



CONVERSATIONAL AGENT FOR WEBSITE ENGAGEMENT USING LLAMA 2

A PROJECT REPORT

Submitted by

**AISHWARYA S
BOLLINENI MANOGNA**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SRI VENKATESWARA COLLEGE OF ENGINEERING
(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)
ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2024

SRI VENKATESWARA COLLEGE OF ENGINEERING
(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)

ANNA UNIVERSITY, CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Project report "CONVERSATIONAL AGENT FOR WEBSITE ENGAGEMENT USING LLAMA 2" is the bonafide work of "AISHWARYA S (2127200801007) and BOLLINENI MANOGNA (2127200801018)" who carried out the Project work under my supervision.

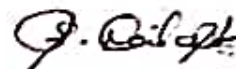

SIGNATURE

Dr. P. Leela Rani, M.E., Ph.D.,

SUPERVISOR

ASSOCIATE PROFESSOR

INFORMATION TECHNOLOGY



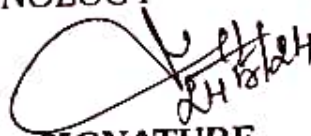
SIGNATURE

Kailash S

EXTERNAL SUPERVISOR

CO-FOUNDER

KINSALE PRIVATE LIMITED


SIGNATURE

Dr. V. Vidhya, M.E, Ph.D.,

HEAD OF THE DEPARTMENT

INFORMATION TECHNOLOGY

Submitted for the project viva-voce examination held on 27/05/24


INTERNAL EXAMINER


EXTERNAL EXAMINER



 www.kinsale.in

 91-98944 53271

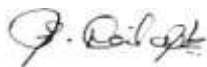
 kailash@kinsale.in

This is to certify that the following students have successfully completed their final year project titled **“Conversational Agent for Website Engagement using LLaMa2”** in our company during the period 2nd Jan 2024 to 2nd May 2024.

2127200801007 AISHWARYA S

2127200801018 BOLLINENI MANOGNA

During the entire project duration, we found them to be very innovative and procedural in their approach. Wish them all success for their future career.



Kailash S
Co-Founder
Kinsale Private Limited - CIN: U72900AP2020PTC115885

ABSTRACT

In an era marked by the ubiquitous deluge of information, the project titled “Conversational Agent for Website Engagement using LLaMA 2” emerges as a beacon of innovation, offering a transformative solution to heighten user engagement on websites. The project strategically employs the cutting-edge capabilities of LLaMA 2, an advanced artificial intelligence developed by Meta AI. At its core, this initiative endeavors to craft a conversational agent that transcends the limitations of traditional chatbots. Powered by the nuanced understanding and generation of human-like text, this agent aspires to create an environment conducive to natural and context-aware conversations on websites.

The overarching objective is to simplify the user experience, allowing individuals to seamlessly connect their queries with the most pertinent and meaningful content. Distinguishing itself from conventional approaches, the conversational agent integrates an inventive technique known as Graph-Based Prioritization. By incorporating this novel approach, the project aims to elevate the conversational interface to new heights of effectiveness and user satisfaction. In the intricate domain of information retrieval, the project strategically leverages the capabilities of PageRank, a venerable algorithm recognized for its efficacy.

ACKNOWLEDGEMENT

We thank our Principal **Prof. S. Ganesh Vaidyanathan**, Sri Venkateswara College of Engineering for being the source of inspiration throughout our study in this college.

We express our sincere thanks to **Dr. V. Vidhya, M.E., Ph.D.**, Head of Department, Information Technology for her permission and encouragement accorded to carry out this project.

We are also thankful to the project coordinators **Dr. G. Sumathi, M.E., Ph.D.**, Professor and **Dr. N. Gobalakrishnan, M.Tech., Ph.D.**, Associate Professor for their continual support and assistance.

With profound respect, we express our deep sense of gratitude and sincere thanks to our guide, **Dr. P. Leela Rani, M.E., Ph.D.**, Associate Professor for her continuous and valuable guidance throughout this project.

We are also thankful to all the faculty members and supporting staff of the Department of Information Technology.

**AISHWARYA S
BOLLINENI MANOGNA**

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	x
	LIST OF TABLES	xi
	LIST OF ABBREVIATIONS	xii
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 SOURCES OF TEXTUAL DATA	2
	1.3 STRUCTURE OF A WEBSITE	2
	1.3.1 DOM Structure	2
	1.3.2 Styling and Logic	3
	1.4 WEB SCRAPING AND CRAWLING	4
	1.5 GRAPH THEORY AND PAGERANK	5
	1.6 LARGE LANGUAGE MODELS	5
	1.7 NEED FOR WEBSITE ENGAGEMENT	6
2	LITERATURE SURVEY	8
3	CONVERSATIONAL AGENT FOR WEBSITE ENGAGEMENT	17
	3.1 OVERVIEW	17
	3.2 EXISTING SYSTEM	17

3.3 ARCHITECTURE OF CONVERSATIONAL AGENT	19
3.4 FRONTEND MODULE	20
3.5 GRAPH PROCESSING MODULE	20
3.6 DATA PROCESSING MODULE	21
3.6.1 Web Scrapping	21
3.6.2 Web Crawling	21
3.6.3 Data Preparation	22
3.6.4 Tokenization of Data	23
3.7 STORAGE MODULE	23
3.7.1 Embedding Unit	24
3.7.2 Ranking	24
3.7.3 Reranking	25
3.7.4 Vector Database	26
3.8 MODEL CUSTOMISATION MODULE	26
3.8.1 Zero-shot Prompting	26
3.8.2 Fine-tuning LLMs	27
3.8.3 Full Fine-tuning	27
3.9 PARAMETER EFFICIENT FINE-TUNING	29
3.9.1 Low Rank Adaptation	29
3.9.2 Quantized Low Rank Adaptation	30
3.10 OUTPUT MODULE	33
3.10.1 Retrieval Augmented Generation	33
3.10.2 Content Delivery	34

4	IMPLEMENTATION	36
	4.1 DATA COLLECTION AND FORMATTING	36
	4.1.1 Data Collection	36
	4.1.2 Data Wrangling	37
	4.1.3 Alpaca Formatting	37
	4.2 DATA STORE FOR MODEL	38
	4.2.1 Embedding using Voyage Large 2	38
	4.2.2 Indexing using Pagerank	39
	4.3 MODEL CONFIGURATION	40
	4.3.1 Fine-tuning Model	41
	4.3.2 Prompt Engineering	42
	4.4 RESPONSE GENERATION	43
	4.4.1 Question Condensation	44
	4.4.2 Chat Engine	45
	4.5 HOSTING THE MODEL	46
	4.5.1 Model Provisioning	46
	4.5.2 Model Utilisation	47
	4.6 RESPONSE PRESENTATION	48
	4.6.1 Response Alignment using Markdown	48
	4.6.2 Streaming Response	48
	4.7 USER INTERFACE FOR CONVERSATIONAL AGENT	49
	4.7.1 Frontend Development using Svelte.js	50
	4.7.2 Tailwind CSS for Styling	52
	4.8 METRICS	52

5	CONCLUSION AND FUTURE SCOPE	54
	5.1 CONCLUSION	54
	5.2 FUTURE SCOPE	54
	REFERENCES	56

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	DOM Structure	3
2.1	Broad Classification of NLP	8
3.1	Architecture of Conversational Agent using LLaMA 2	19
3.2	Working of an Embedding Model	24
3.3	LLaMA 2 Layers	28
3.4	VRAM Requirement of LLaMA 2-7b	28
3.5	VRAM Requirement after Half Precision (fp16)	29
3.6	LoRA Parameters, Gradients and Optimizer State	30
3.7	QLoRA Parameters, Gradients and Optimizer State	31
3.8	Layers of Fine-tuned LLaMA 2	32
4.1	Training Loss Graph of Model Training	42
4.2	Model Deployment in Ollama Platform	46
4.3	Screenshot of the User Interface	50

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO
2.1	Language Distribution in Pre-training Data LLaMA 2	14
4.1	Summary of Metrics	53

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BLEU	Bilingual Evaluation Understudy
BPE	Byte Pair Encoding
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DBMS	DataBase Management System
DL	Deep Learning
DOM	Document Object Model
GLoVe	Global Vectors
GPT	Generative Pre-trained Transformer
GPU	Graphical Processing Unit
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
LLM	Large Language Model
LoRA	Low-Rank Adaptation
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding

PEFT	Parameter Efficient Fine-Tuning
PLM	Pre-trained Language Model
PPO	Proximal Policy Optimization
QA	Question Answering
QLoRA	Quantized Low-Rank Adaptation
RAG	Retrieval Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
ROGUE	Recall-Oriented Understudy for Gisting Evaluation
URL	Uniform Resource Locator
VRAM	Video Random Access Memory

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Natural Language Processing (NLP)[25] is a vibrant and revolutionary domain within Artificial Intelligence (AI)[46], dedicated to empowering computers to comprehend, interpret and produce human language. The interdisciplinary field merges principles from linguistics, computer science and Machine Learning (ML)[47]. The applications of NLP span a broad spectrum, ranging from the development of chatbots for real-time customer assistance to the creation of automated translation services that facilitate communication across diverse languages. NLP acts as the translator, deciphering user messages and generating natural language responses for chatbots. This empowers chatbots to hold meaningful conversations, enhancing user experience and efficiency.

Chatbots[7] have become ubiquitous in various real-world scenarios, offering seamless interactions and assistance across diverse industries. From customer service platforms to e-commerce websites, chatbots are deployed to provide immediate support and information to users. In healthcare, they aid in scheduling appointments and answering basic medical queries. Educational institutions utilise chatbots for student support and course enrollment guidance. Their versatility makes them indispensable tools in modern-day interactions between businesses and consumers. Incorporating chatbots into websites guarantees constant availability, rapid responses, personalised interactions and enhanced engagement. As advancements in AI continue, chatbots are expected to play an even greater role in shaping the future of customer service.

1.2 SOURCES OF TEXTUAL DATA

Different types of content[15] a website may contain are:

- **Website Content:** This refers to the textual content directly displayed on the website's pages. Textual content includes articles, blog posts, product descriptions, about pages, contact information, etc.
- **Document Files:** Websites may host various types of document files for users to download or view. Common document formats include Portable Document Format (PDFs), word documents, presentations (e.g., PowerPoint), spreadsheets (e.g. Excel) and text files.
- **Images:** Websites frequently feature images which include photos, illustrations, diagrams, charts, graphs, logos, icons, etc. Images may be embedded within web pages or linked to external sources.
- **Audio/Video:** Websites increasingly incorporate audio and video content to engage users and deliver information. Audio content may include podcasts, music tracks, sound effects and audio articles. Video content encompasses webinars, tutorials, product demos, etc.

1.3 STRUCTURE OF A WEBSITE

The structure of a website encompasses three main components: the Document Object Model (DOM)[40] structure, Styling and Logic.

1.3.1 DOM Structure

The DOM structure represents the hierarchical organisation of HyperText Markup Language (HTML)[37] elements within a web page. It forms the backbone of the structure of the web page and content, defining the relationships between different elements. HTML tags such as <div>, <p>, <header>, <footer>, , , etc., create the building blocks of the DOM structure. Each element in the DOM tree can contain child elements, attributes and textual content. The DOM structure determines the layout, flow and

accessibility of content on the webpage. The key elements of the DOM structure are depicted in Figure 1.1.

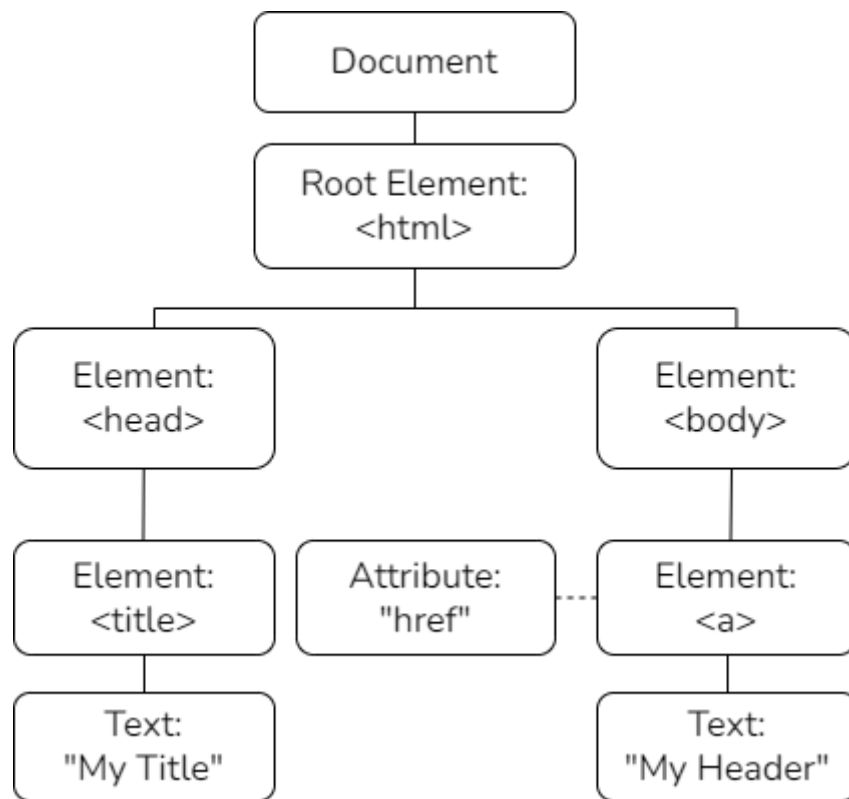


Figure 1.1 DOM Structure

1.3.2 Styling and Logic

Styling refers to the visual presentation and aesthetics of the website, achieved through Cascading Style Sheets (CSS)[13]. CSS rules define how HTML elements should be displayed, including properties such as colours, fonts, margins, padding, borders and positioning.

The logic layer of a website involves interactivity and dynamic behaviour, typically implemented using JavaScript[12]. JavaScript enables client-side scripting, manipulation of DOM elements, handling user interactions and dynamic content updates without requiring page reloads. In addition to client-side scripting, server-side logic may be implemented for handling tasks such as data processing, authentication and server communication.

1.4 WEB SCRAPING AND CRAWLING

Extracting data from the web involves the automated retrieval and extraction of information from various online sources such as websites, documents, images and multimedia content. This process typically utilises web scraping and crawling techniques[24], which involve programmatically accessing web pages, parsing their HTML structure and extracting relevant data based on specified criteria.

Web scraping[11] is the process of automatically extracting specific data from websites. It involves fetching HTML content from web pages and parsing it to extract the desired information. This technique targets specific data elements from one or multiple web pages. These tools simulate human browsing behaviour to access web pages, retrieve content and extract structured data. The extracted data can be stored in various formats, such as Comma Separated Values (CSV)[37], JavaScript Object Notation (JSON)[37] or databases, for further analysis, visualisation or integration into other applications.

Web crawling[18] is the automated process of systematically browsing the web to discover and index web pages. Unlike web scraping, which focuses on extracting specific data from known web pages, web crawling is more generalised and aims to explore the structure of the web comprehensively. Web crawlers start from a set of seed Uniform Resource Locators (URLs) and recursively follow hyperlinks to visit linked pages. They gather information about each page, including its URL, content, metadata and links to other pages. Search engines use web crawling to index billions of web pages, enabling users to find relevant information through search queries.

1.5 GRAPH THEORY AND PAGERANK

Graph theory[53] is a mathematical discipline that explores the properties and relationships of graphs, which represent pairwise connections between

objects. In this context, a graph comprises vertices or nodes and edges or links that delineate the relationships between pairs of vertices. These structures find applications across various domains, from computer science to social networks, where they model interactions, dependencies and networks.

Pagerank[52] is an algorithm that leverages principles from graph theory to evaluate the significance of web pages within the structure of the internet. It operates by treating web pages as vertices in a directed graph and hyperlinks between pages as directed edges. This algorithm determines the importance of a web page by considering the interconnectedness of pages and the quality of links pointing to them. By employing Pagerank, Google introduced a revolutionary approach to ranking web pages in search engine results. This algorithm assigns each page a Pagerank score, which signifies its importance in the network of web pages. This score is calculated iteratively, considering the structure of the web graph and the relevance of inbound links.

Pagerank effectively addresses the challenge of determining the significance of a page amidst the vastness of the web, providing users with more relevant search results. At the heart of Pagerank lies the concept of the random surfer model. This model imagines a hypothetical surfer navigating the web by clicking on links, moving from page to page. The likelihood of landing on a particular page at any given moment is determined by its Pagerank score and the number of incoming links. The importance of each page is proportional to the sum of the importance of pages linking to it, iteratively calculated until convergence. This probabilistic approach ensures that highly linked pages are considered more important and influential within the web ecosystem.

1.6 LARGE LANGUAGE MODELS

Within the domain of AI, Large Language Models (LLMs)[2] have brought about a significant transformation in the field of NLP. These advanced

models are engineered to understand and produce text that closely resembles human language, empowering them to execute various language-oriented tasks with exceptional precision and fluency. The advent of LLMs has propelled the capabilities of AI systems, enabling them to comprehend, analyse and generate human language in a manner that closely emulates human thought processes and communication patterns.

LLMs commonly utilise Deep Learning (DL)[1] architectures, frequently making use of transformer-based models. These models excel at processing sequential data, enabling them to capture intricate connections within text-based information effectively. Equipped with attention mechanisms and multi-layer architectures, these expansive Language Models (LMs)[2] can adeptly discern and understand nuanced patterns and relationships present in language datasets.

A key characteristic of LLMs is their ability to derive insights from extensive datasets containing copious amounts of text sourced from various mediums like books, articles and websites. Through the ingestion and analysis of such expansive data, LLMs develop a thorough grasp of language structure and semantics, facilitating the creation of contextually aware, precise and effective chat systems.

1.7 NEED FOR WEBSITE ENGAGEMENT

Website engagement[17] is pivotal for businesses aiming to create meaningful interactions with their online audience. It serves as the foundation for achieving various objectives such as enhancing user experience, increasing brand awareness, driving conversions and fostering customer loyalty. By prioritising engagement, websites can establish stronger connections with visitors, encouraging them to explore the site, interact with its content, and return for future interactions.

In the digital landscape, chatbots, powered by AI and NLP, serve as virtual assistants capable of interacting with users in real-time. By integrating chatbots into a website, businesses can provide instant support, answer user inquiries and guide visitors through their online journey. Chatbots provide immediate assistance, allowing users to get answers to their questions or resolve issues without delay. This instant gratification enhances the overall user experience and encourages visitors to stay longer on the site.

Chatbots serve as valuable tools for lead generation and customer engagement. By proactively engaging visitors, answering their questions and guiding them through the website, chatbots can capture valuable leads and nurture them towards conversion. Additionally, chatbots can gather valuable feedback from users, helping businesses gain insights into customer preferences and areas for improvement. By leveraging chatbot technology, businesses can create more meaningful experiences for their online audience, driving higher engagement, customer satisfaction and ultimately, business success.

CHAPTER 2

LITERATURE SURVEY

The literature survey delves into previous research, emphasising trends and methodologies, thereby laying the groundwork for a crucial understanding essential for advancing the project.

NLP[25] has gained considerable interest recently due to its capacity to computationally represent and analyse human language, with its applications spanning diverse fields such as machine translation, email spam detection, medical applications and Question Answering (QA). NLP encompasses two key components[16]: Natural Language Understanding (NLU) and Natural Language Generation (NLG), as illustrated in Figure 2.1.

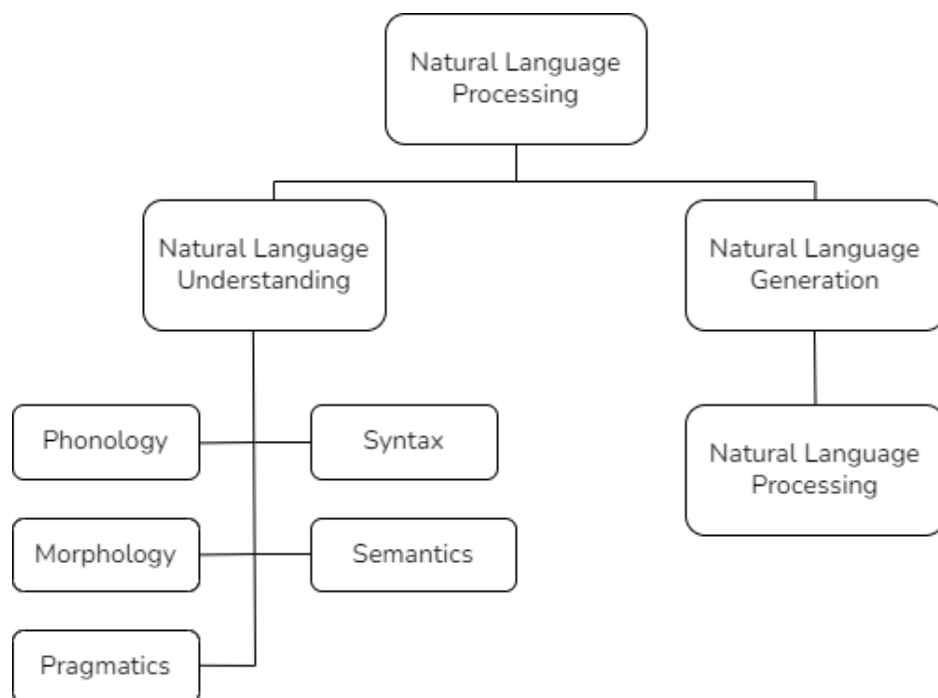


Figure 2.1 Broad Classification of NLP

NLU focuses on enabling computers to comprehend and interpret human language input, allowing them to extract meaning, identify entities and understand the context of the text. This involves tasks such as sentiment analysis, named entity recognition and language translation. On the other hand, NLG involves the process of generating human-like language output from structured data or information stored in a machine-readable format. NLG systems can create coherent and contextually relevant text, such as summaries, responses or reports, based on the input data. Both NLU and NLG play crucial roles in various NLP applications, including virtual assistants, chatbots, automated customer service systems and language translation tools. By combining advances in ML, DL and NLP techniques, researchers and developers continue to enhance the capabilities of NLU and NLG systems, driving innovation in human-computer interaction and communication.

Data wrangling[31], also known as data preparation, is a crucial step in the data analysis process where raw data is transformed and cleaned to make it suitable for analysis. Various techniques are employed in data wrangling to ensure the data is accurate, consistent and ready for further processing. These techniques include data cleaning to remove errors and inconsistencies, data transformation to convert data into a usable format, data integration to combine data from multiple sources and data enrichment to add value to the existing data. Data wrangling is essential in various use cases such as business intelligence, ML, data mining and data visualisation. In business intelligence, data wrangling helps in preparing data for reporting and analysis. In ML, clean and well-prepared data is crucial for training accurate models. Data wrangling is also important in data mining to extract valuable insights from large datasets and in data visualisation to create meaningful visual representations of data. Overall, data wrangling plays a vital role in ensuring the quality and reliability of data for effective decision-making and analysis in diverse fields.

Web scraping[11] is a fundamental process in extracting valuable information from web pages for various applications such as online price monitoring, sentiment analysis and data aggregation. Traditional web scraping methods often involve creating a DOM tree to represent the structure of the web document, which can be time-consuming. However, time efficiency can be improved by utilising additional information obtained during the crawling process to enhance content extraction without the need for a DOM tree. By analysing specific elements within web pages and optimising the extraction process, data retrieval can be streamlined and the overall efficiency of web scraping tasks can be improved.

Web scraping and web crawling[24] are essential processes in the realm of data extraction from the internet. Web scraping involves extracting specific information from websites using automated tools, mimicking human browsing behaviour to collect structured data efficiently. On the other hand, web crawling is a broader process that involves systematically browsing the web to index and retrieve information from multiple pages or websites. The key difference lies in their scope: web scraping targets specific data points, while web crawling focuses on exploring and indexing vast amounts of information. Both techniques are crucial for gathering data for various purposes, such as market research, competitive analysis and academic research. Web scraping and crawling can be approached using a variety of tools and technologies, ranging from custom scripts to specialised software, each tailored to the specific requirements of the data extraction task at hand. By leveraging these techniques effectively, organisations and researchers can harness the power of web data to gain valuable insights and make informed decisions.

PageRank algorithm[27] is a widely-used algorithm for prioritisation in various applications, such as search engines. It was developed as a way to measure the importance of web pages. The algorithm assigns a numerical value,

called PageRank score, to each page based on the number and quality of links pointing to it. Pages with higher PageRank scores are considered more important and are prioritised in search engine results. In the context of prioritisation, the PageRank algorithm is used to rank and order web pages based on their relevance to a given query or topic. When a user enters a search query, the search engine retrieves a list of web pages that contain relevant information. The PageRank algorithm then analyses the links between these pages and assigns them a score based on their importance. Pages with higher scores are more likely to be displayed at the top of the search results, as they are considered more relevant to the query. The algorithm takes into account both the quantity and quality of links. A page with many incoming links from other high-ranking pages will have a higher PageRank score. Additionally, the algorithm considers the relevance of the linking pages. If a highly ranked page links to another page on a similar topic, the linked page will receive a boost in its PageRank score. Overall, the PageRank algorithm plays a crucial role in the prioritisation of search results. By analysing the links between web pages and assigning them scores based on their importance, it helps search engines deliver more relevant and useful results to users.

Graph-based prioritisation[45] is a methodology that uses graph theory to assess and prioritise certain actions over the others. Graph theory is a mathematical approach that uses nodes and edges to represent and analyse relationships between different elements. In the context of river management, graph-based prioritisation focuses on identifying barriers that have the most significant impact on the structural connectivity of river basins and determining which connections should be restored or enhanced to improve overall connectivity. Graph-based prioritisation, exemplified by algorithms like PageRank, operates by assigning numerical weights to elements within a hyperlinked set of documents, such as web pages, based on their relative

importance within the network. In a web context, the internet is represented as a graph, with pages as nodes and hyperlinks as edges, and the importance of each page is determined by factors like the number and quality of incoming links. Once the scores are computed for all pages in the graph through iterative processes, fetching more important pages involves prioritising pages based on their scores. Pages with higher scores, indicative of their greater importance and relevance within the network, are more likely to be displayed prominently in search results. This iterative approach ensures that the scores continuously reflect changes in the web graph, such as the addition of new pages or modifications to existing links, thereby enhancing the quality of search results by directing users to more relevant and authoritative content. Overall, graph-based prioritisation offers a promising approach for fetching more significant pages amongst the numerous pages in the web environment.

Retrieval Augmented Generation (RAG)[34] is a technique used in NLP tasks to refer to models that combine pre-trained parametric models with non-parametric memory for language generation. These models have both explicit memory, such as a dense vector index[44] and implicit memory stored in their parameters. RAG models are designed to improve the ability of pre-trained LMs to access and manipulate knowledge. They achieve this by using a neural retriever to access the non-parametric memory and retrieve relevant information from the index. The RAG models can be fine-tuned and applied to a wide range of knowledge-intensive NLP tasks. They have shown superior performance compared to task-specific architectures and parametric models in open domain QA tasks. RAG models also outperform parametric-only baseline models in language generation tasks. They generate more specific, diverse and factual language. The use of RAG models addresses some limitations of pre-trained LMs. These limitations include the inability to expand or revise their memory, the lack of insight into their predictions and the

potential production of hallucinations. By combining parametric and non-parametric memory, RAG models can directly revise and expand knowledge. Additionally, the knowledge accessed by the models can be inspected and interpreted, providing more transparency in their decision-making process. Overall, RAG models offer a promising approach for knowledge-intensive NLP tasks by effectively combining the strengths of pre-trained parametric models and non-parametric memory. They open up possibilities for more accurate and informed language generation and QA in various domains.

Recently, LMs[50] have reshaped the landscape of NLP. There is widespread acknowledgment that increasing the scale of models, encompassing factors such as training resources and model parameters, results in enhanced performance and sample efficiency across a diverse range of NLP tasks. Unique capabilities emerge in LLMs that are absent in smaller-scale counterparts. These abilities cannot be predicted solely by extrapolating performance improvements from smaller models. A pre-trained LM is presented with a prompt, typically a natural language instruction, and generates a response without requiring further training or parameter updates.

The rise in popularity of LLMs[6] has been remarkable in both academic and industrial circles, thanks to their unparalleled performance across a broad spectrum of applications. Unlike previous models tailored for specific tasks, LLMs possess a versatility that renders them indispensable for various applications, spanning general natural language tasks to specialised domains. LMs are computational models adept at understanding and generating human language. They can predict the likelihood of word sequences or generate new text based on given inputs, with N-gram models being the most prevalent type, or estimating word probabilities from contextual cues. However, LMs face challenges such as handling rare words, guarding against overfitting and

capturing intricate linguistic patterns. Ongoing research is dedicated to enhancing LM architectures and refining training methods to surmount these obstacles.

LLaMA 2[49] represents an upgraded version of LLaMA 1, trained on a fresh blend of publicly available data. The pre-training corpus was expanded by 40% and the context length of the model was doubled. It employs grouped-query attention. Variants of LLaMA 2 with 7B, 13B and 70B parameters have been released. Additionally, there were 34B variants which will not be released. LLaMA 2-Chat, an optimised version of LLaMA 2 for dialogue scenarios, has been developed. Variants with 7B, 13B and 70B parameters have been released. The initial version of LLaMA 2-Chat underwent supervised fine-tuning[16], with training conducted on 2 trillion tokens of data, offering a favourable performance-cost trade-off. LLaMA 2 exhibits a strong command of English, as evidenced in Table 2.1, thanks to its training on an extensive dataset containing a significant amount of English text and code.

Table 2.1 Language Distribution in Pre-training Data LLaMA 2

Language	Percentage	Language	Percentage	Language	Percentage
en	89.70%	ru	0.13%	uk	0.07%
unknown	8.38%	nl	0.12%	ko	0.06%
de	0.17%	it	0.11%	ca	0.04%
fr	0.16%	ja	0.10%	sr	0.04%
sv	0.15%	pl	0.09%	id	0.03%
zh	0.13%	pt	0.09%	cs	0.03%
es	0.13%	vi	0.08%	fi	0.03%
da	0.02%	ro	0.03%	hu	0.03%
sl	0.01%	bg	0.02%	no	0.03%

LLaMA 2-Chat is the culmination of several months of research and the iterative implementation of alignment techniques, encompassing both instruction tuning and Reinforcement Learning from Human Feedback

(RLHF)[21]. This process demanded substantial computational resources and annotation efforts. RLHF is a model training approach applied to a fine-tuned LM to further align its behaviour with human preferences and adherence to instructions. The model underwent iterative refinement using RLHF methodologies, specifically through rejection sampling and Proximal Policy Optimization (PPO)[21]. During the RLHF phase, the accumulation of iterative reward modelling data in conjunction with model enhancements played a pivotal role in ensuring that the reward models remained consistent within the distribution.

Chatbots[47] are increasingly being used in customer service to improve service performance and fulfil customer expectations. These chatbots have specific objectives and functions that are categorised based on scientific literature. One category focuses on improving service performance and includes functions such as interaction, entertainment and problem-solving. Interaction is important as it allows the chatbot to display characteristics similar to a human agent, such as language expertise and empathy. Entertainment can improve customer attitudes by providing funny content. Problem-solving capabilities are highly valued by users, especially for low-complexity tasks, and can even be preferred over human assistance. The other category is about fulfilling customer expectations and includes functions like trendiness and customization. Trendiness involves updating users with current news and emerging trends, which helps meet their desire for a fancy lifestyle. Customisation refers to offering personalised products or services based on customer interests and preferences. Chatbots can collect data from conversations and social media to provide customised content that makes customers feel addressed personally. Overall, chatbots in customer service aim to enhance service performance and meet the evolving expectations of customers.

LangChain[22], a custom LLM showcased in the research paper, stands at the forefront of revolutionising customer service in the digital era. By seamlessly integrating LangChain into customer service platforms, organisations can harness its power to provide efficient, personalised and responsive interactions with customers. This innovative approach marks a paradigm shift from traditional customer support techniques towards context-aware and tailored interactions. The ability of LangChain to understand context, history and nuances enables it to address a wide array of customer inquiries and issues effectively. Moreover, its integration into customer service platforms ensures round-the-clock availability and proficiency in multiple languages, enhancing the overall customer experience and redefining the customer-company relationship. Through LangChain, organisations can elevate their customer service to new heights, fostering customer satisfaction, loyalty and brand image in the competitive landscape of modern business.

CHAPTER 3

CONVERSATIONAL AGENT FOR WEBSITE ENGAGEMENT

3.1 OVERVIEW

In a realm characterised by the overwhelming abundance of information, the initiative titled "Conversational Agent for Website Engagement using LLaMA 2" emerges as a beacon of innovation, offering a transformative solution to enhance user engagement on websites. Strategically harnessing the advanced capabilities of LLaMA 2, this project aims to develop a conversational agent that surpasses the limitations of traditional chatbots. With its foundation rooted in the nuanced comprehension and generation of human-like text, this agent endeavours to cultivate an environment conducive to natural and contextually aware conversations on websites.

The primary goal is to streamline the user experience, enabling individuals to seamlessly connect their inquiries with the most relevant and meaningful content. Setting itself apart from conventional approaches, the conversational agent incorporates an innovative technique called Graph-Based Prioritization. By integrating this unique approach, the project seeks to elevate the conversational interface to new levels of effectiveness and user satisfaction.

3.2 EXISTING SYSTEM

The current system suffers from limitations in user engagement and responsiveness, prompting the adoption of the proposed "Conversational Agent for Website Engagement using LLaMA 2" to address these shortcomings effectively.

The existing system may struggle to effectively engage users due to its reliance on traditional chatbots, which often provide generic and scripted responses. This can lead to user dissatisfaction and disengagement, ultimately impacting website performance and user retention. Moreover, the limitations of the current system may hinder its ability to understand and respond to user queries in a nuanced and contextually relevant manner. Without advanced NLP capabilities, the system may struggle to interpret user intent accurately, leading to suboptimal responses and user frustration.

Certain NLP platforms like Dialogflow[33] are being used to create chatbots. Dialogflow, a powerful conversational AI platform developed by Google, offers various advantages in building NLU into applications. Its use of ML algorithms enables developers to create chatbots and virtual agents capable of understanding and responding to user queries in a conversational manner. However, Dialogflow has its own drawbacks, Dialogflow lies in its reliance on scripted responses. While these scripted responses can provide quick and consistent replies to common user queries, they may lack the flexibility and depth of understanding required for handling complex or nuanced interactions. Scripted responses are static and pre-defined, making it challenging for the chatbot to adapt to unexpected user inputs or understand context effectively. This limitation can result in less engaging and sometimes frustrating user experiences, especially when users deviate from the expected conversation flow. The reliance on scripted responses may hinder adaptability.

In contrast, the proposed system leveraging LLaMA 2 offers several compelling advantages over the current system. Firstly, it harnesses the advanced capabilities of LLaMA 2 to provide a more sophisticated and human-like conversational experience. By generating human-like text and understanding user intent more accurately, the proposed system aims to enhance user engagement and satisfaction. Additionally, the integration of Graph-Based

Prioritization in the proposed system enables more efficient content discovery and navigation. By prioritising relevant content based on user context and preferences, the system can deliver personalised and meaningful interactions, ultimately improving the overall user experience.

Overall, the transition to the proposed system represents a strategic move to address the shortcomings of the current system and leverage cutting-edge technologies to enhance user engagement and satisfaction on websites.

3.3 ARCHITECTURE OF CONVERSATIONAL AGENT

Figure 3.1 depicts the proposed architecture of Conversational Agent for Website Engagement using LLaMA 2 Model.

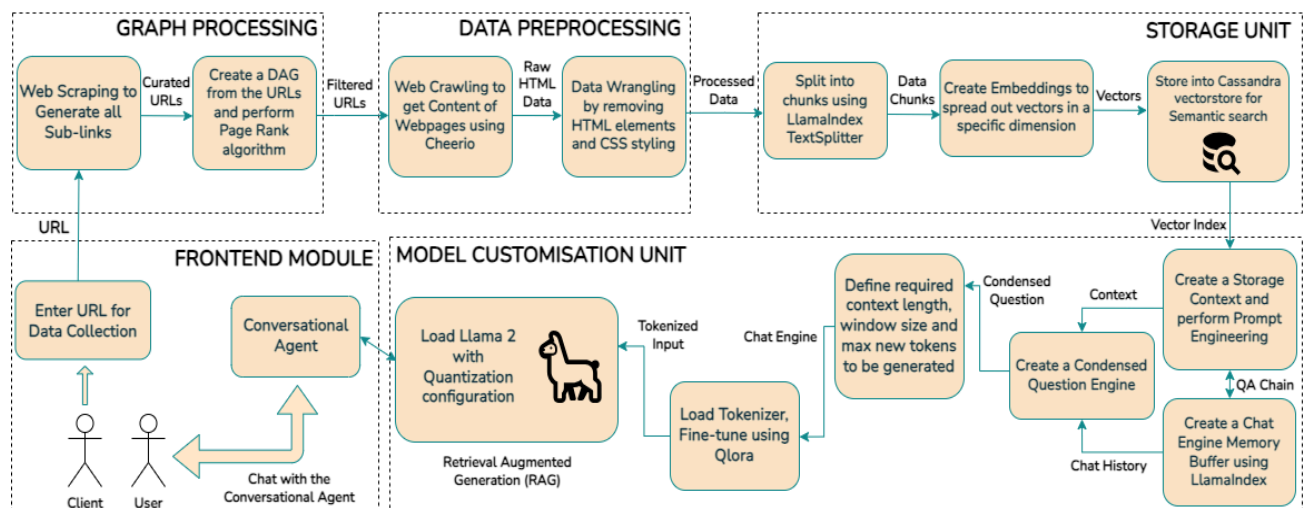


Figure 3.1 Architecture of Conversational Agent using LLaMA 2

LLaMA 2, developed by Meta AI and launched in July 2023, represents an advancement over the original LLaMA LLM. Trained on an expanded dataset with 40% more data and featuring double the context length, it stands out for its extensive training on a vast corpus of text and code. Notably, LLaMA 2 boasts an impressive size, housing 70 billion parameters, thus positioning it among the largest publicly available LLMs. Renowned for its minimal violation rates in both single and multi-turn prompts, the model's creativity tends to

increase alongside the growth in parameters. However, for tasks like Question-Answering (QA)[29], precision rather than creativity is paramount. Hence, there are limitations on utilising a 7 billion parameter model, prioritising precise responses to prompts. LLaMA 2 demonstrates a remarkable capability to comprehend the intent behind a prompt and generate responses consistent with that intent, even when faced with complex or challenging prompts.

3.4 FRONTEND MODULE

Svelte[3], a modern JavaScript framework for frontend development, simplifies User Interface (UI) creation with a component-based architecture, streamlining code management and maintenance. Unlike other frameworks, Svelte shifts the heavy lifting from the browser to the build step, resulting in optimised and highly performant applications. By compiling components to efficient JavaScript code during the build process, Svelte eliminates the need for a virtual DOM[38], leading to faster rendering and smaller bundle sizes. Its concise syntax and reactive programming model promote code clarity and reduce bugs. With a unidirectional data flow model, changes to the application state propagate predictably throughout the UI, simplifying debugging and maintenance. Developed by Rich Harris and the Svelte community, Svelte is a compelling choice for modern, efficient and scalable frontend development.

3.5 GRAPH PROCESSING MODULE

A conversational agent for a website, leveraging graph-based prioritisation and page ranking techniques can significantly enhance user experience and engagement. By constructing a content graph that maps out the website's structure and relationships between pages, developers can assign priorities to nodes based on factors such as relevance, popularity and importance. When users interact with the chatbot, their queries can be analysed to determine the underlying intent, allowing the chatbot to prioritise responses

and guide users to the most relevant pages or sections of the website. Moreover, integrating the PageRank algorithm or similar ranking methods enables the chatbot to assign importance scores to each page based on factors like inbound links and content quality, facilitating the ranking of potential responses to user queries. By incorporating these techniques into the chatbot's decision-making process, users can receive more personalised and accurate recommendations, leading to a more satisfying and efficient browsing experience.

3.6 DATA PROCESSING MODULE

Data processing[42] involves gathering and manipulating data to generate valuable insights. This essential task is integral to numerous industries and enterprises, enabling them to enhance decision-making processes through informed analysis of the data.

3.6.1 Web Scraping

Web scraping is the automated process of extracting data from websites. It involves using software tools or programming scripts to access and retrieve information from web pages. Web scraping allows users to gather data from various websites in a structured format, which can then be analysed, processed or stored for various purposes such as market research, price monitoring, content aggregation or data analysis.

3.6.2 Web Crawling

Web crawling is the automated process of systematically browsing the internet to index and collect information from web pages. It involves using web crawlers or bots, also known as spiders, to visit web pages, follow links and retrieve content. Web crawling is commonly used by search engines to index web pages for search results. This process is fundamental for maintaining

up-to-date search engine databases and for various other applications such as data mining, content aggregation and website monitoring.

3.6.3 Data Preparation

Effective training and fine-tuning of LLMs rely heavily on thorough data preparation. The quantity and quality of the data utilised during training significantly influence the performance of the LLM. Several key considerations come into play during data preparation[28]:

- **Data Size:** LLMs necessitate large volumes of data to achieve optimal training outcomes. More extensive datasets generally lead to better LLM performance.
- **Data Diversity:** It's crucial to expose LLMs to a diverse array of text data, reflective of the contexts in which they will operate. This ensures the model learns a broad spectrum of language patterns and minimises the risk of biased or repetitive text generation.
- **Data Quality:** Clean and well-formatted data is essential for effective LLM training. This involves eliminating errors, inconsistencies and irrelevant information from the dataset.

The Alpaca format[54] offers a structured approach to data preparation for LLM fine-tuning. Each data point in this JSON format comprises three key fields:

- **instruction:** A concise task description guiding the model's action.
- **input:** Optional contextual information aiding the model's understanding.
- **output:** A required field providing the correct answer or output corresponding to the instruction.

The instruction field should be clear and informative, furnishing all necessary details for the model to perform the task accurately. While the input

field is optional, it can enhance the model's comprehension by providing additional context. The output field is mandatory and must supply the correct response to the instruction. Overall, the Alpaca format offers versatility, making it suitable for fine-tuning LLMs across various tasks.

3.6.4 Tokenization of Data

The LLaMA tokenizer[48] is a byte-level Byte Pair Encoding (BPE)[8] tokenizer that is used to encode text for the LLaMA LM. It is based on the SentencePiece tokenizer, but it has been modified to better handle the specific needs of the LLaMA 2 model. The LLaMA tokenizer works by splitting text into bytes and then grouping the bytes into pairs. The pairs are then merged into larger and larger units until a vocabulary of a fixed size is created. The vocabulary is then used to encode the text into a sequence of integers. The LLaMA tokenizer has a number of advantages over other tokenizers, including:

- It is able to handle a wide range of languages and character sets.
- It is able to encode rare and infrequent words.
- It is able to encode subword units, which can help to improve the performance of LLMs.

3.7 STORAGE MODULE

The storage module not only includes the database itself but also encompasses a comprehensive set of procedures designed to ensure the efficient storage and management of data within the database[23] environment. Utilising a DataBase Management System (DBMS)[23], users can execute a wide array of operations, ranging from data creation to deletion, thereby facilitating seamless data management and manipulation. Furthermore, the data extracted through web crawling techniques is systematically stored within the database, serving as a valuable resource for both model training and real-time usage.

3.7.1 Embedding Unit

An embedding[10] is a way of representing data, such as text, in a numerical format that can be understood and processed by ML algorithms. Embeddings are often used to represent text documents or queries in a high-dimensional vector space[44] where the distance between vectors reflects their semantic similarity[5]. This allows search systems to retrieve relevant results even if they do not contain exactly the same keywords as the query.

An embedding refers to the conversion of words or phrases into numerical values represented in continuous vector space as shown in Figure 3.2. Such representations reflect the semantic connections among words by positioning related ones nearby. Words with shared meanings tend to cluster closely in this vector space during training using approaches like Global Vectors (GloVe)[20] and Transformer[4], applied to sizable text collections. Consequently, finding meaningful correlations among words becomes feasible, facilitating improved performance in NLP tasks.

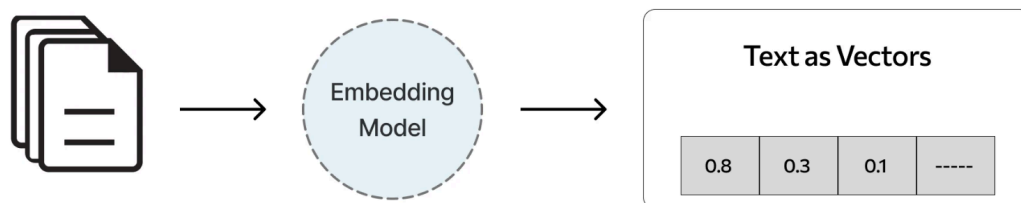


Figure 3.2 Working of an Embedding Model

3.7.2 Ranking

In the process of engaging with embeddings, the practice of ranking[35] consists of evaluating the degree of closeness amongst a specified collection of vectors, consequently establishing their relative positions within the embedding space. Diverse measurements[39], such as cosine similarity, Euclidean distance

or Manhattan distance, can serve to quantify proximity, subsequently yielding dissimilar rankings.

The utilisation of cosine similarity to discover the term most comparable to a particular animal name in an embedding space encompassing numerous animal names. Initiate by computing the cosine similarity between the particular name and every remaining vector, followed by arranging the findings in decreasing order, ultimately designating the outcome featuring the peak score signifies the optimal match corresponding to semantic affinity.

3.7.3 Reranking

Reranking[35] is the procedure of reshuffling the initial ordering produced after performing a primary ranking operation over a series of objects or elements in response to certain criteria, frequently grounded in enhanced knowledge, refined constraints or user feedback. This technique aims at improving the relevance and utility of the original list, thus offering superior matches tailored to preferences or objectives of the users.

Reranking incorporates additional factors beyond the original ranking scope, enhancing precision and accuracy. Through sophisticated calculations, reranked lists showcase heightened fidelity towards end-user expectations, boosting satisfaction levels.

The essence of reranking lies in iterative optimization cycles, wherein fresh parameters or signals get introduced, guiding successive reassessment phases. Subsequent rounds adjust the existing rankings dynamically, accounting for revised conditions and augmented intelligence, culminating in more comprehensive and nuanced arrangements. Overall, reranking amplifies the impact of decision-making processes in diverse areas, such as information retrieval, recommendations, summarization and conversational AI agents.

3.7.4 Vector Database

A vector database[44] is a specific database category designed for storing and organising vectors, which represent data objects mathematically and are also referred to as vector embeddings. These databases are finely tuned to streamline the storage, retrieval, and querying processes of vectors, focusing on their similarity or distance from other vectors. In contrast to conventional database querying methods that rely on exact matches or predetermined criteria, vector databases excel in identifying the most similar or pertinent data based on semantic or contextual significance.

3.8 MODEL CUSTOMISATION MODULE

Customising a model[36] entails adjusting an LLM to better suit a specific task or domain. This process may involve modifying the architecture of the model, training it with a different dataset or utilising fine-tuning techniques.

3.8.1 Zero-shot Prompting

Zero-shot prompting[43] refers to a technique in NLP where a LM generates responses to prompts without being explicitly trained on those prompts. In other words, the model is asked to generate responses to tasks or questions it has never seen before, without any fine-tuning or specific training on those tasks. Instead, zero-shot prompting relies on the understanding of language by the model and its ability to generalise from its training data to infer appropriate responses to novel prompts. Here are some of the potential benefits of zero-shot prompting:

- **Versatility:** Zero-shot prompting allows a model to handle a wide range of tasks without the need for task-specific training data or fine-tuning.
- **Scalability:** Zero-shot prompting models can be applied to a diverse array of tasks without the need to retrain or fine-tune the model.

- **Generalisation:** By leveraging its understanding of language and patterns learned during training, a zero-shot prompting model can generalise to unseen tasks or prompts, making it adaptable to new challenges or scenarios.

3.8.2 Fine-tuning LLMs

Fine-tuning LLMs involves adjusting their parameters to optimise performance for specific tasks. This is achieved by training the model on a relevant dataset tailored to the task at hand. The extent of fine-tuning necessary varies based on task complexity and dataset size. LLMs often yield diverse outputs for the same instruction, necessitating fine-tuning to refine their understanding of desired output formats and input analysis. Due to their extensive knowledge base, LLMs typically require minimal data for fine-tuning. This contrasts with traditional LLM training methods, which involve training on large text and code datasets. Fine-tuning offers a cost-effective and time-efficient alternative, as it enables training on smaller datasets. Supervised learning entails training the model to predict correct outputs for each input example in the dataset.

3.8.3 Full Fine-tuning

Full fine-tuning is a highly effective technique involving training the entire LLM on a task-relevant dataset[26]. Despite being computationally demanding, it often results in superior performance. When computational resources are limited, repurposing may be considered as an alternative. However, for optimal results, full fine-tuning is recommended. In this process, all parameters of a pre-trained model are updated for the specific task, in contrast to partial fine-tuning where only select parameters are adjusted. This method involves training the model with task-specific data, refining its hyperparameters, and evaluating its performance on test data. In LLaMA 2, core

model layers[51] such as the decoder, attention and MLP layers are typically fine-tuned during the full fine-tuning process, as depicted in Figure 3.3.

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaAttention(
          (q_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (v_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (o_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (up_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (down_proj): Linear4bit(in_features=11008, out_features=4096, bias=False)
          (act_fn): SiLUActivation()
        )
        (input_layernorm): LlamaRMSNorm()
        (post_attention_layernorm): LlamaRMSNorm()
      )
    )
    (norm): LlamaRMSNorm()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

Figure 3.3 LLaMA 2 Layers

A 7-billion parameter model is considered highly extensive and demands commercial Graphical Processing Unit (GPUs) such as H100 or A100. Each billion parameters necessitate 4 GB of Video Random Access Memory (VRAM) due to its utilisation of float32 precision, as depicted in Figure 3.4. This exceeds the capacity of consumer GPUs like the T4 with only 14 GB of VRAM. Relying on the CPU for numerous critical tasks leads to time inefficiencies.

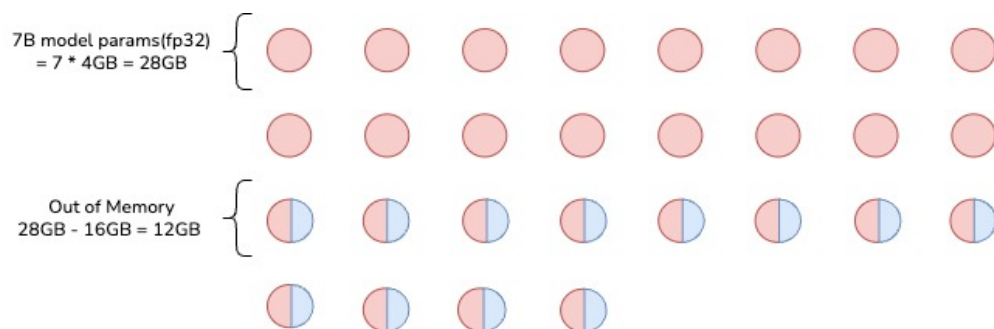


Figure 3.4 VRAM Requirement of LLaMA 2-7b

Lowering the floating-point precision of the model and employing various quantization techniques will assist in constraining the requirements of the model. The type of fine-tuning employed is elaborated further in the subsequent section.

3.9 PARAMETER EFFICIENT FINE-TUNING

Parameter Efficient Fine-Tuning (PEFT) [32] is a technique designed to adapt Pre-trained Language Models (PLMs) and LLMs to new tasks without fine-tuning all parameters of the model. PEFT methods tackle resource constraints by fine-tuning only a subset of parameters. These methods have demonstrated performance comparable to full fine-tuning. Some of the PEFT techniques include Low Rank Adaptation (LoRA) and Quantised LoRA (QLoRA). Additionally, PEFT reduces floating-point precision and aims to optimise the model size for compatibility with consumer GPUs.

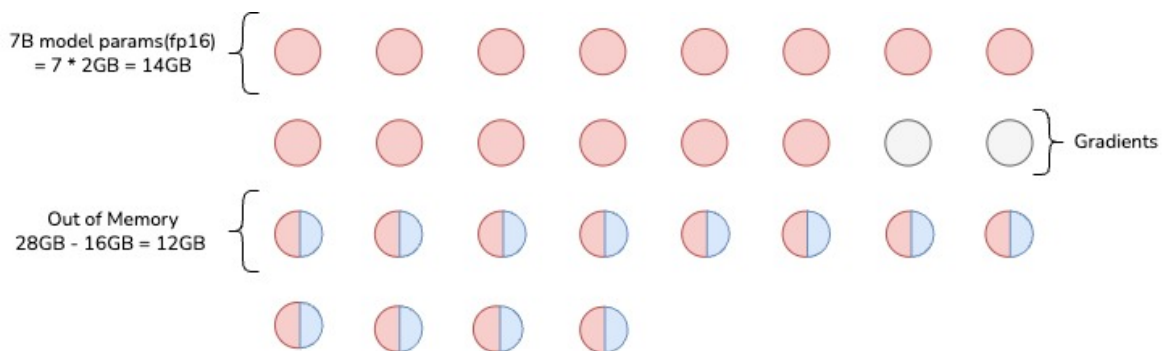


Figure 3.5 VRAM Requirement after Half Precision (fp16)

Even though the model can fit within a 14 GB capacity, the size of gradients and optimizers increases during runtime, leading to a bottleneck as illustrated in Figure 3.5.

3.9.1 Low Rank Adaptation

LoRA[19] fine-tuning presents a method for refining LLMs that is more efficient and impactful compared to traditional fine-tuning approaches.

Traditional methods entail retraining all parameters of the LLM, which can be computationally burdensome and time-intensive, particularly for exceedingly large LLMs. Conversely, LoRA fine-tuning targets only a small subset of the parameters of the LLM, rendering it notably faster and more resource-efficient. This technique operates by utilising two low-rank matrices to approximate the weight matrix of the LLM. These matrices are then fine-tuned on a task-specific dataset. Subsequently, after fine-tuning the low-rank matrices, they are integrated into the weight matrix of the LLM to tailor the LLM for the new task. LoRA fine-tuning has demonstrated effectiveness across various NLP tasks, encompassing text classification, QA and summarisation. LoRA implements a strategy where it freezes the weights of the pre-trained model and introduces trainable rank decomposition matrices into each layer of the Transformer architecture. Additionally, LoRA adopts float16 precision for its parameters, while employing int8 precision for the parameters of the LLaMA-2 model, as depicted in Figure 3.6.

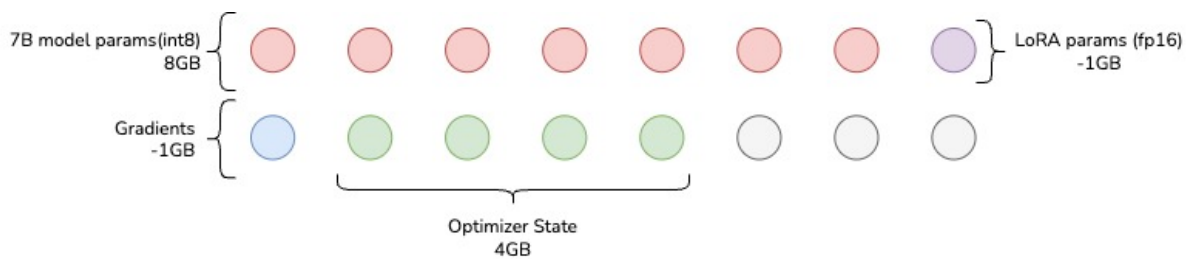


Figure 3.6 LoRA Parameters, Gradients and Optimizer State

Despite having 3GB of free VRAM, activations consume approximately 4GB of VRAM, necessitating further quantization of other parameters.

3.9.2 Quantized Low Rank Adaptation

QLoRA[9] involves propagating gradients through a frozen, 4-bit quantized pretrained LM into LoRA. After confirming that 4-bit QLoRA matches 16-bit performance consistently across various scales, tasks, and

datasets, a comprehensive investigation of instruction fine-tuning is conducted on the largest open-source LMs available for research. The performance of instruction fine-tuning on these models is assessed, as illustrated in Figure 3.7.

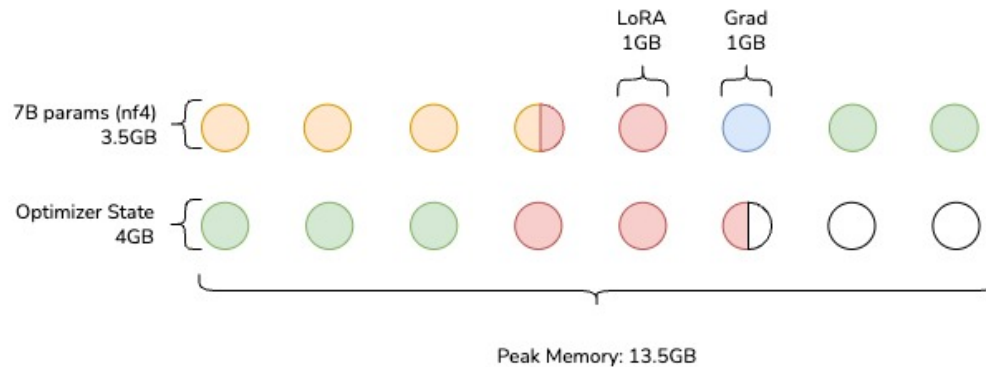


Figure 3.7 QLoRA Parameters, Gradients and Optimizer State

Finally, the quantized configuration is applied to load the model, and the Alpaca format is supplied to the model for tokenization by the tokenizer. QLoRA introduces additional layers while maintaining the frozen status of the existing model layers. Dropout layers[9] serve as a regularisation technique in ML to mitigate overfitting. These layers operate by randomly deactivating neurons within the network during training, compelling the network to rely on a broader array of neurons for task execution. The dropout rate, a hyperparameter, governs the proportion of neurons deactivated. Typically, a dropout rate between 0.5 and 0.8 is recommended for hidden layers, with input layers utilising a higher dropout rate, such as 0.8. Dropout layers have demonstrated considerable efficacy in averting overfitting across diverse ML tasks, including image classification, NLP and machine translation.

PeftModelForCausalLM offers a user-friendly interface for managing causal LMs in Python. It enables users to effortlessly create and manipulate causal graphs, train personalised causal models, and execute diverse causal inference tasks[5]. Figure 3.8 showcases certain layers of LLaMA 2 post fine-tuning.

```

PeftModelForCausalLM(
  (base_model): LoraModel(
    (model): LlamaForCausalLM(
      (model): LlamaModel(
        (embed_tokens): Embedding(32000, 4096)
        (layers): ModuleList(
          (0-31): 32 x LlamaDecoderLayer(
            (self_attn): LlamaAttention(
              (q_proj): Linear4bit(
                in_features=4096, out_features=4096, bias=False
                (lora_dropout): ModuleDict(
                  (default): Dropout(p=0.05, inplace=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (k_proj): Linear4bit(in_features=4096, out_features=4096,
bias=False)
              (v_proj): Linear4bit(
                in_features=4096, out_features=4096, bias=False
                (lora_A): ModuleDict(
                  (default): Linear(in_features=4096, out_features=8,
bias=False)
                )
                (lora_B): ModuleDict(
                  (default): Linear(in_features=8, out_features=4096,
bias=False)
                )
                (lora_embedding_A): ParameterDict()
                (lora_embedding_B): ParameterDict()
              )
              (o_proj): Linear4bit(in_features=4096, out_features=4096,
bias=False)
              (rotary_emb): LlamaRotaryEmbedding()
            )
            (mlp): LlamaMLP(
              (gate_proj): Linear4bit(in_features=4096, out_features=11008,
bias=False)
              (up_proj): Linear4bit(in_features=4096, out_features=11008,
bias=False)
              (down_proj): Linear4bit(in_features=11008, out_features=4096,
bias=False)
              (act_fn): SiLUActivation()
            )
            (input_layernorm): LlamaRMSNorm()
            (post_attention_layernorm): LlamaRMSNorm()
          )
        )
        (norm): LlamaRMSNorm()
      )
      (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
    )
  )
)

```

Figure 3.8 Layers of Fine-tuned LLaMA 2

3.10 OUTPUT MODULE

The output module of the chatbot for website engagement serves as a pivotal component, ensuring seamless interaction between users and the system. It serves as the gateway through which the chatbot delivers responses, ensuring an intuitive and engaging user experience. This module encompasses a range of functionalities aimed at enhancing the effectiveness and versatility of the output. From dynamically generating responses to formatting and presenting information in a user-friendly manner, the output module plays a vital role in shaping the overall interaction between users and the chatbot.

3.10.1 Retrieval Augmented Generation

RAG is a technique that enhances the accuracy and reliability of generative AI models by incorporating facts fetched from external sources. This method is designed to address the limitations of LLMs by integrating neural information retrieval with neural text generation. The retrieval component of RAG acts as a high-powered reading module, rapidly indexing relevant data from vast knowledge stores, while the generation component serves as a creative writing module, synthesising key information from retrieved passages into coherent narratives. The architecture allows RAG systems to overcome the limitations of LLMs, such as Generative Pre-trained Transformer (GPT-4)[41], which cannot explicitly access external datasets within their foundation models. By offloading content ingestion to specialised standalone retrieval modules, generators can focus on mastering linguistic dexterity. The approach significantly expands the reachable information compared to what can be stored internally only, enabling RAG to provide more accurate and nuanced responses. Additionally, RAG mitigates harmful model biases by exposing generators to more diverse perspectives through broader content retrieval and it enables genuine fact-checking against original sources. The method moves closer to

how humans leverage knowledge acquisition to unlock reasoning and communication superpowers far beyond our biological programming.

3.10.2 Content Delivery

Markdown[30] in LLMs is to structure and format text, enhancing the readability and presentation of generated content. This use of Markdown allows for the creation of documents with headings, lists, links, and other formatting elements that can be easily interpreted and displayed in various platforms. The integration of Markdown with LLMs, such as GPT-4 and Llama 2, is particularly beneficial for tasks requiring the generation of structured content. By incorporating Markdown syntax into the text generation process, LLMs can produce outputs that are not only coherent and contextually relevant but also well-organised and visually appealing. This integration facilitates the creation of documents, guides, and other forms of content that are easier to navigate and understand, thereby improving the overall user experience. Moreover, the use of Markdown in conjunction with LLMs enables the generation of content that can be seamlessly integrated into existing documentation systems, websites, and other digital platforms. This integration simplifies the process of content creation and editing, making it more accessible to a wider range of users.

Streaming response[14] in LLMs is implemented to enhance user experience by delivering content in real-time, reducing the wait time for users. This approach is particularly beneficial for applications that generate text, where waiting for the entire generation process to complete can lead to user drop-off. The streaming response mechanism allows for the immediate display of results to users, making the application feel more responsive and interactive. There are multiple methods to implement streaming responses, including polling, Server-Sent Events (SSE)[38] and WebSockets. Polling is considered the simplest approach for applications that do not require real-time bi-directional

communication, such as text or code generation applications. SSE is another method that involves sending server-sent events from the server to the client, allowing for the streaming of generated tokens to the UI. WebSockets, on the other hand, allow for a persistent bi-directional communication channel between the client and server, making them suitable for applications that require real-time transfer of packets from both ends. However, for applications using LLMs that primarily generate text or code, WebSockets might be considered overkill due to the lack of need for real-time bi-directional communication. The implementation of streaming responses in LLM applications involves receiving the streaming response from LLMs, delivering the streaming response to the client-side, and receiving the streaming response on the client side. The choice of method depends on the specific requirements of the application, such as the need for real-time updates and the complexity trade-off between real-time functionality and implementation complexity.

CHAPTER 4

IMPLEMENTATION

4.1 DATA COLLECTION AND FORMATTING

Collection and preparation of the data that will be used to train a LM is part of data selection and formatting. This process is important because the quality and format of the data can have a significant impact on the performance of the model.

4.1.1 Data Collection

All the sub links are generated through web scraping. The curated URLs are fetched, and a DAG is created from them, followed by the execution of the PageRank algorithm. Filtered URLs are fetched, and web crawling is performed to retrieve the content of webpages using Cheerio.

```
const response = await axios.get(url);
const $ = cheerio.load(response.data);
const links: string[] = [];
$('a').each((_, element) => {
    let link = $(element).attr('href');
    if (link) {
        link = removeTrailingSpecialChars(link).trim();
        if (!isDocumentLink(link) && isValidURL(link) &&
link.trim().includes(baseUrl.trim()))
            links.push(link);
    }
});
```

4.1.2 Data Wrangling

The web crawled data is obtained as raw HTML. Data wrangling is performed by removing HTML elements, CSS styling and JavaScript logics. The processed data is then obtained to be stored in the database.

```
function isDocumentLink(link: string) {  
    return link.endsWith('.pdf') || link.endsWith('.doc') ||  
    link.endsWith('.docx');  
}  
  
function removeTrailingSpecialChars(inputString: string) {  
    const regex = /^[^\w\s]+$/;  
    return inputString.replace(regex, '');  
}  
  
import { getHTMLData } from '$lib/helper';  
const contentStr = await getHTMLData($);
```

4.1.3 Alpaca Formatting

By implementing Alpaca formatting on the data, the input data is organised and standardised, thereby improving the accuracy and efficiency of the model. This approach ensures data uniformity, ultimately enhancing the performance of the model during both training and response generation stages.

```
{  
    url: "https://www.svce.ac.in/departments/information-techn  
ology/?page=profile",  
    content: `In 1996, Sri Venkateswara College of Engineering  
pioneered the introduction of the B.Tech degree programme in  
Information Technology under the affiliation of University of  
Madras.`,  
    keywords: "information-technology departments" },
```

4.2 DATA STORE FOR MODEL

Cassandra Vector Store is a database system built on Apache Cassandra, designed specifically for storing and querying high-dimensional vector data efficiently. The system supports storing vectors along with associated metadata across a cluster of Cassandra nodes, providing seamless scalability and fault tolerance. Cassandra Vector Store offers efficient indexing and querying mechanisms optimised for vector data, facilitating fast retrieval of similar vectors or nearest neighbours.

```
import cassandra

import cassio

from llama_index.vector_stores.cassandra import
CassandraVectorStore

cassio.init(
    database_id="<DATABASE_ID>",
    token="<TOKEN>", keyspace="<KEYSPACE_NAME>",
)

cassandra_store = CassandraVectorStore(
    table="new_table",
    embedding_dimension=<EMBEDDING_DIMENSION>,
)

storage_context =
StorageContext.from_defaults(vector_store=cassandra_store)
```

4.2.1 Embedding using Voyage Large 2

The processed data is partitioned into data chunks and subsequently inputted into voyage-large-2 embeddings, facilitating comprehension by the model. These data chunks are segregated by the LlamaIndexTextSplitter to

enhance the understanding of the provided content. The embeddings subsequently generate vectors, which are then stored within the Cassandra Vector Store for performing semantic search.

```
from llama_index.llms.ollama import Ollama

from llama_index.embeddings.langchain import
LangchainEmbedding

from langchain_community.embeddings import VoyageEmbeddings

lc_embed = VoyageEmbeddings(
    model="voyage-large-2",
    voyage_api_key="<VOYAGE_API_KEY>"
)

embed_model = LangchainEmbedding(
    lc_embed, callback_manager=callback_manager
)

llm = Ollama(
    model="aish/svce-ai",
    request_timeout=60.0,
    base_url="<PORT_FORWARDED_BASE_URL>",
    temperature=0.2,
    context_window=8192,
    callback_manager=callback_manager,
    max_new_tokens=6072
)
```

4.2.2 Indexing using Pagerank

The indexing process begins with the instantiation of a VectorStoreIndex object. This instantiation involves the utilisation of a set of components that are

configured to enable efficient indexing operations for subsequent semantic search functionalities. These components are:

- The storage context, which refers to the environment or configuration settings necessary for interacting with the storage system.
- A vector store, which is the cassandra vector store specifically designed to store high-dimensional vector data efficiently.
- The LLaMA-2 model, which plays a role in processing and understanding textual data, possibly in conjunction with the embedding model.
- The embedding model, which serves as the component responsible for generating vector representations of the input data, which are subsequently stored in the vector store for indexing and semantic search.

```
from llama_index.core import VectorStoreIndex
index = VectorStoreIndex.from_vector_store(
    storage_context = storage_context,
    vector_store = cassandra_store,
    llm = llm,
    embed_model = embed_model
)
```

4.3 MODEL CONFIGURATION

Model configuration is the process of setting up the parameters of a ML model to define its architecture, behaviour and performance characteristics. This process involves specifying various hyperparameters such as the number of layers, activation functions, optimizer settings and the learning rate. This process involves the selection of pre-trained weights or embeddings, fine-tuning strategies and input data preprocessing steps. The configured model is then ready for evaluation and deployment as a conversational agent.

4.3.1 Fine-tuning Model

The below steps illustrate the process flow for model implementation.

Step 1: Common dependencies for the entire model.

```
!pip install -q -U torch bitsandbytes einops fastapi[all]  
uvicorn python-multipart pydantic pinecone-client  
!pip install -q -U peft transformers accelerate langchain  
xformers sentencepiece
```

The packages being imported for model implementation are as explained.

- bitsandbytes: A Python library that offers various utilities for bitwise operations and conversions between different data types.
- transformers: A library by Hugging Face for NLP models.

Step 2: Importing transformers and LangChain for text transformation with the quantised model.

```
from peft import PeftModel, PeftConfig  
from transformers import AutoModelForCausalLM, AutoTokenizer,  
pipeline, BitsAndBytesConfig
```

The classes and functions being imported for model implementation are:

- PeftModel: A class that loads and instantiates a PEFT model.
- AutoTokenizer: A class that loads and instantiates a tokenizer class.

Step 3: Configure the LLaMA model which is optimised using PEFT. The LLaMA model is instantiated using the AutoModelForCausalLM class. This model is initialised with pre-trained weights along with configuration.

```
model1 = AutoModelForCausalLM.from_pretrained(  
    "meta-llama/Llama-2-7b-hf", config = config,  
    quantization_config = bnb_config)  
model1 = PeftModel.from_pretrained(model1, "aish/svce-ai")  
tokenizer = AutoTokenizer.from_pretrained(  
    "meta-llama/Llama-2-7b-hf")
```

Training loss for the model training process is as given in Figure 4.1.

Validation feature	output
Validation metric	loss
Best model step	4795
Best model epoch	2
Best model's validation loss	0.20967940986156464
Best model's test loss	0.20638065040111542



Figure 4.1 Training Loss Graph of Model Training

4.3.2 Prompt Engineering

The model is being provided with a `system_condense_prompt` to generate a condensed question, which translates the meaning of the user's query into a comprehensible format for the model. This condensed question incorporates expansions of any abbreviations present in the user's query and provides contextual information to facilitate more effective and efficient model responses. Subsequently, the response generated for the user is based on the condensed question rather than the original user query.

```
SYSTEM_PROMPT_CONDENSE = """
```

```
You are an assistant for Sri Venkateswara College of Engineering (SVCE) called SVCE AI, given the below conversation create a condense question for the given query, based on prioritisation of the conversation from 1 to 10, 1 being the highest.
```

```
Expand the abbreviation with regard to a college in the question for sure, if any.
```

```
The Condense question should have the same meaning as the given question.
```

```
And Do not answer the question.
```

```
Conversation: {questions}
```

```
"""
```

The model is now provided with the condensed question and a `system_prompt` to generate a response to be displayed to the user. The response generated is based on the condensed question, incorporating context and expansions of abbreviations to enhance comprehension. The model plays the role of the SVCE AI, while the one who asks the question is referred to as the user. Additionally, the model is instructed not to engage in hallucination or generate responses beyond the scope of the provided context and condensed question.

```
SYSTEM_PROMPT =(
    "You are a chatbot for Sri Venkateswara College of Engineering
    (SVCE) called SVCE AI developed by students of SVCE, able to
    have normal interactions, as well as talk"

    "Please stick on to the given context and think twice before
    you answer any question. "

    "Here are the relevant documents for the context:\n\n"

        f"{self.context_str}"

    "\n\nDon't blabber wrong answers outside the context. "

    "If a list is being returned, return the complete exhaustive
    list instead of saying excuses for not providing complete
    answers. "

    "DO NOT HALLUCINATE AT ANY POINT OF TIME. STICK TO THE CONTEXT
    AS MUCH AS POSSIBLE "

    "Always start answering by saying SVCE AI: "
)
```

4.4 RESPONSE GENERATION

Response generation refers to the process where the AI model generates a text-based response to a given input or query provided by the user. This response generation typically involves NLP techniques, where the model analyses the input text, understands its context and semantics and produces a

coherent and contextually relevant output that addresses the input query effectively. The goal of response generation in SVCE AI is to provide accurate, informative and engaging responses that mimic human-like conversation and effectively fulfil the user's informational or interactive needs.

4.4.1 Question Condensation

The code defines methods for condensing prompts and retrieving relevant nodes from the vectorstore. A system prompt is generated based on the set of questions asked previously by the user. A condensed engine is created using a SimpleChatEngine initialised with a LLM, service context and the generated prompt. This prompt is then used to generate a response to the condensed question.

```
def condense_prompt(self):  
    SYSTEM_PROMPT_CONDENSE = prompt_condense(self.questions)  
    condense_engine = SimpleChatEngine.from_defaults(  
        llm=condenseLLM,  
        service_context=service_context,  
        system_prompt= SYSTEM_PROMPT_CONDENSE,  
    )  
    condense_response = condense_engine.chat("Given  
Question: " + self.question)  
    self.question_con =  
remove_special_characters_except_dot_and_question(self.question_con)
```

A retriever is initialised based on an index, with parameters set for similarity threshold. The method retrieves relevant nodes based on the condensed question and stores them for further processing. The retrieved nodes are converted to a string format and stored.

```
def retrieve_nodes(self):
    retriever = index.as_retriever(similarity_top_k=5)
    self.nodes = retriever.retrieve(self.question_con)
    for node in self.nodes:
        str_node.add(node.text.strip())
    self.context_str = ("\n\n".join(list(str_node)))
```

4.4.2 Chat Engine

The code orchestrates the conversational flow between the user and the chatting assistant within the system. Initially, the method invokes functions to condense the question and retrieve relevant nodes. The assistant then generates response to the condensed question. Finally, the method updates the chat history with the question and the response before returning the generated answer.

```
def chatter(self, question):
    self.condense_prompt()
    self.retrieve_nodes()
    chat_engine = SimpleChatEngine.from_defaults(
        llm=llm, verbose=True, nodes=self.nodes,
        memory=self.chat_memory_buff, system_prompt=(),
        callback_manager=callback_manager)
    answer = chat_engine.stream_chat(self.question_con)
    self.custom_chat_history.extend([
        ChatMessage(role=MessageRole.USER,
                    content = self.question_con),
        ChatMessage(role=MessageRole.ASSISTANT,
                    content=str(answer))])
    return answer
```

4.5 HOSTING THE MODEL

The deployment and maintenance of the AI model in a production environment where it can interact with users and provide responses to their inputs is done as part of hosting the model. This process involves setting up the necessary infrastructure, such as servers or cloud services, to run the model, ensuring its availability and scalability to handle user requests effectively. Hosting also entails managing dependencies and handling updates or modifications to the model as needed.

4.5.1 Model Provisioning

Model provisioning involves the process of deploying the AI model to the Ollama platform, which enables local execution of the model. The model is pushed to the Ollama platform, as shown in Figure 4.2, allowing for seamless integration and execution within the local environment. This provisioning step ensures that the model is readily available and accessible for deployment, facilitating efficient usage and interaction with users.



Figure 4.2 Model Deployment in Ollama Platform

4.5.2 Model Utilisation

The code initialises an Application Programming Interface (API) based application with CORS middleware configured to allow cross-origin requests. It defines a POST endpoint '/chat' that expects a JSON payload containing a question field. Upon receiving a request, the endpoint invokes the function which processes the question. The resulting response is streamed back to the client as a response. Additionally, the code sets up a ngrok tunnel to expose the FastAPI application to the internet, allowing external access. Finally, the server is run to serve the application.

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import StreamingResponse
import nest_asyncio, uvicorn
from pyngrok import ngrok

!pyngrok authtoken <AUTH_TOKEN>

app = FastAPI()
app.add_middleware(
    CORSMiddleware, allow_origins = ["*"], allow_credentials = True
)

@app.post('/chat')
async def home(item: Item):
    return StreamingResponse(
        generate_data(chat.chatter(item.question))
    )

ngrok_tunnel = ngrok.connect(8000)
nest_asyncio.apply()
uvicorn.run(app, port=8000)
```

4.6 RESPONSE PRESENTATION

Response presentation plays a crucial role in delivering clear and structured interactions between the model and the users. Markdown is utilised to ensure proper alignment and readability of responses and streaming response is employed to facilitate real-time delivery of responses.

4.6.1 Response Alignment using Markdown

Markdown provides a simple and intuitive syntax for formatting text, making it easier for developers to write and maintain structured content. The use of Markdown offers significant benefits, particularly in displaying code or list items within web applications or documents. When displaying code snippets, Markdown preserves the formatting and syntax highlighting, enhancing readability and understanding for both developers and users. Additionally, Markdown enables the creation of well-organised lists, allowing for clear and concise presentation of information.

The code snippet imports the MarkdownIt library and creates an instance. Subsequently, it utilises this instance to render Markdown-formatted content stored in the variable. The rendered HTML is inserted into the document using the {@html} directive. This process facilitates the dynamic rendering of Markdown content within a web application.

```
import MarkdownIt from 'markdown-it';  
const md = new MarkdownIt();  
<p>{@html md.render(chat.message)}</p>
```

4.6.2 Streaming Response

The utilisation of streaming response engines is pivotal in the context of chatbots and chat engines. It allows for the efficient transmission of data in

real-time, enabling seamless communication between the chatbot and the user. This real-time interaction is essential for maintaining the conversational flow and providing timely responses to user queries.

The code defines a generator function which iterates over the response generator yielded by a StreamingResponse object. For each text element yielded by the response generator, the function yields the text itself. This approach enables the iterative retrieval and processing of text data from the streaming response, facilitating the generation of data in a sequential manner. By utilising a generator function, the code efficiently handles large volumes of text data without consuming excessive memory resources, ensuring optimal performance and scalability.

```
def generate_data(streaming_response):  
    for text in streaming_response.response_gen:  
        yield text
```

4.7 USER INTERFACE FOR CONVERSATIONAL AGENT

The user interface for the conversational agent serves as a crucial gateway for users to interact with the AI system. Designed to prioritise simplicity and intuitiveness, the UI enables seamless communication between users and the AI model. Users can effortlessly input queries or commands, receive responses and navigate through conversation threads. The UI also facilitates the presentation of information in a clear and organised manner, ensuring that users can easily comprehend and interact with the response generated. Additionally, the UI of the conversational agent is designed to adapt to various screen sizes and devices, ensuring a consistent and optimal user experience across desktops and mobile devices. By employing responsive design principles, the interface seamlessly adjusts its layout and functionality to

accommodate different viewing environments, enhancing accessibility and usability for a diverse user base. A snapshot of the interface in desktop and mobile views is depicted in Figure 4.3.

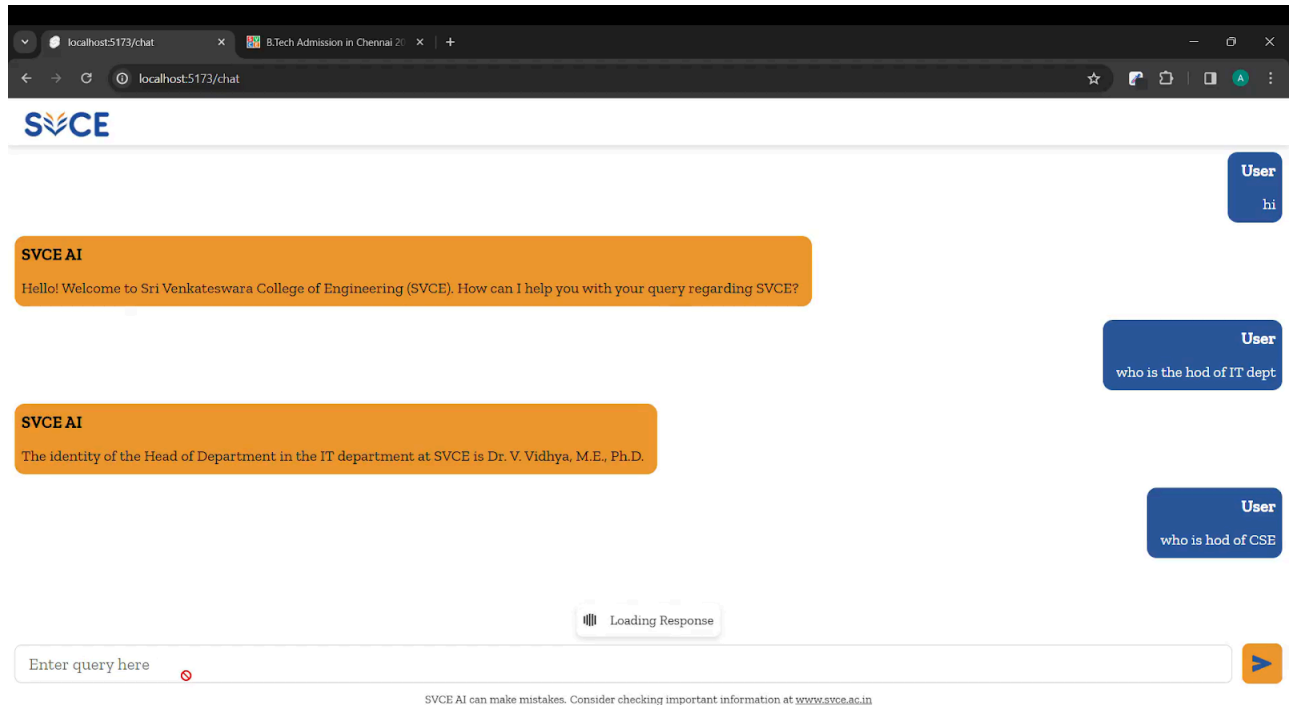


Figure 4.3 Screenshot of the User Interface

4.7.1 Frontend Development using Svelte.js

Svelte.js is a modern JavaScript framework used for building UIs and web applications. Unlike traditional frontend frameworks like React.js or Vue.js, Svelte.js shifts much of the work to compile time, resulting in highly optimised and efficient code at runtime. It introduces a novel approach called compilation at build time, where Svelte.js compiles components into highly optimised JavaScript code during the build process. This approach eliminates the need for a virtual DOM and reduces the runtime overhead, resulting in smaller bundle sizes, faster load times and improved performance compared to other frameworks. Svelte.js emphasises simplicity, reactivity and code efficiency, making it an increasingly popular choice for frontend development projects.

Svelte.js is employed in this project to craft a UI that enhances the interaction between users and the model. This framework integrates seamlessly with backend services and AI models, enabling efficient communication and data exchange between the frontend and backend components of the SVCE AI system. This enhances the reliability and predictability of the frontend components, contributing to a more stable and seamless user experience. Overall, frontend development using Svelte.js empowers SVCE AI to deliver a user-friendly and interactive interface that enhances user engagement and satisfaction.

```
<div class="h-full w-full flex flex-col">

  {#each chats as chat, index}

    <div class={'flex w-full p-2 ' + (index % 2 === 0 ?
'justify-end ' : 'justify-start ')}>

      <div class={'flex flex-col w-fit max-w-96 p-2
gap-3 rounded-xl ' + (index % 2 === 0 ? 'bg-[#2b579a]
text-white items-end pl-4' : ' bg-[#EE972E] text-black
items-start pr-4')}> style="max-width: {maxWidth}px;">

        <div class="text-xl font-bold">

          <p>{chat.role}</p>

        </div>

        <p class={'text-lg ' + (index % 2 === 0 ?
'text-right' : 'text-left')}>

          {@html md.render(chat.message)}

        </p>

      </div>

    </div>

  {/each}

</div>
```


4.7.2 Tailwind CSS for Styling

Tailwind CSS is a utility-first CSS framework that streamlines the process of styling web applications by providing a comprehensive set of pre-defined utility classes. Unlike traditional CSS frameworks that come with predefined components and styles, Tailwind CSS focuses on providing low-level utility classes that can be composed together to create custom designs. This approach offers developers greater flexibility and control over their stylesheets while reducing the need for writing custom CSS code. With Tailwind CSS, developers can rapidly build and style responsive, modern web applications without sacrificing flexibility or performance.

4.8 METRICS

The performance of the project is evaluated based on the following metrics:

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) is a tool for evaluating how well machines can summarise or translate text. It compares the output of a machine to the output of a human and gives a score based on how similar they are.

```
import evaluate
import json
rouge = evaluate.load('rouge')
eresults = rouge.compute(predictions=predictions,
references=references)
print(json.dumps(eresults, indent=4))
```

```
{
  "rouge1": 0.8031746031746031,
  "rouge2": 0.6818181818181818,
  "rougeL": 0.726984126984127,
  "rougeLsum": 0.726984126984127
}
```

Bilingual Evaluation Understudy (BLEU) metrics are a set of metrics for evaluating machine translation output. They are based on the N-gram precision of the machine translation output.

```
import evaluate
bleu = evaluate.load("bleu")
results = bleu.compute(predictions=predictions,
references=references)
```

```
{
  "bleu": 0.8161224696270971,
  "precisions": [
    0.8913043478260869,
    0.8426966292134831,
    0.7906976744186046,
    0.7469879518072289
  ],
  "brevity_penalty": 1.0,
  "length_ratio": 1.0222222222222221,
  "translation_length": 92,
  "reference_length": 90
}
```

F1 score is a ML metric that measures a model's accuracy. It is calculated as depicted in Equation 5.1. It combines the precision and recall scores of a model.

$$F1 = 2 * (Bleu * Rouge) / (Bleu + Rouge) \quad (5.1)$$

```
f1 = 2 * (results["bleu"] * eresults["rouge1"]) /
(results["bleu"] + eresults["rouge1"])
print("F1 Score: ", f1)
```

```
F1 Score: 0.920959677096473
```

The summary of all metrics is given in Table 4.1.

Table 4.1 Summary of Metrics

ROGUE1	ROGUE2	ROGUEL	BLEU	PRECISION	F1 Score
0.80317	0.68181	0.77698	0.81612	0.89130	0.92095

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

The development of the Conversational Agent for Website Engagement using LLaMA 2 marks a significant advancement in enhancing user interaction and engagement within the digital landscape. By leveraging the powerful capabilities of LLaMA 2, coupled with intuitive UI design and seamless integration with backend services, the project has successfully created a dynamic platform for fostering natural and context-aware conversations on websites. Through careful implementation of advanced AI techniques, such as response alignment using Markdown and streaming response mechanisms, the conversational agent offers a user-friendly and interactive experience, empowering users to effortlessly navigate and engage with website content. Moving forward, continuous refinement and iteration based on user feedback will be pivotal in further enhancing the capabilities of the conversational agent and ensuring its continued effectiveness in facilitating meaningful interactions and driving user satisfaction.

5.2 FUTURE SCOPE

In addition to its current milestones, the project holds immense potential for further innovation and impact in the digital sphere. As the project continues to evolve, exploring avenues for deeper integration with emerging technologies such as AR and VR could unlock new dimensions of immersive user experiences. Furthermore, proactive engagement strategies, such as real-time sentiment analysis and proactive content recommendations, can enhance user

satisfaction by anticipating and addressing their needs even before they articulate them.

Moreover, leveraging the vast amounts of data generated through user interactions can fuel advancements in NLP and sentiment analysis, enabling the conversational agent to develop a deeper understanding of user preferences, sentiments and intents. This holistic understanding can inform personalised content recommendations, tailored user experiences and targeted marketing initiatives, driving user engagement and loyalty to unprecedented levels.

As the digital landscape continues to evolve, the Conversational Agent for Website Engagement using Llama 2 stands poised to lead the way in redefining user engagement paradigms and setting new standards for digital experiences. With a commitment to continuous innovation, user-centric design and ethical AI practices, the project is primed to shape the future of digital engagement and elevate the relationship between users and digital content to unprecedented heights.

REFERENCES

1. Alzubaidi L, Zhang J, Humaidi A.J. (2021), ‘Review of deep learning: concepts, CNN architectures, challenges, applications, future directions J Big Data’, pp. 53.
2. Bateman, J. A, Hois J, Ross R, & Tenbrink T. (2010), ‘A linguistic ontology of space for natural language processing’ – Artificial Intelligence, 174(14), pp. 1027–1071.
3. Bhardwaz S., & Godha R. (2023), ‘Svelte.js: The Most Loved Framework Today’ – 2nd International Conference for Innovation in Technology (INOCON), pp. 1-7.
4. B. N. D. Santos, R. M. Marcacini, S. O. Rezende (2021), ‘Multi-Domain aspect extraction using bidirectional encoder representations from transformers’ – IEEE Access Vol. 9, pp. 91604–91613.
5. Chandrasekaran, D., & Mago, V. (2021), ‘Evolution of Semantic Similarity—A Survey’ – ACM Computing Surveys, 54(2), pp. 1–37.
6. Chang, Y. (2023), ‘A survey on evaluation of large language models’ – arXiv.org, Computer Science, Computation and Language.
7. Chiara Valentina Misischia, Flora Poecze, Christine Strauss (2022), ‘Chatbots in customer service: Their relevance and impact on service quality’ – Procedia Computer Science, vol. 201, pp. 421-428, ISSN 1877-0509.
8. David Vilar and Marcello Federico (2021), ‘A Statistical Extension of Byte-Pair Encoding’ – Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT), pp. 263–275, Bangkok, Thailand. Association for Computational Linguistics.
9. Dettmers T., Pagnoni A., Holtzman A., & Zettlemoyer L. (2023), ‘QLORA: Efficient Finetuning of Quantized LLMS’ – arXiv.org, Computer Science, Machine Learning.

10. Dhingra B., Shallue C.J., Norouzi M., Dai, A.M. & Dahl G.E. (2018), 'Embedding Text in Hyperbolic Spaces'.
11. E. Uzun (2020), 'A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages' – IEEE Access, vol. 8, pp. 61726-61740.
12. Gardner P. A., Maffeis S., & Smith G. D. (2012), 'Towards a program logic for JavaScript' – Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL.
13. Geneves P., Layaida N., & Quint V. (2012), 'On the analysis of cascading style sheets' – Proceedings of the 21st International Conference on World Wide Web - WWW.
14. Gomes H. M., Read J., Bifet A., Barddal J. P., & Gama J. (2019), 'Machine learning for streaming data' – ACM SIGKDD Explorations Newsletter, 21(2), pp. 6–22.
15. Grant-Muller S. M, Gal-Tzur A., Kuflik T., Minkov E., Shoor I. & Nocera S. (2015). 'Enhancing transport data collection through social media sources: methods, challenges and opportunities for textual data' – IET Intelligent Transport Systems, 9(4), pp. 407–417.
16. Guha R. (2023), 'Fine-Tuning humans for LLM projects' – Social Science Research Network.
17. Harinder Hari, Radha Iyer & Brinda Sampat (2022), 'Customer Brand Engagement through Chatbots on Bank Websites – Examining the Antecedents and Consequences' – International Journal of Human-Computer Interaction, vol. 38:13, pp. 1212-1227.
18. Hernández I., Rivero C. R. & Ruiz D. (2018), 'Deep Web crawling: a survey' – World Wide Web.
19. Hu E. J. (2021), 'LORA: Low-Rank adaptation of Large Language Models' – arXiv.org, Computer Science, Computation and Language.
20. Jeffrey Pennington, Richard Socher, and Christopher Manning (2014), 'GloVe: Global Vectors for Word Representation' – Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543.

21. Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, Yaodong Yang (2023), ‘Safe RLHF: Safe Reinforcement Learning from Human Feedback’.
22. Keivalya Pandya, Mehfuza Holia (2023), ‘Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations’ – arXiv.org, Computer Science, Computation and Language.
23. Khan F. A, Jamjoom M, Ahmad A, & Asif M. (2019), ‘An analytic study of architecture, security, privacy, query processing, and performance evaluation of databases-as-a-service’ – Transactions on Emerging Telecommunications Technologies.
24. Khder M.A. (2021), ‘Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application’ – International Journal of Advances in Soft Computing and its Applications.
25. Khurana D. Koli, A. Khatter, K. et al (2023), ‘Natural language processing: state of the art, current trends and challenges’ – Multimed Tools Appl 82, pp. 3713–3744.
26. Kirk J. R. (2022), ‘Improving language model prompting in support of semi-autonomous task learning’ – arXiv.org, Computer Science, Machine Learning.
27. Lamberti F., Sanna A., & Demartini C. (2009), ‘A Relation-Based Page Rank Algorithm for Semantic Web Search Engines’ – IEEE Transactions on Knowledge and Data Engineering, 21(1), pp. 123–136.
28. Lean Yu, Shouyang Wang & Lai K. K. (2006), ‘An integrated data preparation scheme for neural network data analysis’ – IEEE Transactions on Knowledge and Data Engineering, 18(2), pp. 217–230.
29. Li G., Zhang Y., Jiang X., Bai S., Peng G., Wu K., & Jiang Q. (2013), ‘Evaluation of the ArcCHECK QA system for IMRT and VMAT verification’ – Physica Medica, 29(3), pp. 295–303.
30. Loh E., Khandelwal J., Regan B., & Little D.A. (2022), ‘Prometheus: An End-to-End Machine Learning Framework for Optimizing Markdown in Online Fashion E-commerce’ – Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

31. Malini M. Patil, Basavaraj N. Hiremath (2018), 'A Systematic Study of Data Wrangling' – International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.1, pp.32-39.
32. M. H. Zahweh (2023), 'Empirical Study of PEFT techniques for Winter Wheat Segmentation' – arXiv.org, Computer Science, Computer Vision and Pattern Recognition.
33. Muhammad, A. F, Susanto, D, Alimudin, A, Adila, F, Assidiqi, M. H, & Nabhan, S. (2020), 'Developing English Conversation Chatbot Using Dialogflow' – 2020 International Electronics Symposium (IES).
34. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela (2021), 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks' – arXiv.org, Computer Science, Computation and Language.
35. Pei C., Ou W., Pei D., Zhang Y., Sun, F Ge, J. (2019), 'Personalized re-ranking for recommendation. Proceedings of the 13th ACM Conference on Recommender Systems' - RecSys '19.
36. Pereira Detto S., Santos E. A. P., Panetto H., Loures E. D., Lezoche M. & Cabral Moro Barra C. (2019), 'Applying process mining and semantic reasoning for process model customisation in healthcare' – Enterprise Information Systems.
37. Pezoa F., Reutter J. L, Suarez F., Ugarte M. & Vrgoč, D. (2016). 'Foundations of JSON Schema' – Proceedings of the 25th International Conference on World Wide Web - WWW '16.
38. Prateek Rawat, Archana N. Mahajan (2020), "ReactJS: A Modern Web Development Framework" – International Journal of Innovative Science and Research Technology ISSN No:-2456-2165, Volume 5, Issue 11.
39. Qian G., Sural S., Gu Y. & Pramanik S. (2004), 'Similarity between Euclidean and cosine angle distance for nearest neighbor queries' – Proceedings of the 2004 ACM Symposium on Applied Computing - SAC.
40. Rekha M., K. P. M, K. Selvi and R. Karthiga, "CRUD application using ReactJS hooks," EAI Endorsed Transactions on Internet of Things, vol. 10, Mar. 2024, doi: 10.4108/eetiot.5298.

41. Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, Tie-Yan Liu (2022), ‘BioGPT: generative pre-trained transformer for biomedical text generation and mining, Briefings in Bioinformatics’.
42. Sakr S., Liu A., & Fayoumi A. G. (2013), ‘The family of mapreduce and large-scale data processing systems’ – ACM Computing Surveys, 46(1), pp. 1–44.
43. Sanh V., Webson A., Raffel C., Bach S.H, Teehan R, Biderman S, Gao L, Bers T, Wolf T, & Rush A.M. (2021), ‘Multitask Prompted Training Enables Zero-Shot Task Generalization’ – ArXiv, abs/2110.08207.
44. Schulz S. (2013), ‘Simple and Efficient Clause Subsumption with Feature Vector Indexing’ – Lecture Notes in Computer Science(), vol 7788. Springer, Berlin, Heidelberg.
45. Segurado, P, Branco, P, & Ferreira, M. T. (2013), ‘Prioritizing restoration of structural connectivity in rivers: a graph based approach’ – Landscape Ecology, 28(7), pp. 1231–1238.
46. Shabbir J. & Anwer (2018), ‘Artificial Intelligence and its Role in Near Future’, ArXiv, abs/1804.01396.
47. Shawar B. A, & Atwell E. S. (2005), ‘Using corpora in machine-learning chatbot systems’ – International Journal of Corpus Linguistics, 10(4), pp. 489–516.
48. T. Lin and I. Joe (2023), ‘An Adaptive Masked Attention Mechanism to Act on the Local Text in a Global Context for Aspect-Based Sentiment Analysis’ – IEEE Access(2023), Vol. 11, pp. 43055-43066.
49. Touvron H., Martin L., Stone K. H., Albert P. J., Almahairi A., Babaei Y., Bashlykov N., Batra S., Bhargava P., Bhosale S., Bikel D., Blecher L., Ferrer C. C., Chen M., Cucurull G., Esiobu D., Fernandes J., Fu J., Fu W., Scialom T. (2023), ‘Llama 2: Open foundation and Fine-Tuned chat models’ – arXiv.org, Computer Science, Computation and Language.
50. Wei, J. (2022), ‘Emergent abilities of large language models’ – arXiv.org, Computer Science, Computation and Language.
51. W. Kuang (2023), ‘FederatedScope-LLM: A comprehensive package for fine-tuning large language models in federated learning’ – arXiv.org,

Computer Science, Artificial Intelligence.

52. Xing W, & Ghorbani A. (2004), ‘Weighted PageRank algorithm’ – Proceedings Second Annual Conference on Communication Networks and Services Research, 2004.
53. Yıldırım M., Okay F. Y., & Özdemir S. (2021), ‘Big data analytics for default prediction using graph theory’ – Expert Systems with Applications, 176, 114840.
54. Yoon S. (2023), ‘Short answer grading using one-shot prompting and text similarity scoring model’ – arXiv.org, Computer Science, Computation and Language.
55. Zeng A. (2023), ‘AgentTuning: Enabling Generalized agent abilities for LLMs’ – arXiv.org, Computer Science, Computation and Language.



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

SCHOOL OF COMMERCE & MANAGEMENT

NATIONAL CONFERENCE

"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"

**CERTIFICATE
OF PARTICIPATION**

This is to certify that

Ms. Aishwarya S

has Participated & Presented Paper Titled

Conversational Agent for Website Engagement using LLaMA 2

at Two Days National Conference on **"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"** organized by D. Y. Patil International University, Akurdi on 26th & 27th April, 2024.

Dr. Priyanka Dhoot
Conference Co-Convenor
& Assistant Professor,
SCM-DYPIU

Dr. Madhavi Deshpande
Conference Convenor
Director-SCM,
Dean Academics-DYPIU



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

SCHOOL OF COMMERCE & MANAGEMENT

NATIONAL CONFERENCE

"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"

CERTIFICATE OF PARTICIPATION

This is to certify that

Ms. Manogna Bollineni

has Participated & Presented Paper Titled

Conversational Agent for Website Engagement using LLaMA 2

at Two Days National Conference on **"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"** organized by D. Y. Patil International University, Akurdi on 26th & 27th April, 2024.

Dr. Priyanka Dhoot
Conference Co-Convenor
& Assistant Professor,
SCM-DYPIU

Dr. Madhavi Deshpande
Conference Convenor
Director-SCM,
Dean Academics-DYPIU



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

SCHOOL OF COMMERCE & MANAGEMENT

NATIONAL CONFERENCE

"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"

CERTIFICATE OF PARTICIPATION

This is to certify that

Dr. Leela Rani P

has Participated & Presented Paper Titled

Conversational Agent for Website Engagement using LLaMA 2

at Two Days National Conference on **"TRANSFORMING BUSINESS BY LEVERAGING TALENT, INNOVATION & DISRUPTIVE TECHNOLOGIES"** organized by D. Y. Patil International University, Akurdi on 26th & 27th April, 2024.

RP Dhoot

Dr. Priyanka Dhoot
Conference Co-Convenor
& Assistant Professor,
SCM-DYPIU

MDeshpande

Dr. Madhavi Deshpande
Conference Convenor
Director-SCM,
Dean Academics-DYPIU