can be used as an extension of other application that makes use of its API
Correctness	The system at any given time should be able to give accurate prediction result. Not doing so results in the application being invalid. Even if result is incorrect, it should be able to provide reason as to why it was deemed incorrect (via percentage-based values)
Adaptability	The system should be built in a way that it can be easily extended when new features are to be built into the system.
Reliability	The system at the very least, must be able to correctly identify basic and straightforward sarcasm but in the best case be able to identify possible complex sarcastic statements.

Table 6.1 – Design Goals

6.3 System Architecture Design

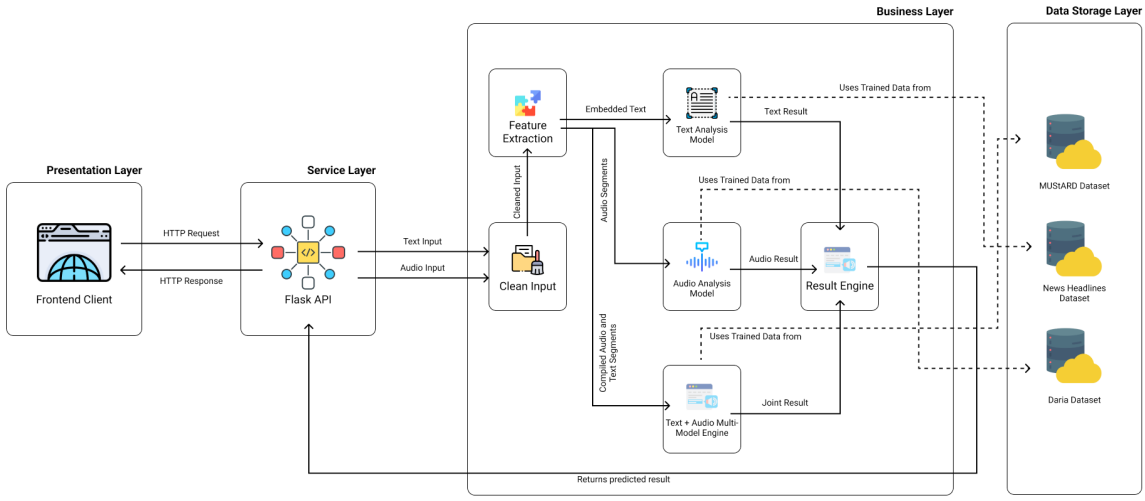


Figure 6.1 - System Architecture Design

The above diagram shows the high-level overview of the overall system of Sarcastically. The system is divided into 4 tiers namely, the Presentation Layer, Service Layer, Business Layer and Data Storage Layer

6.3.1 Presentation Layer

The Presentation Layer is client-side application that will make use of the library produced by sarcastically. This can be any JavaScript based client-side library or framework or even an interface layer such as JSInterop in Blazor Applications written in C# and ASP.NET. As long as the client can make requests to the JavaScript API for Sarcastically's lib package, the client should be able to access the pre-built models exposed by Sarcastically to its full ability.

6.3.2 Service Layer

The service layer is the library API that will expose functionality allowed by Sarcastically and will allow client/server-side JavaScript libraries to interface with it and access functionality provided by the model.

6.3.3 Business Layer

The business layer is where most of the system logic happens. This is where the data will be cleaned, features will be extracted and then fed into the relevant model for processing and predicting. The models will have been trained on predefined datasets for their relevant tasks for example the Daria dataset for Audio Analysis and the News Headlines Dataset for Text Analysis. The preceding results will be fed into a result engine that will return the prediction in a readable format back to the service layer for processing by the client.

6.3.4 Data Storage Layer

This will simply be a collection of pre-made datasets that the models will reference for processing their data and predicting results. It's the main source of truth for data which the model is trained on. It consists of the MUStARD and Daria datasets which are mainly focused on text and audio and audio respectively.

6.4 System Design

6.4.1 Choice of the Design Paradigm

For the development and design of Sarcastically we will make use of the Structured Systems Analysis and Design Method (SSADM). It is a system that follows a waterfall process from feasibility studies till the final product development stage. SSADM benefits from breaking the project into small manageable modules which have well defined success scenarios and objectives which help manage our application design a bit easier. The traditional SSADM makes for things to be done sequentially, in order so that we have better control over the steps taken to get from start to finish in our initial design. With SSADM, there are 3 main techniques which are logical data modelling, data flow modelling and entity behavior modelling. These

techniques can be incredibly time consuming however does result in an extremely bespoke solution towards the end. Each stage of the SDLC in SSADM is evaluated and agreed upon before moving towards the next stage. This is important to maintaining an error free system that focuses on the big picture. This is especially important in the situation of Sarcastically which is intended to be a research project that requires both time and effort to cover points of interest when it comes to development of the architecture of how the system will be planned out over the research year.

6.4.2 Data Flow Diagram

The Data flow diagram shows the flow of processes within a system. The following Data flow diagram shows a level 2 diagram which demonstrates a more in-depth view of the context diagram previously shown.

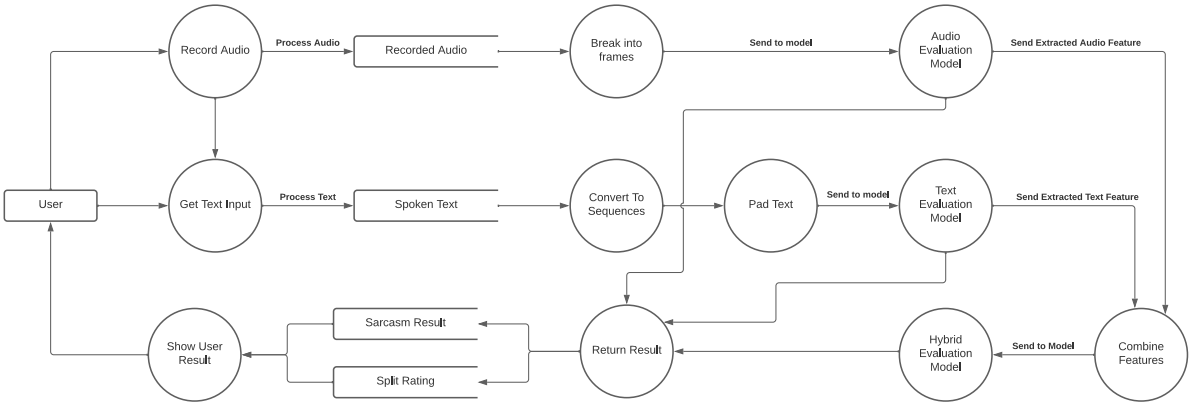


Figure 6.2 - Data Flow Diagram

6.4.3 Process Flow

The activity diagram below demonstrates the flow of Sarcastically's classification system. The swim lanes defined are the user and the system due to them being the only valid actors in the system process. They help to clearly identify point of control for each action in the system. The actions defined in the diagram closely coincide with the use case diagram previously defined.

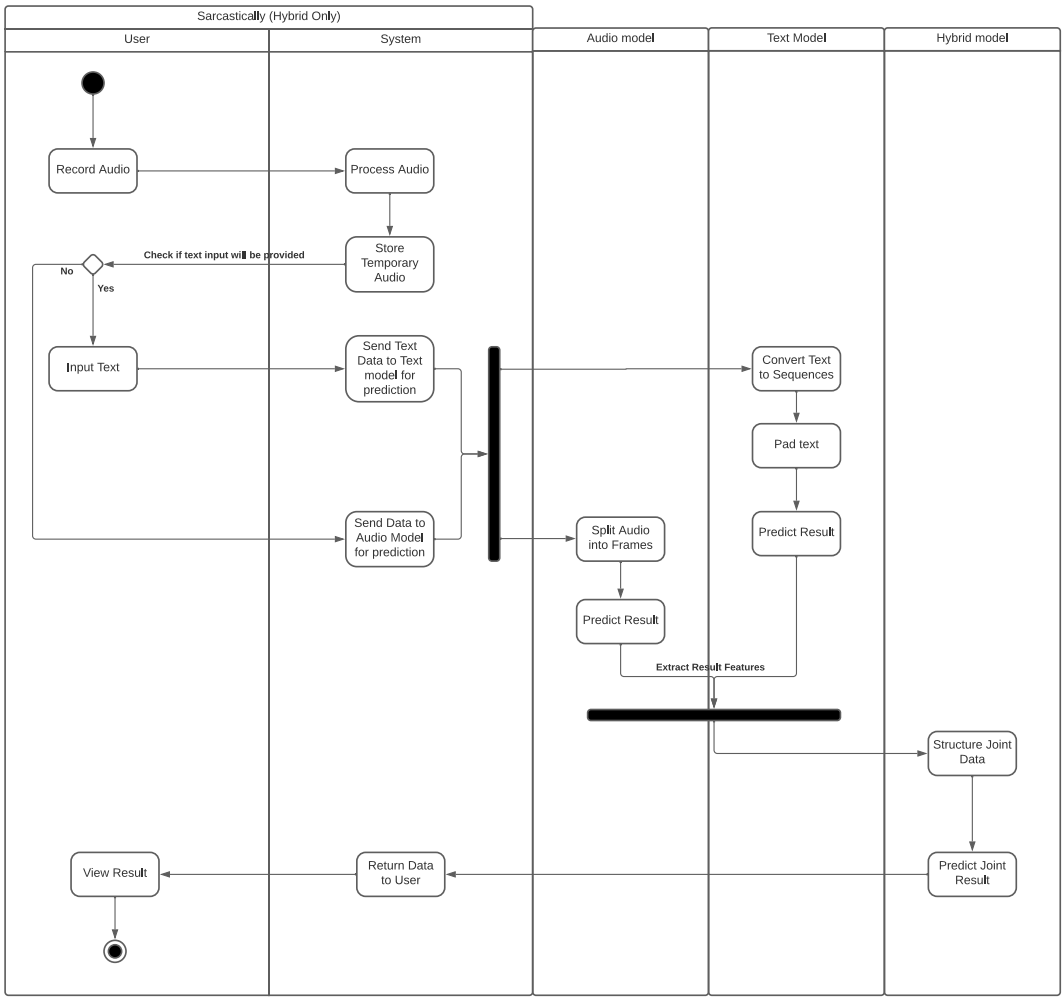


Figure 6.3 - Activity Diagram

6.5 Summary

The chapter covered the all the related design decisions to the implementation of the system covering all the high level and low-level designs and philosophies covered in the application. We also briefly discussed the design goals we hope to achieve in the application and showed examples of the relevant diagrams describing parts of the applications architecture.

7 IMPLEMENTATION

7.1 Overview

This chapter will focus on the implementation which was done to fulfill the system design. It will run through the technology selection for the system along with the libraries and tooling that were used to build and develop the application prototype. Finally, a discussion on core feature implementation details will be done and will run through the things that could've been done better and what was done to begin with.

7.2 Technology Selection

7.2.1 Technology Stack

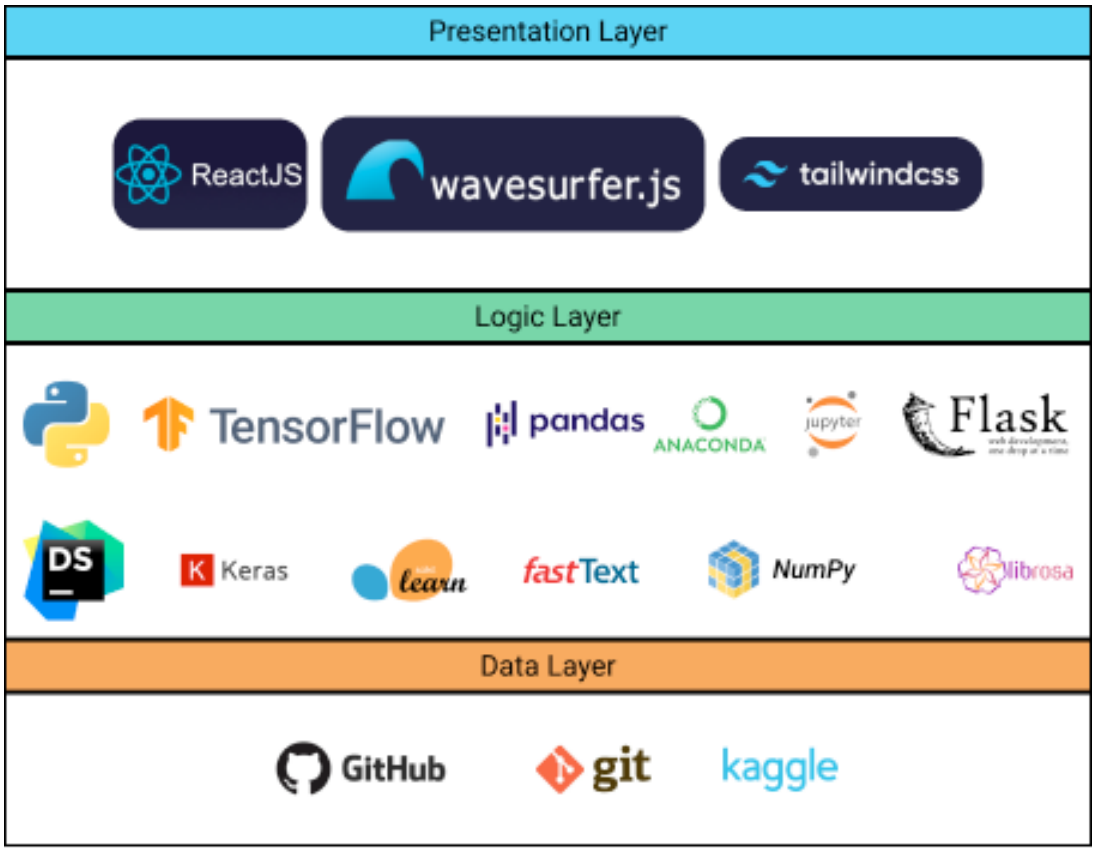


Figure 7.1 - Technology Stack

7.2.2 Data Selection

7.2.2.1 Datasets

We make use of multiple datasets due to the lack of available datasets in the specific area being investigated. We in turn use multiple datasets to train specific parts of the overall model, all related to sarcasm but in various aspects such as text, audio and joint audio and text.

In terms of the audio dataset, we are making use of the Daria dataset, which makes use of audio clips from a show recorded back in the 80's which used acted speech for sarcasm which should give us a pretty good representation of sarcastic statements. And for the joint model, we will make use of the single available dataset MUStARD (Castro et al., 2019) which joins data from famously sarcastic and comedically funny shows such as Big Bang Theory, Friends etc. The Dataset contains annotated data including their video clip timestamps indicating where the sarcastic statement occurs which will be used to train the combined model. Consequently, the text model will also be individually trained on the text portion of the MUStARD Dataset used in the joint model.

7.2.3 Programming Language

The language that will be used for the development of the model will be Python. Python has an extensive and mature system of data science-based libraries that make development for models easy. Python is a simple and scalable language that is easy to pick up and learn which makes it easier to focus on the model building. It also has an extremely large community which can be helpful for debugging and error validation.

For the frontend and making the library package, we will make use of JavaScript, which is widely used in many client-side web projects and has a wide array of libraries and frameworks which will make the UI easy to develop. It will also make building the library easier due to the large array of packages available on Node Package Repository (npm) which can assist in making a quality library for other users to use.

7.2.4 Libraries

Library	**Reason for Use**
WaveSurfer.js	Used for microphone recording animation in frontend and generating relevant soundwaves in the frontend web app
ReactJS	To build the frontend web application
TailwindCSS	Frontend utility class library for rapid prototyping
TensorFlow + TensorFlow Keras	Provides tooling and API's for carrying out complex machine learning actions with minimal effort
Tensorflow.js	Provides TensorFlow package and its relevant companion packages for building the JavaScript package needed for Sarcastically's final product
Scikit Learn	Provides low-level APIs for predictive data analysis
Librosa	Provides low level APIs for music and audio analysis used in the audio model
Pandas	Pandas is used for data analysis tasks across the project among other things it helps with

	reading the csv and json files used to train the models in Sarcastically.

Table 7.1 – Libraries utilized for implementation

7.2.5 IDE's

IDE	Reason for use
DataSpell	• Is built on top of PyCharm with specialized support for Jupyter Notebooks • Enhanced data science tool support • Direct support for hosted notebooks via Google Cloud Platform among others • Support for Data and Visualization Outputs to better view data
Visual Studio Code	• Will be used for building the JavaScript library that will make use of the model built via python (conda) • Extensive support for JavaScript/TypeScript • Wide array of extensions that will help speed up development • Support for Python and Jupyter Notebooks via Rich Extension Ecosystem just in case DataSpell does not provide certain features which are required.

Table 7.2 – IDE's used for development

7.2.6 Summary of Technology Selection

The technology selected above has been carefully evaluated as to their usefulness and ease of use to make development smooth and care-free. Each of the tooling and libraries selected above have their own usefulness inside the application and have proven to be helpful for the most part.

7.3 Implementation of core functionalities

The following section will discuss the process of implementing the core functionalities of Sarcastically. Giving a breakdown of how each model was developed and what was taken into consideration when making the feature along with its intended use. We will also discuss some of the challenges the author came across whilst attempting to build the model.

7.3.1 Re-cap of the structure of Sarcastically

Sarcastically is a multi-model approach at solving sarcasm making use of text and audio. The use of a multi-model approach means that Sarcastically is a model built on 3 individual models which are the text model which handles the pure text input, the audio model which handles only the pure audio side of the detection and the aggregator model that combines the other 2 models and aggregates their features to train itself. The models are trained individually to make sure that the process flow of getting from the text model to the aggregator is done correctly and efficiently to assure max throughput.

7.3.2 Data Preprocessing

The common dataset which the models were being trained on, MUStARD was a very small dataset at the time of writing and came in a JSON format that had unnecessary labelling in a way that it would have been difficult to use as is. So, in order to make it into a readable array of data, the data had to be flattened the data and formed into a new CSV file which is more appropriate for ML tasks such as this. The following is the script used to generate the CSV by removing one level of the JSON from the MUStARD dataset.

```
data = json.load(open("mustard_dataset.json"))
nested_data = json.dumps(pd.json_normalize(data, max_level=0).values.flatten().tolist())
nested_data_df = pd.read_json(nested_data)

nested_data_df.to_csv("normalized_mustard_dataset.csv",index=False)
```

Figure 7.2 - MUStARD dataset pre-processing script

7.3.3 The Text Model

Sarcastically's text model is the simplest part of the whole equation due to the sheer number of ways to carry out text-based classification in Sarcasm. We could've used tooling such as OpenAI-3 as a pre-built model and generated output but that would not have counted as original work. Instead, it was opted to go in for a basic custom model that generated the required output in a minimal way to prove the point of the research gap. There was not much preprocessing required for the text model since the data had already been formatted in a way that was usable in MUStARD.

The required changes that were made were initially changing the “sarcasm” state of each value in the dataset to a binary value (1 or 0) instead of the true or false in the JSON to reduce the conversion overhead when using the JSON directly in python. As mentioned, it was changed to a 1 or 0 to indicate true or false respectively which can help since the calculations will all be done using numerical results, so it seemed appropriate. This resulted in the below structure in CSV format.

```
data['sarcasm'].replace({True:1,False:0},inplace=True)
data.drop(data.columns[[1,2,3,4]], axis=1, inplace=True)
data.head()
```

	utterance	sarcasm
0	It's just a privilege to watch your min...	1
1	I don't think I'll be able to stop thin...	1
2	Since it's not bee season, you can have...	0
3	Lois Lane is falling, accelerating at a...	0
4	I'm just inferring this is a couch beca...	1

Figure 7.3 - Preprocessed Dataset Sample

After the above, we tokenized the words in the entire dataset to create a word index for our text model to run on top of. These were then padded to maintain uniformity between each vector and was then converted to sequences for training.

We further created a variable called “y-train” to make use of it in the training phase. The to_categorical method used in the second to last line of the code below converts the vectors into a binary class matrix which helps when using the categorical cross-entropy loss function during the training phase.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(utterances)
train = tokenizer.texts_to_sequences(utterances)
padded_train_sequences = pad_sequences(train, maxlen=max_length, padding=padding_type)
y_train = to_categorical(sarcasm_states, num_classes=2)

vocab_size = len(tokenizer.word_index)
```

Figure 7.4 – Snippet for pre-processing for text model data

In situations such as this where Sarcasm is the focus, word embeddings play a major part of the process since the weights given in these embeddings are usually very helpful when understanding the meaning behind words in context. Instead of generating our own, we instead make use of a pre-built embedding from FastText by Facebook Research which contains 1-

million-word vectors trained on the Wikipedia 2017, UMBC web corpus and the statmt.org news dataset which contains 16 billion pre-weighted tokens. This removes a large amount of work to create a strong embedding set.

```
words_not_found = []
nb_words = len(tokenizer.word_index)
embedding_matrix = np.zeros((nb_words + 1, 300))
for word, i in tokenizer.word_index.items():
    if i >= nb_words:
        continue
    embedding_vector = w2v_model.get(word)
    if (embedding_vector is not None) and len(embedding_vector) > 0:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
    else:
        words_not_found.append(word)
```

Figure 7.5 – Snippet for building embedding matrix for FastText pre-trained embedding

After this, all that was left was to train the CNN and consequently the LSTM using the desired layers.

```
input_data = Input(shape=(max_length,), name='main_input')
embedding_layer = Embedding(vocab_size + 1, 300, weights=[embedding_matrix], trainable=False)(input_data)
conv_1 = Conv1D(filters=50, kernel_size=4, activation='relu')(embedding_layer)
max_1 = MaxPooling1D(pool_size=2)(conv_1)
lstm_layer = Bidirectional(LSTM(128,return_sequences=True))(max_1)

flatten = Flatten()(lstm_layer)
dense = Dense(100, activation='relu', name='fully_connected')(flatten)
out = Dense(2, activation='softmax')(dense)

model = Model(inputs=[input_data], outputs=[out])
```

Figure 7.6 - Layers for Text Model

We initially make use of an Embedding layer which takes the embedding matrix of the pre-built FastText vectors as its weights rather than building our own embedding layer. After this we make use of a 1-Dimensional CNN network layer which will help build an output of tensors that we can further train using other layers such as our LSTM layer which splits the tensor into 128 neurons. It is then flattened to get a streamline set of results before passing it on to our 1st Dense layer which will use the Rectified linear Unit activation function to convert the output into a binary result to simplify the output and then pass it on to our 2nd dense layer which uses the SoftMax function to predict a distribution of probabilities that will aid in our prediction.

Finally, the model is trained using a binary cross entropy loss function for approximately 5 epochs to get a minimal model that carries out basic sarcasm detection. Binary Cross Entropy works well in this scenario since there are only 2 possible outcomes, sarcastic and non-sarcastic and there are only 2 possible classes to be considered when training out model. This also happens to be the reason we picked binary cross entropy over a categorical cross entropy since the latter is better used for multi-class data.

```
model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy'])
history = model.fit(x_train, y_train,  batch_size=64, epochs=5, verbose=1)
```

Figure 7.7 - Snippet for Compiling and Training Text model

7.3.4 Audio Model

The audio model of sarcastically required a bit more work into it due to the amount of domain knowledge required to even do the basic things in the model. Due to this, the implementation was kept relatively simple to satisfy the penultimate goal of comparison without over-engineering the solution. There aren't a lot of ready-made audio models, so this had to be built from the ground up just like the text model. The data for the audio model was yet again provided by the MUStARD dataset but required some important pre-processing beforehand.

The preprocessing step for the audio model required 2 things. First, the audio files needed to be converted from the .mp4 format to the .wav format for reasons explained at the end of **Chapter 2.5.2**. This was essential to get as much data out of the audio file as possible. We used FFMPEG which a library for audio manipulation which allowed the author to make the conversion without any major data loss. Second, a restructure of the csv used in the text model to include the audio file name in the csv to map it to its utterance and sarcasm state. Without this, we would be unable to tell the difference between a sarcastic audio clip and a non-sarcastic one during training unless we carried the audio model via an unsupervised learning approach.

The code for generating the CSV file as well as the audio conversion to .wav is shown below,

```
loadedJson = json.load(open("mustard_dataset.json"))

table = pd.DataFrame(columns=["file_name","utterance","context","sarcasm"])

for key, val in loadedJson.items():
    table = table.append({
        "file_name": f"{key}.wav",
        "utterance":val["utterance"],
        "context":val["context"],
        "sarcasm":val["sarcasm"]
    }, ignore_index=True)

table.head()
table.to_csv("normalized_mustard_dataset.csv",index=False)
```

Figure 7.8 - Code to create the updated CSV which maps the audio file name to the sarcasm state

```
converted_utterances_dir = "../mmsd_raw_data/converted_utterances"
for filename in tqdm(filenames):
    old_file = f"{utterances_dir}/{filename}"
    new_file = f"{converted_utterances_dir}/{filename[:-4]}.wav"
    os.system(f"ffmpeg -i {old_file} {new_file}")
```

Figure 7.9 - Code for converting .mp4 files into .wav files

After the pre-processing step was completed, we needed to convert all used audio clips into the representations of their Mel-Frequency Cepstral Coefficients (MFCCs). We approached this by converting each clip into their MFCC representations and then calculating the mean of each nested array (which represents the number of windows in a frame) to output a single 13-dimension vector which would be used during training. The code for this is shown, below,

```
def create_mfcc(filepath:str, n_fft:int, hop_length: int, n_mfcc:int = 13):
    """
    Creates MFCC for file at filepath

    :param filepath: Location of file to be used
    :param n_fft: Number of Fast Fourier Transforms
    :param hop_length: Number of Hops within samples
    :param n_mfcc: Number of MFCC's to be outputted
    :return: Array containing mean of all MFCC's
    """
    signal, sample_rate = librosa.load(filepath)
    mfccs = librosa.feature.mfcc(y=signal, sr=sample_rate, n_fft=n_fft, hop_length=hop_length, n_mfcc=n_mfcc)
    mean_mfccs = np.mean(mfccs.T,axis=0)
    return mean_mfccs
```

Figure 7.10 - Code for generating mean of MFCC of audio clip

This was done for each clip by the function below, which outputted a Pandas dataframe containing the MFCC and the relevant sarcasm state for each clip which was produced.

```python
def create_mfcc_features(data):
    """
    Creates a list of Mel-Frequency Co-Efficients
    :param data: Pandas Dataframe of Input Data
    """
    hop_length = 512
    n_fft = 2048

    _mfcc_df = pd.DataFrame(columns=["features", "sarcasm_state"])

    tqdm_data = tqdm(zip(data["file_name"], data["sarcasm"]))

    for file_name,sarcasm_state in tqdm_data:
        tqdm_data.set_description(f"Creating MFCC for {file_name}")
        fp = f"{audio_data_fp}/{str(file_name)}"
        mean_mfccs = create_mfcc(fp, n_fft, hop_length)
        _mfcc_df = _mfcc_df.append({
            "features": mean_mfccs,
            "sarcasm_state": sarcasm_state
        }, ignore_index=True)

    return _mfcc_df
```

Figure 7.11 - Function generating the MFCC for all audio clips

This setup is how the data required for training the model was generated. At this point we have a data frame containing the above-mentioned information which can then be passed to our ANN. This is shown below,

	features	sarcasm_state
0	[-146.32887, 99.83219, -50.631638, 9.27...	True
1	[-147.90367, 104.595116, -39.535, 4.108...	True
2	[-21.120022, 86.11223, -35.381474, 22.6...	False
3	[-23.78374, 70.78856, -34.25742, 23.382...	False
4	[1.6350226, 82.73289, -46.96253, 14.140...	True

Figure 7.12 - Outputted Dataframe containing mapped data for training

The author then made use of a Label Encoder which converted the true and false tags into a label which could then be inversed later in order to find the result once the prediction was made.

```python
# Label Encoder for getting sarcasm state
label_encoder = LabelEncoder()
y = to_categorical(label_encoder.fit_transform(y))
```

```
y.shape
```

```
(690, 2)
```

Figure 7.13 - Label Encoder for Audio Model

The author then proceeded to do the usual train-test split of the data in order to create two sets of data for training and testing and fed it off to a Neural Network which was kept fairly simple for the purpose of demonstration.

```
model = Sequential()
model.add(Dense(100, input_shape=(13,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(label_count))
model.add(Activation('softmax'))

model.summary()
```

Figure 7.14 - ANN for Audio Model

This was then compiled and fitted alongside some metrics provided to give the author some information about the training as well as when the model was evaluated.

```
model.compile(loss="categorical_crossentropy", metrics=['accuracy',f1_m,precision_m,recall_m], optimizer='adam')

num_epochs = 100
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath="../models/audio_model.h5", verbose=True, save_best_only=True)

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, callbacks=[checkpointer])
```

Figure 7.15 - Audio Model Compilation and Fitting

7.3.5 The Aggregator Model

The aggregator model has once again been kept very simple due to a lack of both time and background knowledge in related technologies. The approach used for the aggregator model was initially supposed to follow Modality Fusion such as defined by Cai et al., 2019 but will now follow a Weighted Average approach in the interest of keeping it simple and in a working state. This is a ensemble learning method rather than a multimodal approach but uses the prediction values of the 2 individual models instead of combining their features together as a single model which is usually how ensemble learning works. This may downgrade the accuracy of the project since a more advanced solution is not used and therefore hinders the result. But for the purpose of a working model, advancements have been made to make sure that the text and audio models catch up the slack of the aggregate model

7.4 Implementation of API's

The API for Sarcastically was implemented via a Flask API. The author figured it was best to streamline the process by utilizing an API written in the same language as the model itself

hence Python and therefore Flask. This helped us to utilize the same libraries and conversion methods used during the data pre-processing stage of building the model.

```
@staticmethod
def convert_audio_to_wav(file):
    command = ['ffmpeg', '-y', '-i', '-', '-f', 'wav', '-']
    process = subprocess.Popen(command, stdin=subprocess.PIPE, stdout=subprocess.PIPE)
    wav_file, error_data = process.communicate(file)
    return wav_file, error_data

@staticmethod
def prepare_audio(audio: FileStorage):
    audio, audio_sample_rate = librosa.load(audio)
    audio_features = librosa.feature.mfcc(y=audio, sr=audio_sample_rate, n_mfcc=13)
    audio_scaled_features = np.mean(audio_features.T, axis=0)
    reshaped_audio_scaled_features = audio_scaled_features.reshape(1, -1)
    return reshaped_audio_scaled_features

@staticmethod
def prepare_text(text: str):
    tokenizer = Tokenizer()
    sequences = tokenizer.texts_to_sequences(text)
    padded_sequences = pad_sequences(sequences, maxlen=60, padding='post')
    return padded_sequences

@staticmethod
def predict_audio(scaled_features):
    model = load_model(os.path.join(Sarcastically.model_folder_path, "audio_model.h5"))
    return model.predict(scaled_features)

@staticmethod
def predict_text(padded_sequences):
    model = load_model(os.path.join(Sarcastically.model_folder_path, "text_model.h5"))
    return model.predict(padded_sequences)

@staticmethod
def get_prediction_result(audio_prediction, text_prediction):
    text_model_weight = .195
    audio_model_weight = 1 - text_model_weight

    total_0 = (audio_prediction[0][0] * audio_model_weight + text_prediction[0][0] * text_model_weight)
    total_1 = (audio_prediction[0][1] * audio_model_weight + text_prediction[0][1] * text_model_weight)

    total = total_0 - total_1
    return 0 if total <= 0 else 1
```

Table 7.3 - Utility methods for pre-processing data in API

There was only one POST method implemented since the application functionality is limited to that of getting both text and audio input and processing them together. The POST method utilized libraries such as Librosa, TensorFlow and NumPy for pre-processing as in the model itself. This allowed us to have little to no difference in the way we pre-processed data in the application as well as the model when it was being built.

```
# Controllers
def post(self):
    text_prediction = []
    audio_prediction = []
    predictions = {"audio": audio_prediction, "text": text_prediction}

    if "multipart/form-data" in request.content_type:
        audio_file: FileStorage = request.files['audio_file']
        audio_file_content_type = audio_file.content_type

        if "audio/" not in audio_file_content_type:
            return {"data": "Invalid File Type Provided"}, 400

        scaled_audio_features = self.prepare_audio(audio_file)
        audio_prediction = self.predict_audio(scaled_audio_features)

        text_utterance = request.form["text_utterance"]
        padded_text_sequences = self.prepare_text(text_utterance)
        text_prediction = self.predict_text(padded_text_sequences)

        if len(audio_prediction) == 0 and len(text_prediction) == 0:
            predictions["audio"] = 0
            predictions["text"] = 0
        elif len(audio_prediction) != 0 and len(text_prediction) == 0:
            predictions["audio"] = audio_prediction
            predictions["text"] = 0
        elif len(audio_prediction) == 0 and len(text_prediction) != 0:
            predictions["audio"] = 0
            predictions["text"] = text_prediction.tolist()
        else:
            predictions["audio"] = audio_prediction.tolist()
            predictions["text"] = text_prediction.tolist()

        prediction = self.get_prediction_result(predictions['audio'], predictions['text'])
        return {"result": prediction}, 200
    return 200
```

Table 7.4 - POST method for getting prediction

7.5 User Interface

The user interface has been kept simple, displaying all required metrics along with the result and simple validation to make sure that the required data is submitted.

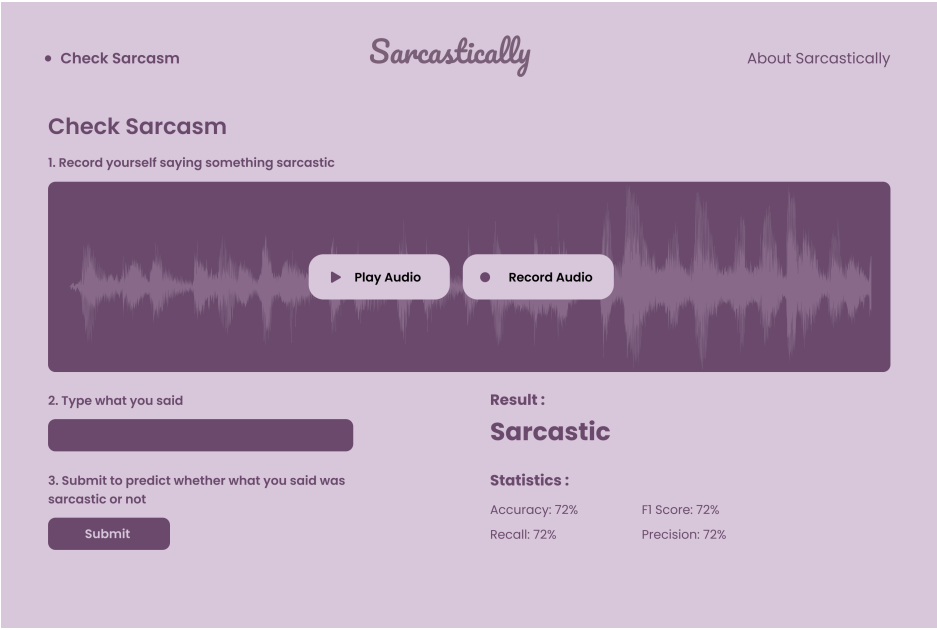


Figure 7.16 - Main UI for Sarcastically

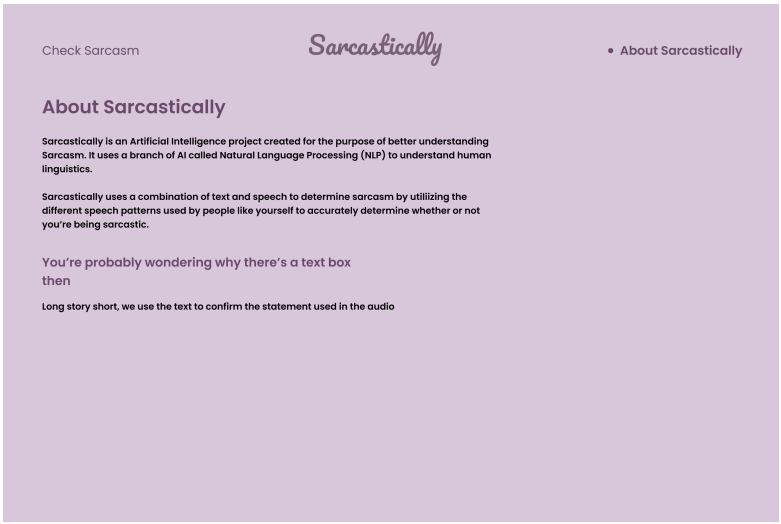


Figure 7.17 - Sample UI for About us page in application

7.6 Chapter Summary

The above chapter discussed the different tooling, libraries and technologies that were utilized for the implementation of the prototype including a summary on the choice of data used and the development framework that was to be followed for the implementation.

8 TESTING

8.1 Chapter Overview

This chapter goes over the testing phases for all functional and non-functional features of Sarcastically. Initially, the objectives and goals for testing will be elaborated on along with the testing criteria. Following this, the functional requirements will undergo Blackbox testing to make sure that the system fundamentally works. After this, the non-functional requirements will be tested alongside the limitations of the testing phase in this project.

8.2 Goals and Objectives of Testing

The ultimate objective of testing is to make sure that the application works according to the requirements specification mentioned in prior chapters. To make sure that the testing is done efficiently, the following goals are defined.

1. Validate the working nature of functional and non-functional requirements in the system
2. Validate correct code practices during development.
3. Record bugs and flaws in implementation.

8.3 Testing Criteria

For the testing criteria, the author will follow a testing specification which primarily focuses on the testing of functional and non-unfunctional features in the system. To test these features, test cases were written to validate their performance. These will ultimately outline the working state of Sarcastically.

8.4 Testing Functional Requirements

This section is responsible for testing the main functionality of Sarcastically which is required for the application to perform as intended. Testing will conduct via Black Box testing since only the expected and actual outputs are tested rather than the inner workings of the system.

ID	Description	Input	Expected Output	Actual Output	Status
FR01	The system should accept any grammatically correct statement in English via Text Input				
FT01	Verify that the application accepts English text	It's a privilege to watch your mind at work	True	True	Pass
FT02	Verify that the application does not accept empty input	-	True	True	Pass
FR02	The system should accept any kind of audio-based input				
FT01	Verify that the application accepts input	It’s a privilege to watch you work	True	True	Pass
FT02	Verify that the application can save the audio as a file	It's a privilege to watch you work	True	True	Pass
FT03	Verify that the application can convert the saved audio file to .wav format	It’s a privilege to watch you work	True	True	Pass
FR03	The system should be able to correctly differentiate sarcasm in most common scenarios after a text and audio input				

FT01	Verify that system returns the correct response when a sarcastic text input is passed	It’s a privilege to watch you work	True	True	Pass
FT02	Verify that the system returns the correct response when a. Non-sarcastic text input is passed	Since it's not bee season, you can have my epinephrine.	True	True	Pass
FT03	Verify that the system returns the correct response when a sarcastic audio-input is passed	It’s a privilege to watch you work	True	True	Pass
FT04	Verify that the system returns the correct response when a non-sarcastic audio-input is passed	Since it's not bee season, you can have my epinephrine.	True	True	Pass
FT05	Verify that the system returns the correct response when a sarcastic audio-input and text input is passed	It’s a privilege to watch you work	True	True	Pass
FT06	Verify that the system returns the correct response when a non-sarcastic audio-input and text input is passed	Since it's not bee season, you can have my epinephrine.	True	True	Pass
FR04	The system should print out any available statistics given by the by the model				
FT01	System should be able to provide the accuracy	-	True	True	Pass

FT02	System should be able to provide the f1 score	-	True	True	Pass
FT03	System should be able to provide the recall	-	True	True	Pass
FT04	System should be able to provide the precision	-	True	True	Pass

Table 8.1 – Functional Testing

8.5 Module and Integration Testing

Module	Input	Expected Output	Actual Output	Status
Web UI	Text and Audio Clip	Sarcasm result	Sarcasm result	Pass
	Text and Audio Clip	Statistics	Statistics	Pass
NLP Module	Text Model: Text Passage or Statement	Sarcasm result and statistics	Sarcasm result with statistics	Pass
	Audio Model: Audio Clip	Sarcasm result and statistics	Sarcasm result with statistics	Pass
	Aggregator Model: Text and Audio model predictions	Sarcasm result	Sarcasm result	Pass
REST API	POST: Text and Audio Clip	Sarcasm Result with Statistics	Sarcasm result with Statistics	Pass

Table 8.2 – Module and Integration Testing

8.6 Testing Non-Functional Requirements

The non-functional requirements of the system which are given in chapter 4.10.2 will be tested in the following section. Since sarcastically's performance is based on 3 separate models, the

model we will ultimately test is the final aggregate model which based on the implementation is nothing but a weighted average due to constraints in development. The statistics used to calculate the performance of the model will be manually run and generated. The author will test for the typical identifiers such as accuracy, precision, recall and the F1 score but will also evaluate all Non-functional requirements listed in the previously mentioned chapter. Sarcastically ultimately is a binary classifier predicting either true or false indicating sarcastic or non-sarcastic intent. Therefore, the performance for the model will be tested using predicted and actual classes utilizing results of true positives, false negatives, true negatives and false positives. This is the confusion matrix, and it is denoted by the table below.

	Predicted Value	
Actual Value	True Positive (Sarcastic)	False Negative (Sarcastic but detected as non-sarcastic)
	False Positive (Not Sarcastic but detected Sarcasm)	True Negative (Non-Sarcastic)

Table 8.3 – Confusion Matrix for Sarcastically

For testing statistical factors, the following equations will be used, we will make use of sklearn's metrics package which has these 4 equations built in.

Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision

$$Precision = \frac{TP}{TP + FP}$$

Recall

$$Recall = \frac{TP}{TP + FN}$$

F1 Score

$$F1\ Score = \frac{Recall * Precision}{Recall + Precision}$$

8.6.1 Accuracy Testing

The following are the results of the aggregate model on all available testing data.

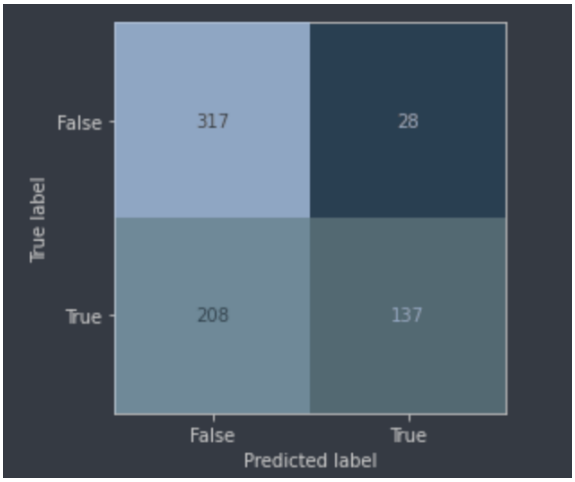


Figure 8.1 - Confusion Matrix without normalizing data

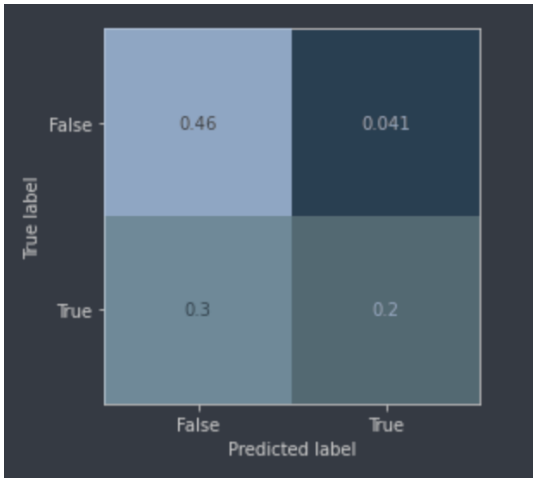


Figure 8.2 - Normalized Confusion Matrix

The above two diagrams show that around 66% of all tested values came up correctly positive or negative. However, the 34% of all the remaining data is still a large chunk of data which is arriving at a true positive of a false negative and this a concern when it comes to accuracy since it means that the model still hasn't matured enough to make an informed decision. However, this is most likely due to 2 things, the sarcasm type which is acted sarcasm which can be relatively misleading without context and the dataset size which is only around 690 entries which is extremely small making it very difficult to predict more sarcasm using such a model. When the author utilized the sklearn accuracy score metric function we came to a similar result as shown below.

```
accuracy_score(test_y_actual, test_y_pred)

 0.6579710144927536
```

Figure 8.3 - Accuracy Score using shared representation of the models predictions

The author also made use of sklearn's in-built metric functions to get the scores for recall, precision and the f1 score which is shown down below.

```
Recall: 0.4          Precision: 0.83
Accuracy: 0.66       F1 Score: 0.54
```

Figure 8.4 - Other metrics for Sarcastically

Elaborating on the metrics above, it is apparent that the recall is relatively low compared to a lot of existing models however in this case the precision would be considered relatively more

important since the model itself is measuring something as subjective as sarcasm, what the model considers to be "actual" sarcasm (denoted by the recall) is something that can be argued from person to person. For this reason, knowing the percentage of statements that the model "considers" to be sarcastic is more beneficial since we can then tune the model to improve the detection slowly but incrementally by fine tuning the model to understand the utterances it considered as false positives. This ultimately requires more data which is currently not available and whether it is successful depends on the outcome of this.

8.6.2 Performance Testing

The demo for the application was built using web technologies but was made responsive for mobile. As such the application was tested on the 4 main and most used browsers 3 of which run some form of the chromium engine namely Google Chrome, Mozilla Firefox, Microsoft Edge, Safari and Opera in no particular order. A variety of tests were run on the application via the browsers dev tools which provide extremely well detailed metrics to evaluate application performance.

8.6.2.1 Lighthouse Tests

Lighthouse is an open-source tool initially built by Google to help developers audit and improve the quality of their web applications by providing metrics against web standards which are baked into Lighthouse. For Sarcastically, this meant monitoring how performant the application was in a real-life scenario based on Performance, SEO, Accessibility and best practices. The details for these results are listed in **Appendix VI.**

In both mobile and desktop, the application got a score above 75 in both instances with the desktop version doing significantly better due to better processing ability.

8.6.3 Load Balancing and Scalability

8.6.3.1 Response Time

Checking response times off the request is one of the major indicators of application performance. This helps us identify how the server performs under various network loads. On average, the server responded on an average of 4.48 seconds per request. The following is the last recorded response in an average of 10 responses.

Figure 8.5 - Response Time Average of Sarcastically's API

8.6.3.2 Memory Usage

When testing memory usage using Chrome and Firefox's Memory Sampling tools using Heap Allocation recording, the application only makes use of a meager 700-800Kb of memory from start till end of getting a sarcasm result. This allows the application to be well performant without blocking any activities on the browser which may use up more memory and thus remains lightweight in relativity.

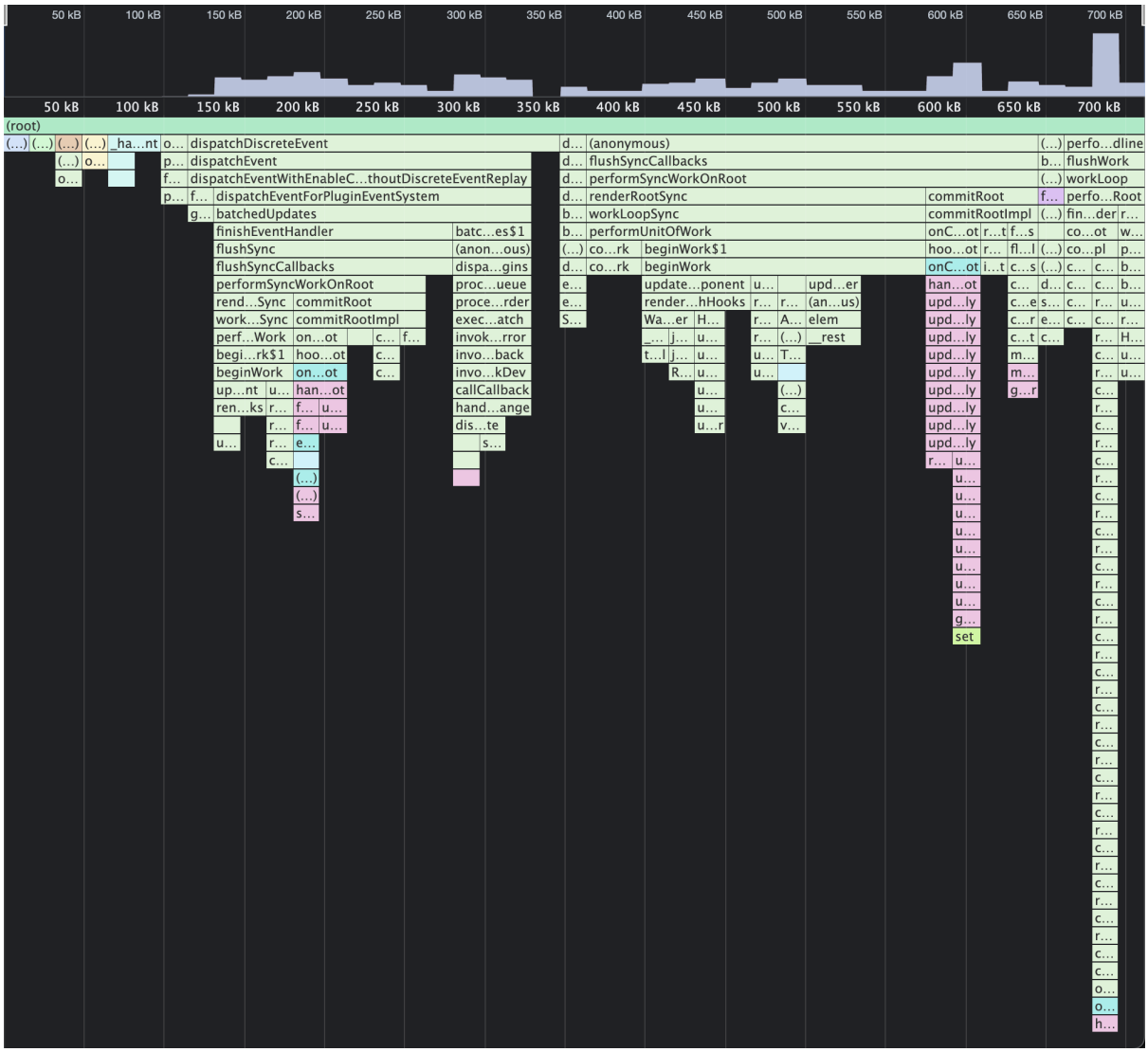


Figure 8.6 - Heap Allocation for Memory when running Sarcastically

8.6.3.3 Network Usage

Sarcastically deals with Audio files in addition to text so the natural expectation is for the application to use somewhat higher bandwidth considering the need to transport binary versions of the audio file over the network. Even with this constraint, after reading the network waterfall via Chrome DevTools when using sarcastically's frontend. We see that only around 8.1 MB in resources was downloaded with a large chunk of it being the preloaded audio files

required by WaveSurfer.js in order to render the wave form in the green block at the start of the page alongside other resources such as images and icons. In XHR requests to and from the server we see a minute 231 Bytes of data which is relatively small considering the use of audio files.

Name	Status	Type	Initiator	Size	Time	Wate
localhost	304	document	Other	299 B	16 ms	
css2?family=Pacifico&family=Poppins:wght@400;600;700&...	200	stylesheet	(index)	(disk cache)	21 ms	
bundle.js	304	script	(index)	302 B	24 ms	
axe.js	200	script	content.bundle.js:1	435 kB	312 ms	
FwZY7-Qmy14u9lezJ-6H6MmBp0u-.woff2	200	font	css2?family=Pacifico&family...	(memory cache)	0 ms	
pxiEyp8kv8JHgFVrJJfecnFHGPc.woff2	200	font	css2?family=Pacifico&family...	(memory cache)	0 ms	
pxiByp8kv8JHgFVrLEj6Z1xlFd2JQEk.woff2	200	font	css2?family=Pacifico&family...	(memory cache)	0 ms	
pxiByp8kv8JHgFVrLCz7Z1xlFd2JQEk.woff2	200	font	css2?family=Pacifico&family...	(memory cache)	0 ms	
react_devtools_backend.js	200	script	injectGlobalHook.js:2202	523 kB	80 ms	
ws	101	websocket	WebSocketClient.js:16	0 B	Pending	
default.wav	304	fetch	fetch.js:148	0 B	48 ms	
default.wav	304	fetch	fetch.js:148	365 B	12 ms	
highlighter.js	200	script	content.bundle.js:1	17.9 kB	28 ms	
default.wav	304	media	(index)	365 B	5 ms	
default.wav	304	media	(index)	365 B	4 ms	
favicon.ico	304	x-icon	Other	363 B	6 ms	
manifest.json	304	manifest	Other	363 B	7 ms	
logo192.png	304	png	Other	364 B	5 ms	
sarcastically	200	xhr	xhr.js:220	231 B	3.43 s	

Figure 8.7 - Network Waterfall of Sarcastically from Start till end of Sarcasm Detection request

8.7 Compatibility Testing

As mentioned before, sarcastically is designed to run on browsers supporting the most recent stable releases of CSS and JS technology. Therefore, the main 4 browsers namely, Google Chrome, Mozilla Firefox, Microsoft Edge, Safari and Opera were all tested and accounted for with full support for the CSS used by TailwindCSS v3. This extended to the mobile views as well.

8.8 Testing Limitations

The testing phase for sarcastically had the following limitations:

1. Testing for the API was done locally on the development machine and therefore is not subject to live server environments where performance can vary on several different levels.
2. The models were built using simple methods and therefore lack the performance of more high scaled systems available. These results can heavily alter.
3. Load balance testing was unable to be done due to the API itself not being hosted and therefore a variety of online tools which helped in load balance testing were unable to be used.

8.9 Chapter Summary

The chapter covered all issues and clarifications required due to testing in terms of accuracy, performance, load balancing, scalability and limitation. The author discussed the testing criteria and the goals for this testing phase. The author also validated the working nature of the functional and non-functional system requirements.

9 EVALUATION

9.1 Chapter Overview

The following chapter evaluates the prototype constructed for Sarcastically. It will contain evaluations on the system architecture, concept and solution by selected evaluators who have the respective knowledge in both the domain and technical aspects. The evaluation criteria will be discussed in respect to the evaluations provided by the selected evaluators. Finally, the chapter will also include a self-evaluation done by the author.

9.2 Evaluation Methodology and Approach

The evaluation of Sarcastically will utilize a combination of qualitative and quantitative approaches. The qualitative approach will be done using critical reviews by the selected evaluators since the nature of the project is subjective. Both in terms of the domain and the technical implementation of it. The feedback will be obtained via semi-formal interviews which will be adapted due to the current working conditions during the development of sarcastically. These conditions will be discussed in the following chapters. The author will make use of a thematic approach for evaluation where the author will be able to gain a better understanding of the process and objective understandings of the evaluators taking part.

9.3 Evaluation Criteria

The following themes have been identified after carrying out feedback sessions with evaluators.

Theme	Description
Concept of the Project	To provide insight into the concept of the project and the research gap it targets
Scope of the Project	To validate the scope of the project and whether it's something that is doable within a specific time
Architecture and Design	To validate the system specifications, design and architecture

Implementation	To validate the functional features of the application
Observed Limitations or possible enhancements	To identify any limitations and possible enhancements to the application given the current state

Table 9.1 – Themes for evaluation

9.4 Self-Evaluation

Theme	Self-Evaluation
Overall Concept	Many systems for sarcasm detection are already in place using various modalities. However, based on reviews on literature and survey feedback, the author was able to determine that there was a lack of research being done in the field of text and audio combined sarcasm detection and thus tackled the problem using a multi-model and ensemble learning approach.
Project Scope	The author was able to scope the research well such that it was able to mostly be completed by the end of 1 year.
System Design and Architecture	The system was designed to be an API which was able to be used on request. Overall, the application delivers as promised but is still fairly small with the scope for larger improvements
Implementation	The system is implemented well for someone who's never worked in the field before but while saying that, there is a lot of room for improvement from fixing rookie mistakes in training the model to using more powerful tooling to enhance the prediction of the model. It is believed that the dataset has heavily impacted the accuracy and therefore

	readiness of the application, but this is something that can be fixed over time. All in all, what was promised is delivered to some regard
Observed Limitations and/or future enhancements	These are discussed in the evaluation chapter.

Table 9.2 – Self Evaluation

9.5 Selection of the Evaluators

Evaluators have been selected based on various backgrounds related to Sarcastically, its domain and technical areas. The main priority in Sarcastically however will be end users since Sarcastically's output is subjective by nature and thus needs to be evaluated accordingly.

Category	Description	Scope
End Users	Sarcastically is subjective by nature and as such is heavily dependent on end user feedback since they too are subjective about sarcastic statements.	Non-Technical
Industrial Experts	Evaluations from individuals who work with various technical fields related to Sarcastically's implementation are required to assess the systems design, architecture, performance and usability.	Technical
Domain Experts	Evaluations by experts in domains such as text and audio NLP are required to evaluate Sarcastically's model and its implementation	Technical

Table 9.3 – Selection of Evaluators

9.5.1 Technical Evaluators

Following are a list of technical evaluators selected and willing to review sarcastically along with their designations and fields of study.

1. Ruchira Perera – Technical Lead at Ascentic (Pvt) Ltd.
2. Dominic Warchalowski – Systems Engineer at Consid SE

9.5.2 Domain Evaluators

Following are the list of domain evaluators selected and willing to review sarcastically along with their relative fields of study and qualifications

1. Gihan Liyanage - Senior Software Engineer in Data Science at Ascentic (Pvt) Ltd. and currently a PhD candidate in a subfield of data science **(refer Appendix IV for more)**
2. Thiloshon Nagarajah – Master's Degree in Computer Science and Artificial Intelligence and a Student Research Assistant at the University of Southern California Information Sciences Institute

9.6 Summary of Evaluation

9.6.1 Expert Opinion

The following are expert opinions on various parts of Sarcastically. These include both Domain Experts and Technical Experts

9.6.1.1 Domain Experts

The evaluations made by Domain Experts will be focused primarily on the project idea and scope, however they will also comment on other areas such as architecture, design and development processes. Their reviews will be attached under **Appendix IV**.

9.6.1.2 Technical Experts

These evaluations are from technical experts who evaluated the whole project, its structure and architecture among other things. The full reviews will be attached under **Appendix IV.**

9.6.2 Focus Group Testing

Focus group testing allows us to gather immediate experiences and expectations from a select group of people to evaluate the application. The author selects users based on aspects such as age, education and occupation to give an all-around impression on the application.

The following age ranges have been categorized as such:

Age Group	Categorization
18-30	Young Adult
30-40	Mature Adult
40-60	Aging Adult
60+	Senior Citizen

Table 9.4 – Age Groupings for Focus Group

The following participants were selected:

Name	Age Category (as per table 9.4)	Highest level of Education	Occupation Field
Dilshan Manathunge	Young Adult	Bachelor's Degree	Software Engineering

Thilanka Ishara	Mature Adult	Bachelor's Degree	Software Engineering
Tania Deckker	Aging Adult	Bachelor's Degree	House-wife
Jackie Bocks	Mature Adult	Master's Degree	Business Administration

Table 9.5 – Focus Group Participants

The focus group tests were done on an online call using google meets where the screen was shared so the participants could see everything that was being done on the authors side. The following process was followed

1. Send 3 random audio clips to participant
2. Ask participant to repeat what they heard in the audio clip and enter it manually in the text field on Sarcastically's web interface
3. Ask and record the participants response when asked whether they thought it was sarcastic
4. Submit the relevant audio file and text to sarcastically and submit for response
5. Record Sarcastically's prediction and observe whether it matched with the participants
6. Ask Participant to log all recorded events in the google form provided

Appendix VII contains all the requested questions in the survey and **Appendix VIII** contains all the user responses.

In most cases, the model got the correct result, however with the older generation of individuals especially those in the aging adult range, the results tended to be quite different due to their lack of comprehension in the subject area. This is reflected in the answer percentage.

9.7 Evaluation on Functional Requirements

Functional Requirement	Priority	Status
The system should accept any grammatically correct statement in English via Text Input	Critical	Complete
The system should accept any kind of audio-based input	Critical	Complete
The system should be able to correctly differentiate sarcasm in most common scenarios after a text and audio input	Critical	Complete

The system should print out any available statistics given by the by the model	Important	Complete

Table 9.6 – Functional Requirements Evaluation

9.8 Evaluation on Non-Functional Requirements

Non-Functional Requirement	Priority	Status
Fetch result under 5 seconds	Critical	Complete
Application should contain best practices for development to maintain readability and extensibility of model	Important	Complete
The application should be able handle many requests if handled by a server	Desirable	Untested
The application should be able to process large statements	Preferable	Untested
The application should be able to handle complex statements	Desirable	Incomplete

Table 9.7 – Non Functional Requirement Evaluation

9.9 Chapter Summary

The chapter covered a thematic analysis on sarcastically using evaluations by both the author and experienced individuals from Sarcastically's technical and problem domains. The chapter also uncovered the requirement evaluations and evaluation methodologies earlier in the chapter.

10 CONCLUSION

10.1 Chapter Overview

This chapter covers the conclusion of Sarcastically by discussing how well the aims and objectives of the project were achieved and then attempting to see how well the author used the skills obtained from the course. Additionally, it also goes through any problems and challenges faced throughout the project lifetime alongside any deviations made from the initial project proposal. Finally, it covers some of the research limitations and some concluding remarks by the author

10.2 Achievements of Research Aims & Objectives (Based on chapter 1)

10.2.1 Aim

This project aims to design, develop and test a machine learning model to improve sarcasm detection by utilizing better audio models made with natural speech over text-to-speech.

The project aim was achieved by utilizing a multi-model approach to sarcasm detection which utilized audio and text passages to better improve the prediction rate of sarcasm. The project was delivered via an API built to provide the model results. The system has also been heavily evaluated by end-users, technical and domain knowledge holders as well which further confirm its capability.

10.3 Use of Existing Skills

Knowledge gained from the internship year of the course helped drastically in being able to design such a large-scale system. The Object-Oriented Programming Module along with the Programming Principals Modules helped in better structuring the code for the project. Finally, the Enterprise Application Development module massively helped in both organization and planning in building such a complicated application structure.

10.4 Problems and Challenges Faced

1. **Lack of experience in Data Science and the domains involved in the project**
   The skills required for the domains of NLP, Audio and Text processing at the time of the proposal were relatively under-developed and required large amounts of time and effort to understand and combine into a doable project. Eventually the author had to cut corners on complexity to get most the project done in time. This was especially true when it came to the Audio side of Sarcastically. The advice of multiple experienced individuals in the NLP field alongside those who understood Multi-Model approaches was considered in order to better understand the domains and how to approach the relevant problems
2. **Lack of sufficient training data**
   While the author was able to find a dataset which was able to be used in the development of Sarcastically, the dataset contained no more than 690 text and audio clips which could be used. In a realistic scenario, this is nowhere close enough to train a highly accurate model. However, for the case of a prototype, this was sufficient to even get a working result.
3. **Effects of Covid-19**
   During the development of sarcastically, the author underwent serious medical treatment because of contracting the omricon variant of Covid-19. This resulted in a loss of around 2.5 weeks of time due to recovery. The author did work during this time

but only momentarily due to fatigue. Eventually after recovery, work resumed as usual but there wasn't much the author could do to recover that time.

4. **Effects of political and financial issues in Sri Lanka**

   Due to reasons out of the authors control, the authors country underwent a series of delayed and extended power cuts preventing the author from continuing work on the project. These power cuts extended as long as 6 hours a day with a minimum of around 3 hours a day. Due to insufficient power backups and a lack of alternatives thereafter, the author lost precious time for implementation due to this. The author made use of UPS systems but that only extended the power for up to a half hour more than what was previously possible.

10.5 Deviations – Any deviations from the original plan should be mentioned and justified

1. **Shift from releasing as library to an API**

   The initial plan was to release Sarcastically as a reusable library in JS via the node package manager however, due to unforeseen complications regarding the aggregator model, the author decided to pull back on the idea of using a library and instead shift it to a python-based API written using the Flask Library. This kept the application simple and allowed the author to better focus on the final implementation of Sarcastically itself. This in no way, affects the model itself but can bring about the potential for a better performing model if the time is utilized well.

2. **Sticking to original dataset instead of training on a larger one**

   The model was trained on the MUStARD dataset which contains a list of text and audio clips containing acted sarcasm. In a situation where there's was ample time to refine the model further in order to understand types of sarcasm out away from the "acted" sarcasm type, this would've been something to be considered. But as mentioned in 10.6.1, the unforeseen complications in the aggregate model made this difficult. In order to train it for multiple types of sarcasm we would need to make sure that there was a large enough dataset to properly train the model, as of right now, only MUStARD meets the requirements of an audio and text dataset. The others need to be heavily pre–processed in order to make sure that the data matches the already trained data.

10.6 Limitations of the Research

1. **Scope**

Due to the limited time allocated for the project, the scope had to be reduced into something that was achievable within a year. The project was ultimately completed just not in the desired way. However, it still produces a working competency of results and has still achieved the initial goal.

2. **Dataset and Model Accuracy**

   The model and its accuracy were heavily limited by the dataset being used. Unfortunately, while a dataset was able to be found for such a research topic, the extent to how well it was able to be trained was limited due to the size of the dataset (690 entries in both text and the audio). Additionally, the audio model was trained on the same dataset and as such the model is also heavily linked to the dataset and as such has no variation between the types of sarcasm that the models predict. Which is focused on Irony.

3. **Implementation Limitations**

   The implementation only accepts plain text input and cannot accept characters such as emoticons which could help supplement the models understanding of sarcasm in such situations. Additionally, the audio model isn't as in depth as it should be resulting in inaccuracies with specific tonal deviations in the audio.

4. **Limitation in Aggregate Model**

   The aggregator model is a fundamental part of the research project and has been significantly underdeveloped due to a lack of domain knowledge in Ensemble learning and time limitations. Therefore, it only uses a weighted average rather than

10.7 Future Enhancements

1. Increase size of dataset

Whether it be through crowdsourcing or some other method of collection. It is detrimental that the size of the dataset be increased as much as humanly possible to get a much higher accuracy level than currently achievable.

2. Capability of Audio Model

Right now, the audio model makes use of clips taken from tv shows which are usually free from a lot of background noise and are extremely contextualized to acted sarcasm. This is not realistic, and the audio model would be better tested with realistic clips of sarcasm in real life situations. This kind of data will need to be crowdsourced with permission from the participants.

3. Capability of the text model

Currently, the text model is only capable of understanding plain English text and nothing more advanced than this. It would be beneficial for its development if it were able to understand more advanced text features such as emoticons. Additionally, as mentioned above in the capability of the audio model, the data is trained on acted sarcasm and is not representative of real-life sarcasm where the distinction is not as clear and therefore needs this if it is to be improved further

4. Domain

Right now, Sarcastically focuses on acted sarcasm which is but a subset of a huge range of sarcasm types. To fully bring out its potential, it would be beneficial to include data from all these different types of sarcasm to build variety in detection.

10.8 Concluding Remarks

Sarcastically provides a solution to detect sarcasm using text and audio, a fairly unresearched field using NLP at its core alongside some minor audio engineering. The research fairly evaluates the existing solutions regarding the project and critically analyzes the techniques, architectures and development approaches that they use. Even in testing, end user feedback was taken in raw form and was strictly evaluated to comment on possible future enhancements to the project. The focus groups and evaluators were also considered heavily to represent a wide variety of individuals from different backgrounds and experience levels on both the technical and domain areas. The prototype itself is basic and lacks any additional features due to the complexity of the solution involved so it delivers on the main goal which is to see if it is possible or not to gain somewhat of an accurate prediction by combining the two features. The author hopes that ultimately, Sarcastically's research will turn out to be useful to those who participate in relevant field research and is helpful in their search for a solution to the problem Sarcastically attempts to solve.

Bibliography

11 Types of Sarcasm [WWW Document], n.d.. Simplicable. URL https://simplicable.com/new/sarcasm (accessed 2.13.22).

Bagate, R., Ramadass, S., 2020. Different Approaches in Sarcasm Detection: A Survey. pp. 425–433. https://doi.org/10.1007/978-3-030-34080-3_48

Bharti, S.K., Naidu, R., Babu, K.S., 2017. Hyperbolic Feature-based Sarcasm Detection in Tweets: A Machine Learning Approach, in: 2017 14th IEEE India Council International Conference (INDICON). Presented at the 2017 14th IEEE India Council International Conference (INDICON), pp. 1–6. https://doi.org/10.1109/INDICON.2017.8487712

Cai, Y., Cai, H., Wan, X., 2019. Multi-Modal Sarcasm Detection in Twitter with Hierarchical Fusion Model, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Presented at the ACL 2019, Association for Computational Linguistics, Florence, Italy, pp. 2506–2515. https://doi.org/10.18653/v1/P19-1239

Capelli, C.A., Nakagawa, N., Madden, C.M., 1990. How Children Understand Sarcasm: The Role of Context and Intonation. Child Development 61, 1824. https://doi.org/10.2307/1130840 Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R., Poria, S., 2019. Towards Multimodal Sarcasm Detection (An _Obviously_ Perfect Paper). pp. 4619–4629. https://doi.org/10.18653/v1/P19-1455

Chapter 4 – Controller and processor - General Data Protection Regulation (GDPR) [WWW Document], n.d. URL https://gdpr-info.eu/chapter-4/ (accessed 4.22.22).

Colston, H.L., 1997. Salting a wound or sugaring a pill: The pragmatic functions of ironic criticism. Discourse Processes 23, 25–45. https://doi.org/10.1080/01638539709544980

Difference between Structured Programming and Object Oriented Programming, 2021. . GeeksforGeeks. URL https://www.geeksforgeeks.org/difference-between-structured-programming-and-object-oriented-programming/ (accessed 2.27.22).

Gan, L., n.d. Improved Sentiment Classification by Multi-model Fusion 92.

Gibbs, R.W., 2000. Irony in Talk Among Friends. Metaphor and Symbol 15, 5–27. https://doi.org/10.1080/10926488.2000.9678862

González-Ibáñez, R., Muresan, S., Wacholder, N., 2011. Identifying Sarcasm in Twitter: A Closer Look, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Presented at the ACL-HLT 2011, Association for Computational Linguistics, Portland, Oregon, USA, pp. 581–586.

GuhaThakurta, S., Chetty, P., 2015. Understanding research philosophy. Understanding Research Philosophy. URL https://www.projectguru.in/research-philosophy/ (accessed 10.18.21).

Johannesson, P., Perjons, E., 2014. Research Strategies and Methods, in: An Introduction to Design Science. Springer International Publishing, Cham, pp. 39–73. https://doi.org/10.1007/978-3-319-10632-8_3

Joshi, A., Bhattacharyya, P., Carman, M.J., 2016. Automatic Sarcasm Detection: A Survey. arXiv:1602.03426 [cs].

Kiros, R., Salakhutdinov, R., Zemel, R., n.d. Multimodal Neural Language Models 9.

Kunneman, F., Liebrecht, C., van Mulken, M., van den Bosch, A., 2015. Signaling sarcasm: From hyperbole to hashtag. Information Processing & Management 51, 500–509. https://doi.org/10.1016/j.ipm.2014.07.006

Lehnert, W.G., Ringle, M.H., 2014. Strategies for Natural Language Processing. Psychology Press.

Mehta, P., 2020. Multimodal Deep Learning [WWW Document]. Medium. URL https://towardsdatascience.com/multimodal-deep-learning-ce7d1d994f4 (accessed 10.26.21).

Morency, L.-P., Mihalcea, R., Doshi, P., n.d. Towards multimodal sentiment analysis: harvesting opinions from the web 8.

Object Oriented Analysis & Design Tutorial [WWW Document], n.d. URL https://www.tutorialspoint.com/object_oriented_analysis_design/index.htm (accessed 4.21.22).

OOAD - Object Oriented Analysis [WWW Document], n.d. URL https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_analysis.htm (accessed 2.18.22).

OOAD - Object Oriented Design [WWW Document], n.d. . OOAD - Object Oriented Design. URL https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_design.htm (accessed 11.1.21).

Rakov, R., Rosenberg, A., n.d. ``Sure, I Did the Right Thing'': A System for Sarcasm Detection in Speech 5.

Schumacher, M., n.d. The use of SSADM (Structured Systems Analysis and Design Methodology) as a standard methodology on Information Systems Projects.

University of Westminster Code of Practice Governing the Ethical Conduct of Research 2020-21, n.d. 48.

Wang, H., Meghawat, A., Morency, L.-P., Xing, E.P., 2017. Select-Additive Learning: Improving Generalization in Multimodal Sentiment Analysis. arXiv:1609.05244 [cs].

Wells, C.J., n.d. Development Methodologies [WWW Document]. URL https://www.technologyuk.net/computing/software-development/systems-analysis/methodologies.shtml (accessed 2.18.22).

Zadeh, A., Chen, M., Poria, S., Cambria, E., Morency, L.-P., 2017. Tensor Fusion Network for Multimodal Sentiment Analysis. arXiv:1707.07250 [cs].

Appendix 1 – References

Baltrušaitis, T., Ahuja, C., Morency, L.-P., 2019. Multimodal Machine Learning: A Survey and Taxonomy. IEEE Transactions on Pattern Analysis and Machine Intelligence 41, 423–443. https://doi.org/10.1109/TPAMI.2018.2798607

Banse, R., Scherer, K.R., 1996. Acoustic profiles in vocal emotion expression. Journal of Personality and Social Psychology 70, 614–636. https://doi.org/10.1037/0022-3514.70.3.614

Cheang, H.S., Pell, M.D., 2008. The sound of sarcasm. Speech Communication 50, 366–381. https://doi.org/10.1016/j.specom.2007.11.003

Kreuz, R.J., Caucci, G.M., 2007. Lexical influences on the perception of sarcasm, in: Proceedings of the Workshop on Computational Approaches to Figurative Language, FigLanguages '07. Association for Computational Linguistics, USA, pp. 1–4.

Ren, Y., Ji, D., Ren, H., 2018. Context-augmented convolutional neural networks for twitter sarcasm detection. Neurocomputing 308, 1–7. https://doi.org/10.1016/j.neucom.2018.03.047

Suryawanshi, S., Chakravarthi, B.R., Arcan, M., Buitelaar, P., 2020. Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text, in: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying. European Language Resources Association (ELRA), Marseille, France, pp. 32–41.

Bamman, D., Smith, N., 2015. Contextualized Sarcasm Detection on Twitter. Proceedings of the International AAAI Conference on Web and Social Media 9, 574–577.

Cai, Y., Cai, H., Wan, X., 2019. Multi-Modal Sarcasm Detection in Twitter with Hierarchical Fusion Model, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Presented at the Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, pp. 2506–2515. https://doi.org/10.18653/v1/P19-1239

Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R., Poria, S., 2019. Towards Multimodal Sarcasm Detection (An _Obviously_ Perfect Paper), in: Proceedings of

the 57th Annual Meeting of the Association for Computational Linguistics. Presented at the ACL 2019, Association for Computational Linguistics, Florence, Italy, pp. 4619–4629. https://doi.org/10.18653/v1/P19-1455

Dave, A.D., Desai, N.P., 2016. A comprehensive study of classification techniques for sarcasm detection on textual data, in: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). Presented at the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 1985–1991. https://doi.org/10.1109/ICEEOT.2016.7755036

Davidov, D., Tsur, O., Rappoport, A., 2010. Semi-supervised recognition of sarcastic sentences in Twitter and Amazon 10.

Ding, C., Tao, D., 2015. Robust Face Recognition via Multimodal Deep Face Representation. IEEE Trans. Multimedia 17, 2049–2058. https://doi.org/10.1109/TMM.2015.2477042
Elsevier Enhanced Reader [WWW Document], n.d. https://doi.org/10.1016/j.procs.2018.05.069

Eremyan, R., n.d. Four Pitfalls of Sentiment Analysis Accuracy [WWW Document]. Toptal Engineering Blog. URL https://www.toptal.com/deep-learning/4-sentiment-analysis-accuracy-traps (accessed 4.25.22).

Filik, R., Țurcan, A., Thompson, D., Harvey, N., Davies, H., Turner, A., 2016. Sarcasm and emoticons: Comprehension and emotional impact. Quarterly Journal of Experimental Psychology 69, 2130–2146. https://doi.org/10.1080/17470218.2015.1106566

Ghosh, A., Veale, T., 2016. Fracking Sarcasm using Neural Network, in: Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. Association for Computational Linguistics, San Diego, California, pp. 161–169. https://doi.org/10.18653/v1/W16-0425

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning, in: Deep Learning. MIT Press.
Gu, Y., Yang, K., Fu, S., Chen, S., Li, X., Marsic, I., 2018. Hybrid Attention based Multimodal Network for Spoken Language Classification, in: Proceedings of the 27th International

Conference on Computational Linguistics. Presented at the COLING 2018, Association for Computational Linguistics, Santa Fe, New Mexico, USA, pp. 2379–2390.

Hazarika, D., Poria, S., Gorantla, S., Cambria, E., Zimmermann, R., Mihalcea, R., 2018. CASCADE: Contextual Sarcasm Detection in Online Discussion Forums. arXiv:1805.06413 [cs].

Hotelling, H., 1936. Relations Between Two Sets of Variates. Biometrika 28, 321–377. https://doi.org/10.2307/2333955

Indolia, S., Goswami, A.K., Mishra, S.P., Asopa, P., 2018. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. Procedia Computer Science, International Conference on Computational Intelligence and Data Science 132, 679–688. https://doi.org/10.1016/j.procs.2018.05.069

John, G.H., Langley, P., 2013. Estimating Continuous Distributions in Bayesian Classifiers. arXiv:1302.4964 [cs, stat].

Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K., 2001. Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation 13, 637–649. https://doi.org/10.1162/089976601300014493

Khaleghi, B., Khamis, A., Karray, F.O., Razavi, S.N., 2013. Multisensor data fusion: A review of the state-of-the-art. Information Fusion 14, 28–44. https://doi.org/10.1016/j.inffus.2011.08.001

Lahat, D., Adali, T., Jutten, C., 2015. Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects. Proceedings of the IEEE 103, 1449–1477. https://doi.org/10.1109/JPROC.2015.2460697

le Cessie, S., van Houwelingen, J.C., 1992. Ridge Estimators in Logistic Regression. Journal of the Royal Statistical Society: Series C (Applied Statistics) 41, 191–201. https://doi.org/10.2307/2347628

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y., 2011. Multimodal Deep Learning 8.

Patro, J., Bansal, S., Mukherjee, A., 2019. A deep-learning framework to detect sarcasm targets, 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). Presented at the EMNLP-IJCNLP 2019, Association for Computational Linguistics, Hong Kong, China, pp. 6336–6342. https://doi.org/10.18653/v1/D19-1663

Poria, S., Cambria, E., Hazarika, D., Vij, P., 2016. A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks 12.

Ramachandram, D., Taylor, G.W., 2017. Deep Multimodal Learning: A Survey on Recent Advances and Trends. IEEE Signal Processing Magazine 34, 96–108. https://doi.org/10.1109/MSP.2017.2738401

Simonyan, K., Zisserman, A., 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. arXiv:1406.2199 [cs].

Spacey, J., 2018. 11 Types of Sarcasm [WWW Document]. Simplicable. URL https://simplicable.com/new/sarcasm (accessed 4.23.22).

Subramanian, J., Sridharan, V., Shu, K., Liu, H., 2019. Exploiting Emojis for Sarcasm Detection, in: Thomson, R., Bisgin, H., Dancy, C., Hyder, A. (Eds.), Social, Cultural, and Behavioral Modeling, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 70–80. https://doi.org/10.1007/978-3-030-21741-9_8

Suhaimin, M.S.M., Hijazi, M.H.A., Alfred, R., Coenen, F., 2017. Natural language processing based features for sarcasm detection: An investigation using bilingual social media texts, in: 2017 8th International Conference on Information Technology (ICIT). Presented at the 2017 8th International Conference on Information Technology (ICIT), pp. 703–709. https://doi.org/10.1109/ICITECH.2017.8079931

Sykora, M., Elayan, S., Jackson, T.W., 2020. A qualitative analysis of sarcasm, irony and related #hashtags on Twitter. Big Data & Society 7, 2053951720972735. https://doi.org/10.1177/2053951720972735

Thelwall, M., 2018. Big Data and Social Web Research Methods 142.

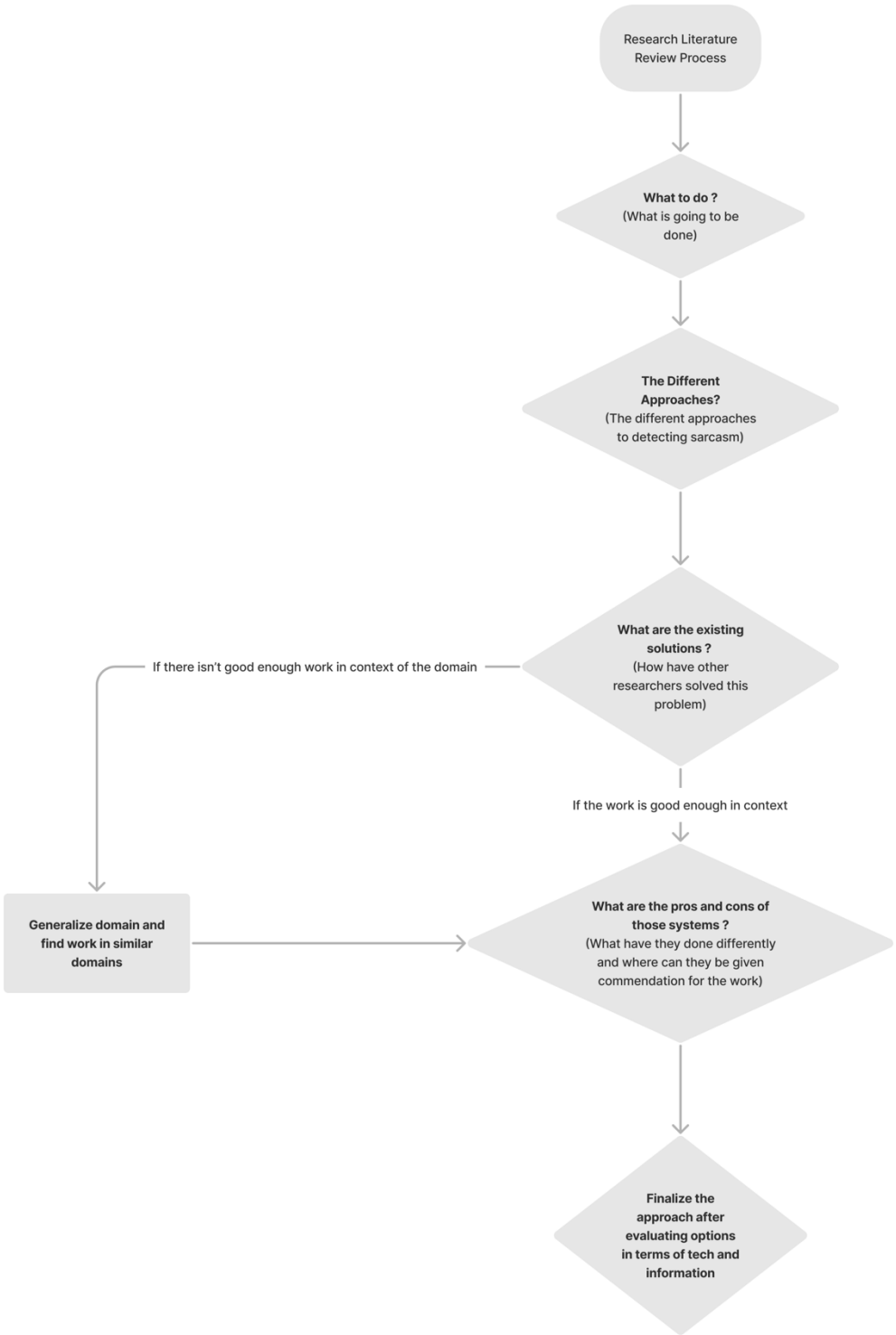
Tibebu, H., 2020. Introduction to Data Fusion. Introduction to Data Fusion. URL https://medium.com/haileleol-tibebu/data-fusion-78e68e65b2d1 (accessed 4.28.22).

Wang, X., Sun, X., Yang, T., Wang, H., 2020. Building a Bridge: A Method for Image-Text Sarcasm Detection Without Pretraining on Image-Text Data, in: Proceedings of the First International Workshop on Natural Language Processing Beyond Text. Presented at the EMNLP-nlpbt 2020, Association for Computational Linguistics, Online, pp. 19–29. https://doi.org/10.18653/v1/2020.nlpbt-1.3

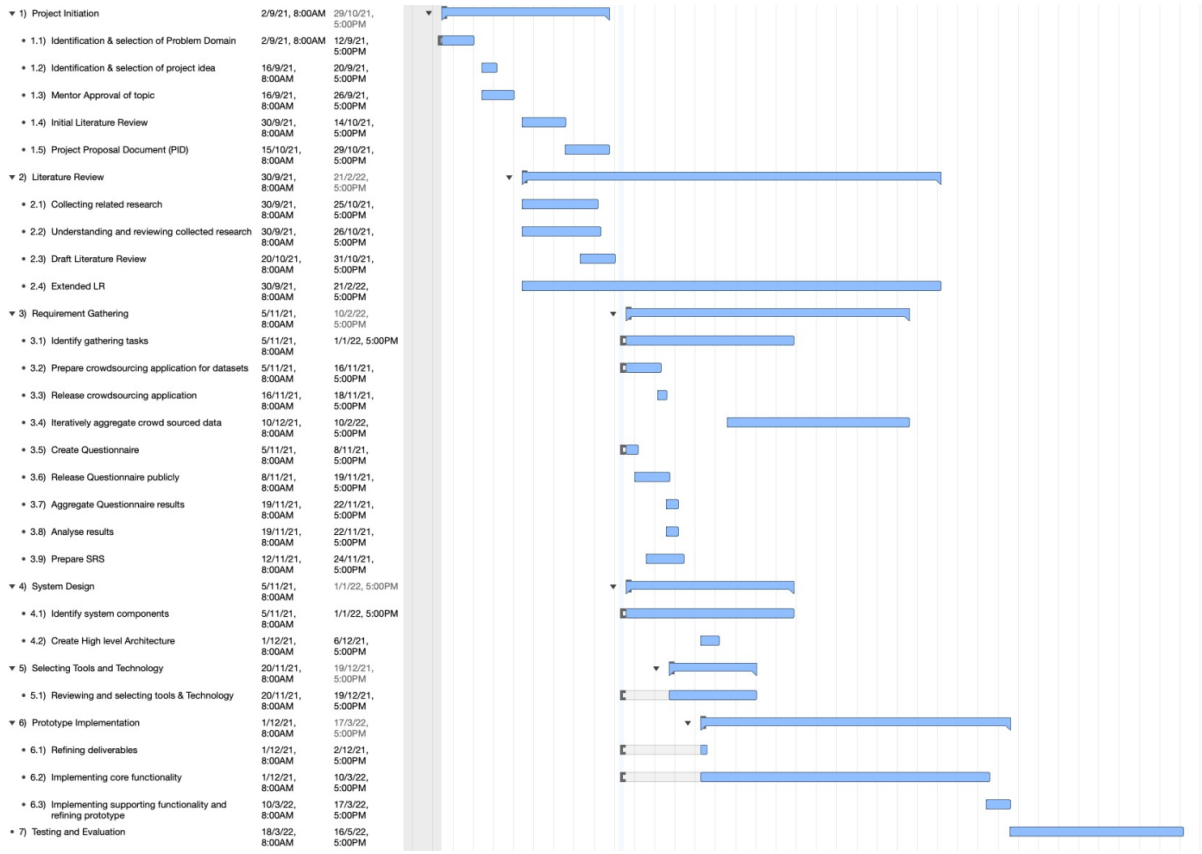
Wu, D., Pigou, L., Kindermans, P.-J., Le, N.D.-H., Shao, L., Dambre, J., Odobez, J.-M., 2016. Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 1583–1597. https://doi.org/10.1109/TPAMI.2016.2537340

Yi, D., Lei, Z., Li, S.Z., 2015. Shared representation learning for heterogenous face recognition, in: 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG). Presented at the 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), pp. 1–7. https://doi.org/10.1109/FG.2015.7163093

Appendix II – Literature Review Structure



Appendix III – Gantt Chart



Appendix IV – Expert Reviews

Reviewer 1	
Reviewer	Gihan Liyanage
Qualifications & Background	- BSc. (Special) in Statistics and Computer Science, University of Colombo - Doctoral Student for PhD in Geometric deep Learning in Natural Language Processing - Lead Data Scientist at Ascentic (Pvt) Ltd.

An interesting problem and well formulated research question and objectives. The background study has been done properly and has identified the research gap correctly. The decisions to incorporate both audio data and text data, utilizing multi-model architecture are very effective since the tone of the voice and other audio features can definitely do a huge contribution in detecting sarcasm.

When it comes to technical aspects of the implementation, number of models have been identified through thorough literature review and a successful attempt has been made to implement them. The tools, libraries and methodologies have been chosen wisely and best practices can also be seen in the work which is impressive. The performance of the implemented models is quite satisfactory with the facts that the algorithms are trained by himself and the sample sizes.

I feel that the performance can be further improved by increasing the sample sizes (especially in the audio dataset) and utilizing some other feature types. Model optimization was also done to a certain extend which can be further improved. For the aggregator model, a weighted average mechanism has been used and I would suggest looking into other algorithms available under ensemble learning as further improvements. But it's quite understandable that the amount of work he has done with the given timespan is beyond expectations compared to the scope of the overall project and the timeline.

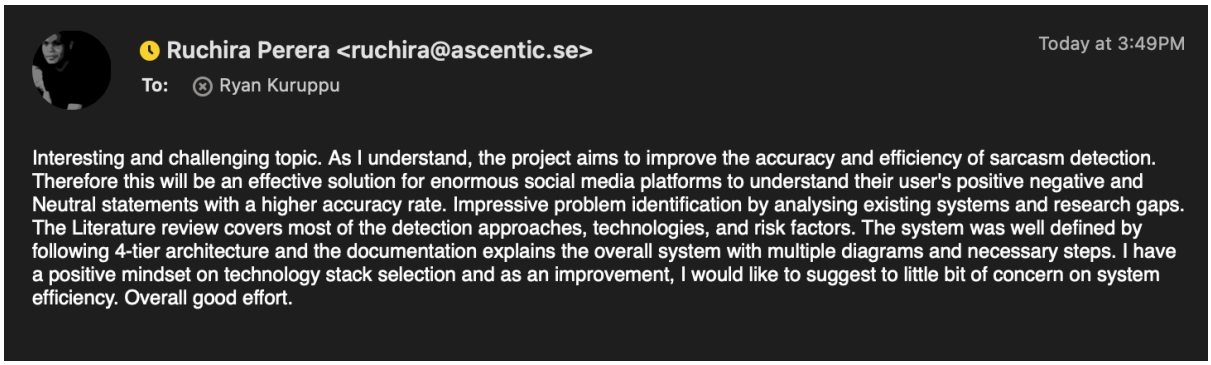
In general, impressive work which is hard expect from an undergraduate. The work has been done compared to the duration of the project is beyond expectations and the depth of the research is also superb. The work clearly demonstrates his capability to conduct R & D work. I would like to see the research is taken to the next level as a well-developed data product. All the very best!

Reviewer 2	
Reviewer	Ruchira Perera
Qualifications & Background	- MSc (Hons) Computer Science from Staffordshire University - Optimizely Content Cloud Developer (Gold License)

Interesting and challenging topic. As I understand, the project aims to improve the accuracy and efficiency of sarcasm detection. Therefore, this will be an effective solution for enormous social media platforms to understand their user's positive negative and Neutral statements with a higher accuracy rate. Impressive problem identification by analyzing existing systems and research gaps. The Literature review covers most of the detection approaches, technologies, and risk factors. The system was well defined by following 4-tier architecture and the documentation explains the overall system with multiple diagrams and necessary steps. I have a positive mindset on technology stack selection and as an improvement, I would like to suggest to little bit of concern on system efficiency. Overall good effort.

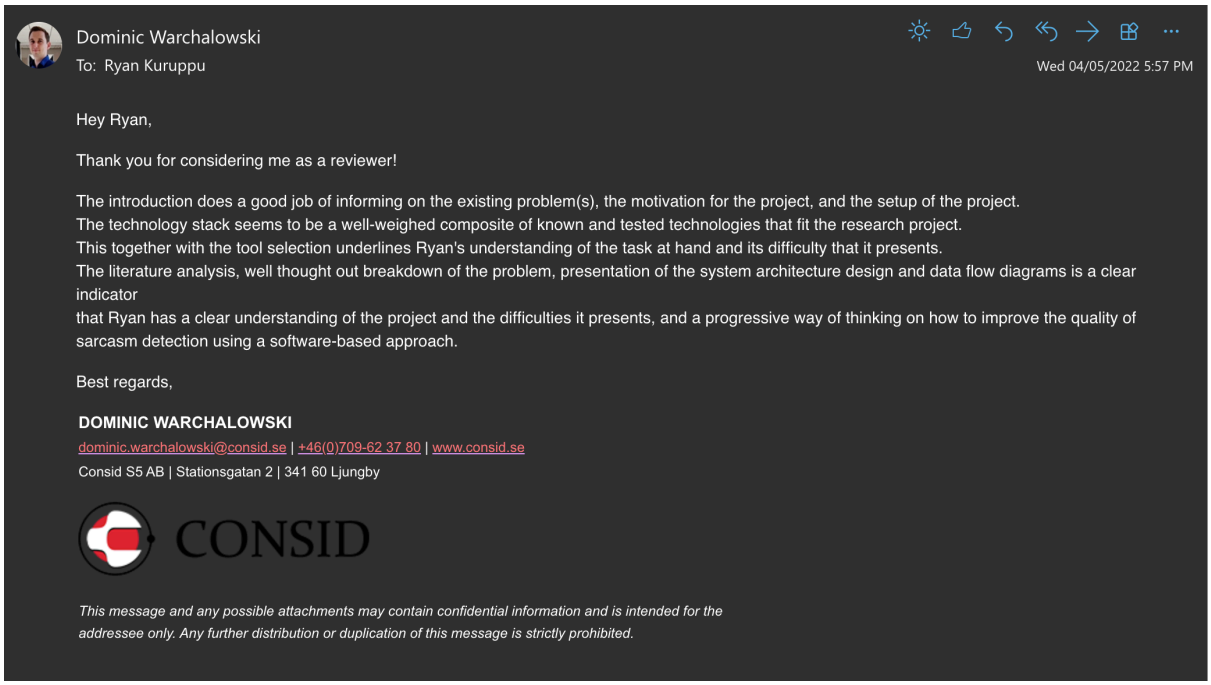
Reviewer 3	
Reviewer	Dominic Warchalowski
Qualifications & Background	- BSc (Hons) in Applied Sciences, Rotterdam University of Applied Sciences - Graduation project was based on researching and developing novel oxime compounds with the ability to regenerate organophosphate inhibited acetylcholinesterase.
The introduction does a good job of informing on the existing problem(s), the motivation for the project, and the setup of the project. The technology stack seems to be a well-weighed composite of known and tested technologies that fit the research project. This together with the tool selection underlines Ryan's understanding of the task at hand and its difficulty that it presents. The literature analysis, well thought out breakdown of the problem, presentation of the system architecture design and data flow diagrams is a clear indicator that Ryan has a clear understanding of the project and the difficulties it presents, and a progressive way of thinking on how to improve the quality of sarcasm detection using a software-based approach.	

Appendix V - Expert Review – Proof



Ruchira Perera <ruchira@ascentic.se>
To: Ryan Kuruppu
Today at 3:49PM

Interesting and challenging topic. As I understand, the project aims to improve the accuracy and efficiency of sarcasm detection. Therefore this will be an effective solution for enormous social media platforms to understand their user's positive negative and Neutral statements with a higher accuracy rate. Impressive problem identification by analysing existing systems and research gaps. The Literature review covers most of the detection approaches, technologies, and risk factors. The system was well defined by following 4-tier architecture and the documentation explains the overall system with multiple diagrams and necessary steps. I have a positive mindset on technology stack selection and as an improvement, I would like to suggest to little bit of concern on system efficiency. Overall good effort.



Dominic Warchalowski
To: Ryan Kuruppu
Wed 04/05/2022 5:57 PM

Hey Ryan,

Thank you for considering me as a reviewer!

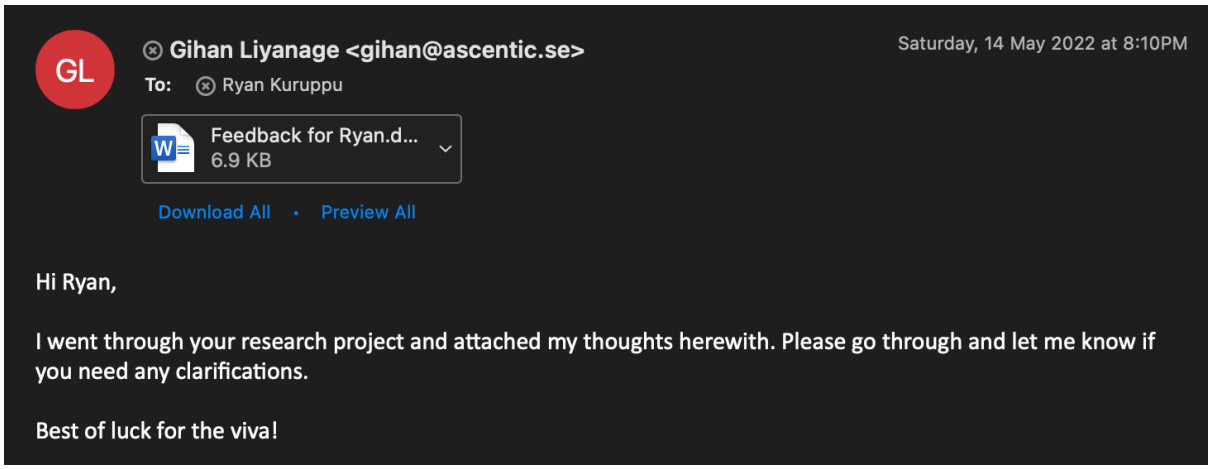
The introduction does a good job of informing on the existing problem(s), the motivation for the project, and the setup of the project.
The technology stack seems to be a well-weighed composite of known and tested technologies that fit the research project.
This together with the tool selection underlines Ryan's understanding of the task at hand and its difficulty that it presents.
The literature analysis, well thought out breakdown of the problem, presentation of the system architecture design and data flow diagrams is a clear indicator
that Ryan has a clear understanding of the project and the difficulties it presents, and a progressive way of thinking on how to improve the quality of sarcasm detection using a software-based approach.

Best regards,

DOMINIC WARCHALOWSKI
dominic.warchalowski@consid.se | +46(0)709-62 37 80 | www.consid.se
Consid S5 AB | Stationsgatan 2 | 341 60 Ljungby

CONSID

This message and any possible attachments may contain confidential information and is intended for the addressee only. Any further distribution or duplication of this message is strictly prohibited.



GL
Gihan Liyanage <gihan@ascentic.se>
To: Ryan Kuruppu
Saturday, 14 May 2022 at 8:10PM

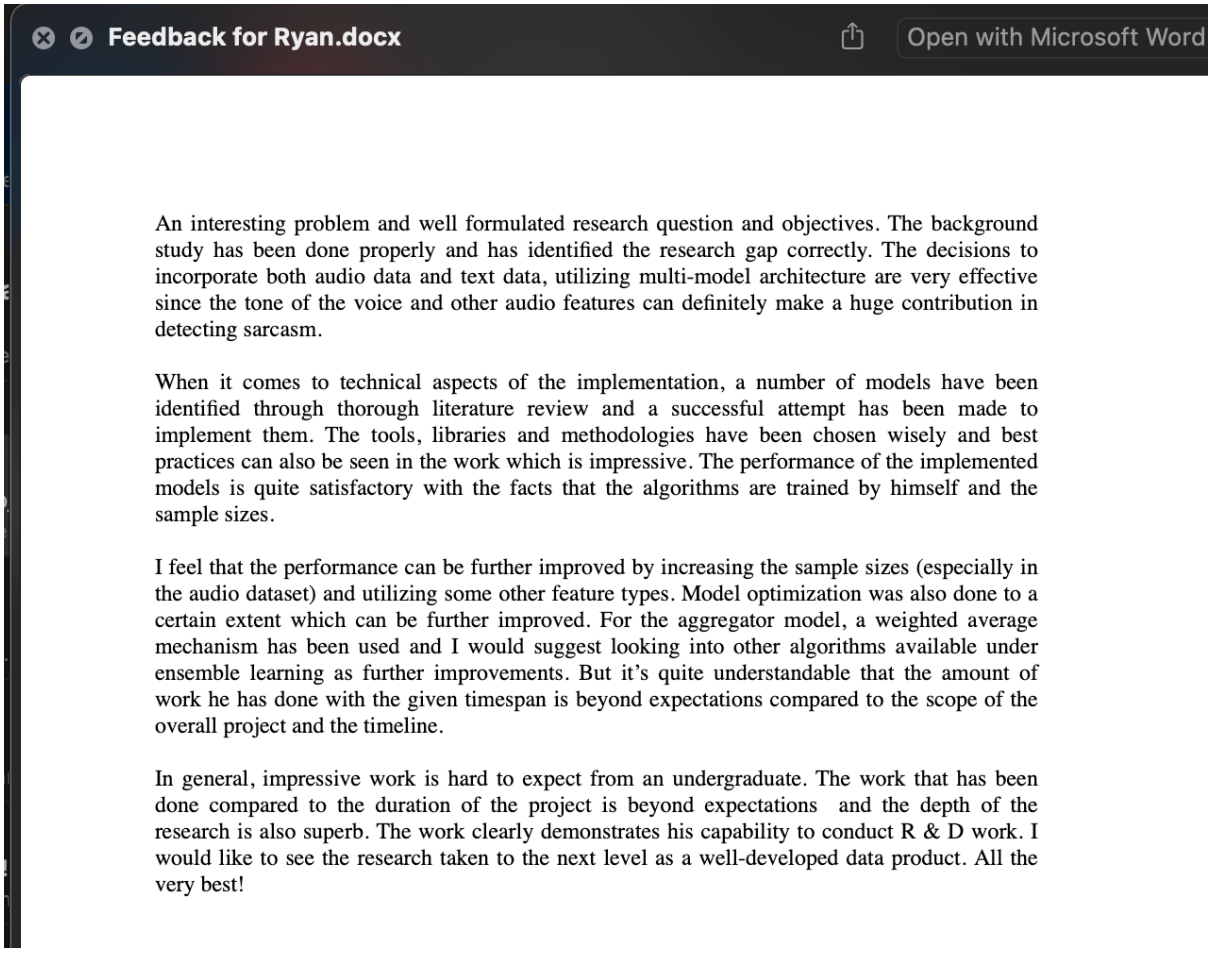
Feedback for Ryan.d...
6.9 KB

Download All • Preview All

Hi Ryan,

I went through your research project and attached my thoughts herewith. Please go through and let me know if you need any clarifications.

Best of luck for the viva!



Feedback for Ryan.docx

Open with Microsoft Word

An interesting problem and well formulated research question and objectives. The background study has been done properly and has identified the research gap correctly. The decisions to incorporate both audio data and text data, utilizing multi-model architecture are very effective since the tone of the voice and other audio features can definitely make a huge contribution in detecting sarcasm.

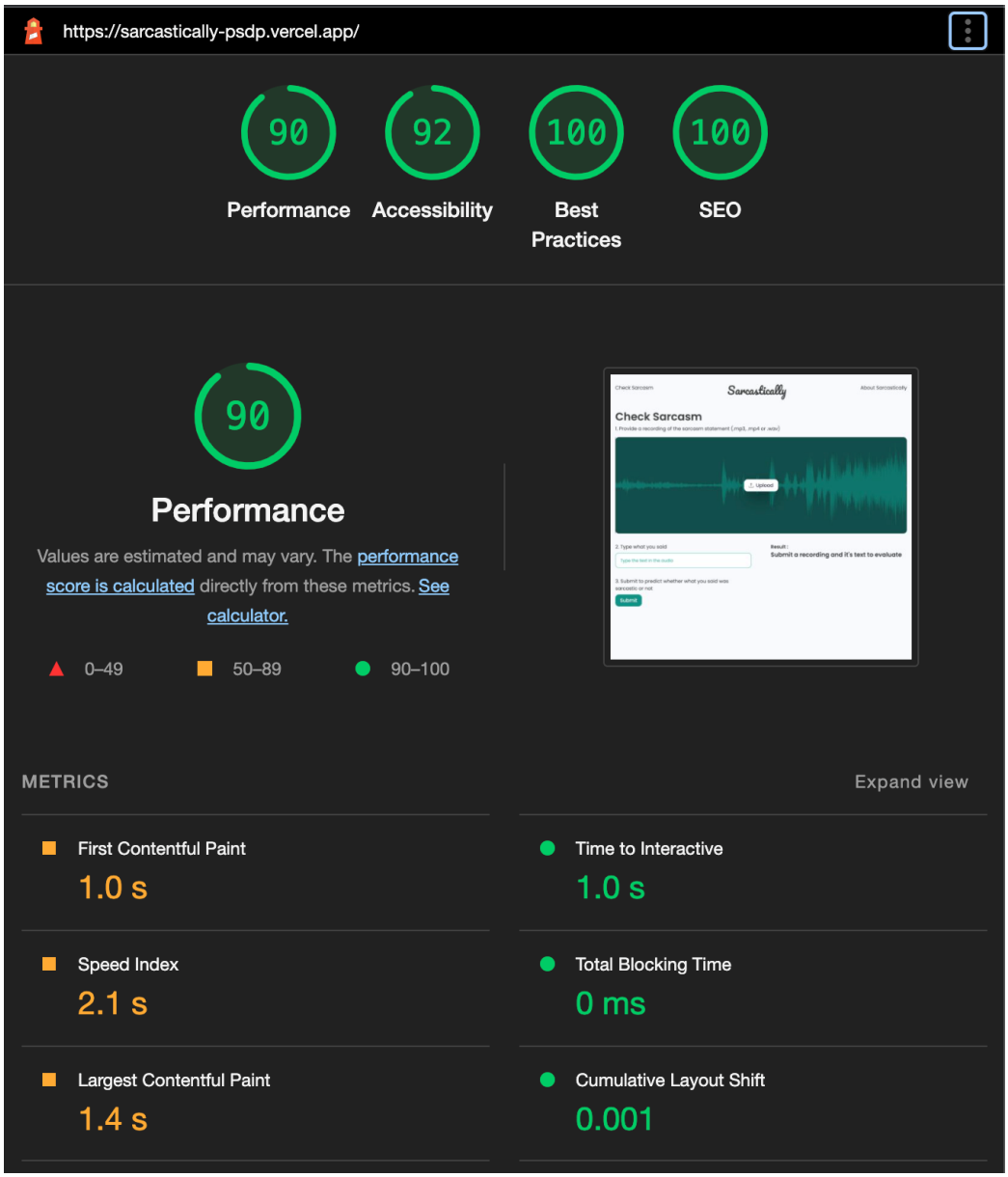
When it comes to technical aspects of the implementation, a number of models have been identified through thorough literature review and a successful attempt has been made to implement them. The tools, libraries and methodologies have been chosen wisely and best practices can also be seen in the work which is impressive. The performance of the implemented models is quite satisfactory with the facts that the algorithms are trained by himself and the sample sizes.

I feel that the performance can be further improved by increasing the sample sizes (especially in the audio dataset) and utilizing some other feature types. Model optimization was also done to a certain extent which can be further improved. For the aggregator model, a weighted average mechanism has been used and I would suggest looking into other algorithms available under ensemble learning as further improvements. But it's quite understandable that the amount of work he has done with the given timespan is beyond expectations compared to the scope of the overall project and the timeline.

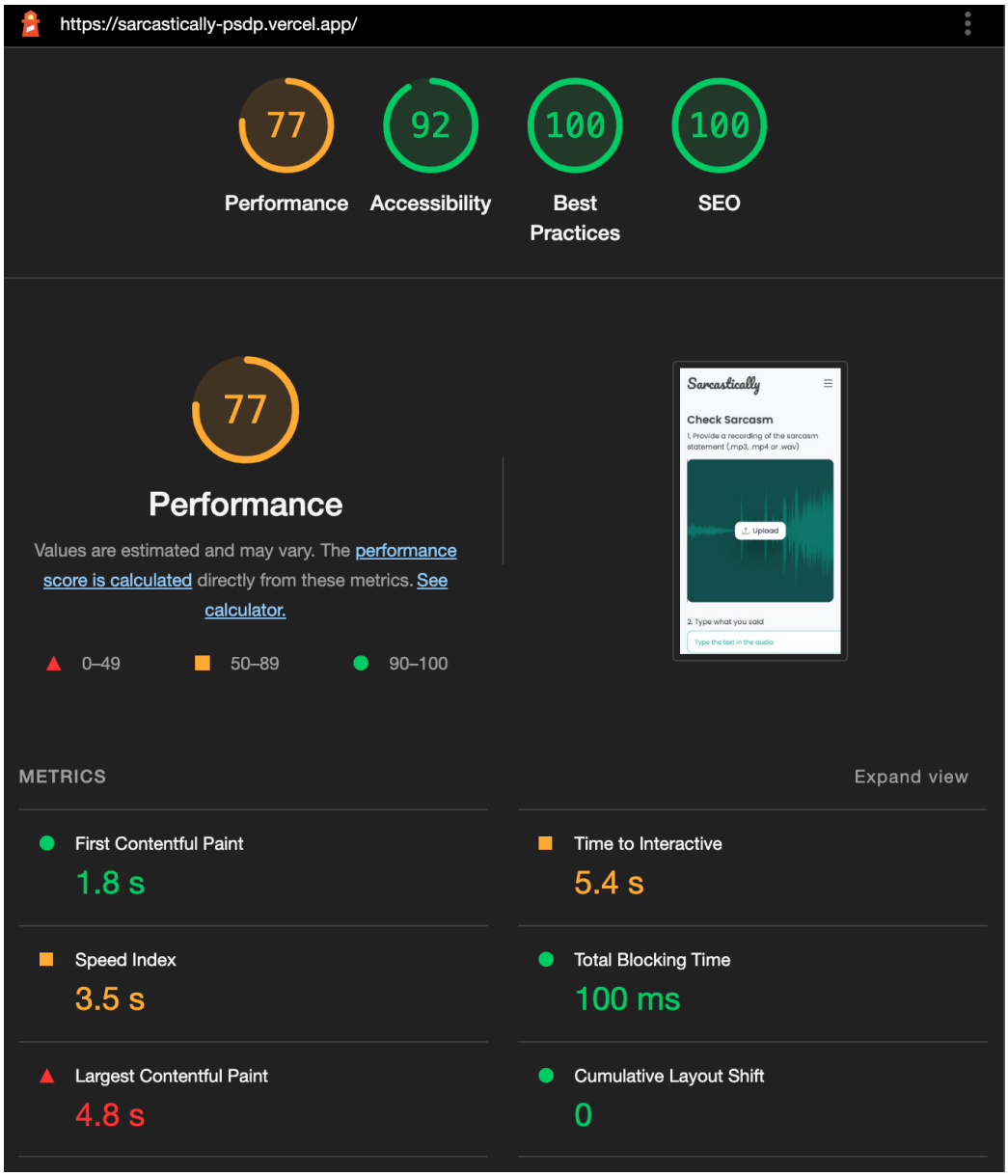
In general, impressive work is hard to expect from an undergraduate. The work that has been done compared to the duration of the project is beyond expectations and the depth of the research is also superb. The work clearly demonstrates his capability to conduct R & D work. I would like to see the research taken to the next level as a well-developed data product. All the very best!

Appendix VI – Lighthouse Score

For Desktop



Mobile



Testing Environment

Appendix VII – Focus Group Survey Questions

What is your name ? (First and Last)

What's your gender ? Hide options

- Male
- Female

What age group do you belong to ? Hide options

- Under 18
- 18-30
- 30+
- 50+

What's your highest level of education

What did you hear in the first audio clip ?

Did you think it was sarcastic ? Hide options

- Yes
- No

Did Sarcastically get the same answer ? Hide options

- Yes
- No

Was the interface easy to understand ? Hide options

- Yes
- No

Any improvements you think are necessary ?

Appendix VIII – Focus Group Responses

In-Person Survey (Part 1)

What did you hear in the first audio clip ?

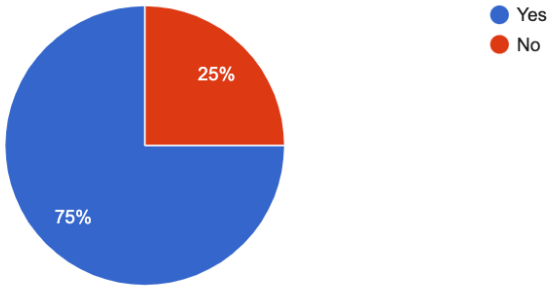
4 responses

- it just a privilege to watch your mind at work
- It's a privilege to watch your mind at work
- Its just a privilege to watch your mind at work
- Nice job Joe, you're quite the craftsman

Did you think it was sarcastic ?

Copy

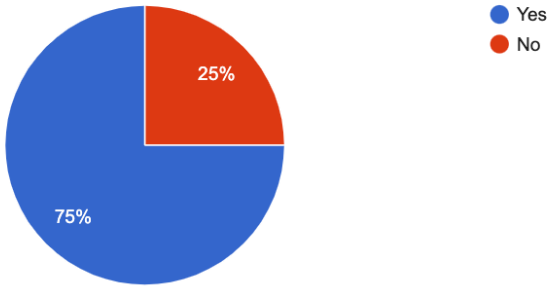
4 responses



Did Sarcastically get the same answer ?

Copy

4 responses



In-Person Survey (Part 2)

What did you hear in the second audio clip ?

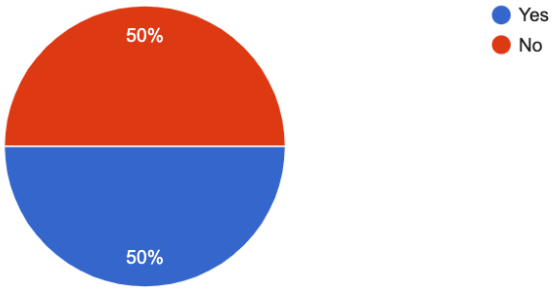
4 responses

- i got some feelers out, in the meantime listen to this
- Just the latest copy of applied particle physics quarterly, its so weird that your's came and mine didn't
- there's just no room for you in my wallet
- Let me see the ring; Great, okay. Here

Did you think it was sarcastic ?

Copy

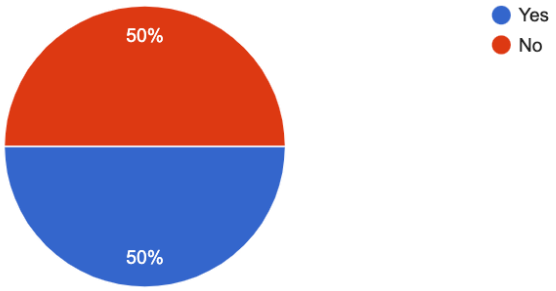
4 responses



Did Sarcastically get the same answer ?

Copy

4 responses



In-Person Survey (Part 3)

What did you hear in the third audio clip ?

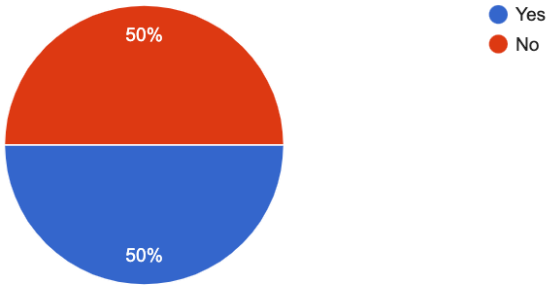
4 responses

- you know this can go on us, why aren't you just come with us
- I think tonight was a very good start
- perhaps someone took it off by mistake
- Uh, a t-shirt that says I don't belong here

Did you think it was sarcastic ?

Copy

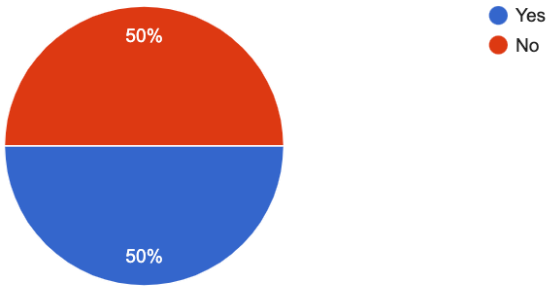
4 responses



Did Sarcastically get the same answer ?

Copy

4 responses

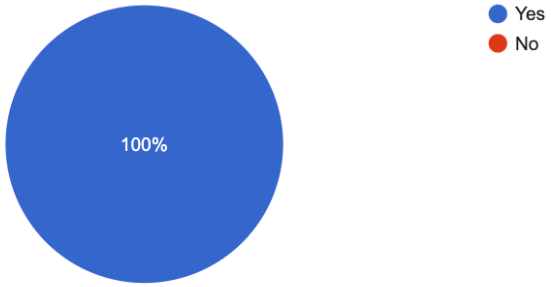


In-Person Survey (Part 4)

Was the interface easy to understand ?

Copy

4 responses



Any improvements you think are necessary ?

3 responses

None needed

n/a

A history of clips checked.