

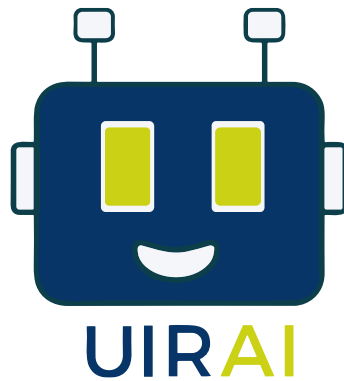
INTERNSHIP REPORT
FOR
FINAL-YEAR PROJECT (PFE)
5th Year Engineering Cycle

Higher School of Computer Science and Digital
Technologies

By

Aymane JOUHARI & Marouane BENBRAHIM

**Chatbots Utilizing LangChain, Pinecone, GPT-3.5, and Llama 2,
Based on UIR Program's Information**



Academic and Professional Tutor:

Hakim HAFIDI

2023 -2024

Chatbots Utilizing LangChain, Pinecone, GPT-3.5, and Llama 2, Based on UIR Program's Information

Abstract:

This project explores the development and comparative analysis of two educational chatbots using GPT-3.5 and Llama 2, supplemented by GPT Builder. It delves into the design, development, and integration of advanced technologies like Langchain, Pinecone, and Sentence-Transformers. The chatbots aim to enhance student interaction and information accessibility at Université Internationale de Rabat (UIR). The study evaluates the chatbots' performance in terms of response accuracy, speed, and user satisfaction, providing insights into the potential of Large Language Models (LLMs) in educational settings.

Key Words:

Chatbots, GPT-3.5, Llama 2, GPT Builder, Langchain, Pinecone, Sentence-Transformers, Large Language Models (LLMs), Together AI, AI in Education, User Interaction.

Acknowledgements

We would like to thank our supervisor, Mr. Hafidi Hakim, for his invaluable guidance and support throughout our final project. His expertise and insight have been pivotal in shaping the direction and success of our work.

We are immensely thankful to the International University of Rabat for providing an enriching and challenging academic environment over the past four years. The knowledge and experiences gained here have been foundational in our personal and professional growth.

Our heartfelt appreciation extends to all our professors who have imparted their wisdom and fostered a spirit of inquiry and learning. Their dedication to teaching and mentoring students has profoundly impacted our academic journey.

We would also like to acknowledge our colleagues and friends, who have been a source of inspiration, motivation, and support. The collaborative spirit and camaraderie we shared made our time at the university both enjoyable and rewarding.

Lastly, we extend our deepest gratitude to our families, whose unwavering support and encouragement have been our strength throughout these years. Their belief in us and constant support have been the cornerstone of our achievements.

Table of contents

I.	Introduction	6
a.	Context and justification of the project.....	6
b.	Objectives of the final year project	7
c.	Technologies Used	8
II.	Literature Review:.....	11
1.	Review of existing literature on chatbots and their application in education.....	11
a.	Introduction to Chatbots in Education:.....	11
b.	Historical Evolution of Chatbots:.....	14
c.	Types of Educational Chatbots:.....	17
d.	Benefits and Challenges of Chatbots in Education	19
2.	Overview of recent advancements in LLMs and their use in conversational chatbots:	22
a.	Introduction to Large Language Models (LLMs):.....	22
b.	Introduction to Natural Language Processing (NLP):	26
c.	Introduction to Transformers:.....	31
d.	Introduction to Lang chains:	38
e.	LangChain Components:	41
III.	Methodology	47
a)	Detailed explanation of the method used for collecting and preparing UIR's data.	47
b)	Description of the process for embedding storage using Pinecone.....	48
c)	Steps involved in integrating OpenAI GPT-3.5 Turbo and Llama 2 into the chatbot.	49
IV.	Chatbot Design and Development	51
a)	Design (Streamlit).....	51
1.	User Interface.....	51
2.	Streamlit Components and Their Usage:	52
3.	Streamlit's Design in the Code:	53
b)	Development (Langchain)	54
1)	Llms	54
2)	Chains.....	55
3)	Memory.....	55
4)	Prompt Template.....	56
5)	Query Refiner	57
6)	gTTS.....	58
V.	Evaluations, results, and Analysis	58

a) GPT Builder	58
b) Evaluation Procedure:	58
c) Presentation of achieved results.	59
Question 1 :	61
Question 2 :	63
Question 3 :	64
Question 4 :	66
Question 5 :	68
Question 6 :	70
Question 7 :	71
Question 8 :	73
Question 9:	75
Question 10 :	76
d) Final Score	78
e) Strengths and Limitations:.....	79
VI. Conclusion	81
References:	82

I. Introduction

a. Context and justification of the project

Picture yourself as a student eager to know more about the programs available at Université Internationale de Rabat (UIR). It is tough when you are not close by—asking UIR staff is not easy, and finding the exact information on their website can be a challenge. This is where our solution steps in! The chatbot comes as a handy helper. It acts like a smart buddy, swiftly providing answers about UIR programs. It is created to make life simpler for students, ensuring they get all the needed information without needing to visit the campus or spend hours searching the UIR website.



This chatbot is a response to the common issue of students not getting quick answers about UIR programs. It is like having a reliable friend on call, always ready to answer any questions students have about the university's programs and programs. Its main aim is to make students' lives easier by offering fast and accurate information about the education UIR provides.

The idea of this chatbot aligns perfectly with UIR's vision. It is all about making things easier for students—bridging the gap between what they need to know and the information available. By leveraging this AI-driven chatbot, UIR aims to create a smoother and friendlier experience for students seeking details about the university's educational offerings.

This chatbot is a one-stop-shop for students. It is like having a reliable and super-quick source to get all the necessary information about UIR's wide range of academic programs and programs.

b. Objectives of the final year project

- **User-Friendly Interface:** Develop a chatbot interface that is incredibly easy for students to use. It should swiftly provide information about UIR's programs, regardless of their location or technical expertise.
- **Swift and Accurate Responses:** Ensure the chatbot swiftly delivers accurate responses to students' queries regarding UIR's programs, minimizing waiting times and enhancing user satisfaction.
- **Detailed Information Access:** Provide comprehensive details about various programs, application procedures, faculty profiles, and other essential aspects of studying at UIR, making this information easily accessible through the chatbot.
- **Language Adaptability:** Implement language adaptability within the chatbot so that it responds in the language the user prefers. If a student asks a question in French, the bot responds in French, ensuring a seamless conversation.
- **Smart Technology Integration:** Utilize smart technologies like Langchain, All MiniLM v6, and OpenAI GPT-3.5 Turbo to fetch and present information effectively, making the chatbot intuitive and user-friendly.
- **Efficient Information Retrieval:** Develop a system that efficiently retrieves the necessary information, ensuring the chatbot always provides accurate and updated answers.
- **Alignment with UIR's Goals:** Ensure that the chatbot's purpose aligns with UIR's vision of leveraging technology to simplify student engagement and enhance access to information effortlessly.



- ✓ These **objectives** aim to ensure the chatbot offers a seamless and helpful experience for students seeking information about Université Internationale de Rabat's programs while focusing on ease of use, adaptability, and accurate information delivery without specifically mentioning any database platform.

c. Technologies Used

- **Langchain:**

Langchain was launched in October 2022 is an open-source framework allowing software developers engaged in artificial intelligence (AI) and its machine learning sector to merge substantial language models with additional external components for creating applications powered by LLM (Longformer Language Model). The primary aim of LangChain is to connect robust LLMs like OpenAI's GPT-3.5 and GPT-4 with diverse external data sources, enabling the creation and utilization of natural language processing (NLP) applications to gain their advantages.



- **Pinecone (Vector Database for Embeddings):**

Pinecone serves as a vector database where embeddings, transforming language elements into numerical vectors, are stored. It helps in finding similarity between user queries and stored embeddings, enabling efficient and accurate responses based on similarity metrics.



- **Sentence-Transformers:**

Sentence-Transformers is a library offering simple methods for generating embeddings (dense vector representations) for sentences, paragraphs, and images. This process embeds texts into a vector space, ensuring that similar text becomes proximate. This capability allows for applications like semantic search, clustering, and retrieval. The library includes over 500 Sentence-Transformer models, sourced from Hugging Face.

- **All MiniLM v6 (Embedding Model):**

This Sentence-Transformers model is designed to serve as a sentence and short paragraph encoder. Upon receiving input text, it generates a vector that encapsulates the semantic information. The resulting sentence vector can be used for various purposes like information retrieval, clustering, or sentence similarity tasks. It maps sentences and paragraphs to a 384-dimensional dense vector space, facilitating tasks like clustering and semantic search.



- **OpenAI GPT-3.5 Turbo (Large Language Model):**

OpenAI GPT-3.5 Turbo, being a large language model, significantly contributes to the project by empowering the chatbot with an extensive understanding of language nuances. It enables the generation of coherent and contextually relevant responses to user queries.



- **Llama 2 (Large Language Model):**

Llama 2, Meta's large language model (LLM), stands as the Facebook parent company's answer to OpenAI's GPT models and Google's AI models. However, its standout feature lies in being openly accessible for research and commercial applications, setting it apart by offering free usage to anyone.



- **Streamlit (User Interface for the Chatbot):**

Streamlit serves as the user interface (UI) for the chatbot. It provides an interactive platform where users can seamlessly interact with the chatbot, making the experience user-friendly and intuitive.



- **gTTS (Google Text-to-Speech):**

gTTS is utilized to transform text responses generated by the chatbot into speech. It enhances the chatbot's functionality by providing users with an audio output of the text responses, ensuring accessibility and convenience.



- ✓ These **technologies** collectively form the backbone of the project, contributing to various aspects such as language understanding, information retrieval, user interaction, and response generation within the chatbot.

II. Literature Review:

1. Review of existing literature on chatbots and their application in education.

a. Introduction to Chatbots in Education:

▪ What are chatbots:

Chatbots are interactive or conversational agents designed to provide immediate responses to users. Operating as software applications, these artificial intelligence entities engage users in text or voice-based conversations, offering information, assistance, or guidance in a manner akin to human interaction. Deployed across various platforms, chatbots contribute to a wide array of applications, particularly in education, where their role in enhancing student engagement and support is increasingly recognized.



In education, chatbots are these talkative assistants that quickly reply when you ask them something. Especially now, when many students in higher education use smartphones and spend a lot of time online, chatbots are like helpful apps that can give you instant info. They are often set up as mobile web apps, making it easy for students to get standardized details quickly.

▪ Chatbot Functionality

The functionality of chatbots relies on their ability to process and comprehend user input, generate appropriate responses, and execute predefined actions. Key components include:

- ✓ **Natural Language Processing (NLP):** Chatbots employ NLP algorithms to understand and interpret human language, allowing them to grasp user intent and context.
- ✓ **Pattern Recognition:** Machine learning algorithms enable chatbots to recognize patterns in user input, improving their ability to generate relevant and contextually appropriate responses over time.
- ✓ **Predefined Responses and Actions:** Chatbots are programmed with predefined responses and actions based on anticipated user queries. This enables them to provide information, answer questions, or execute specific tasks within their programmed capabilities.
- ✓ **Learning and Adaptation:** Advanced chatbots can learn from user interactions and adapt their responses over time. This adaptability enhances their effectiveness in addressing a diverse range of user queries.



- **Chatbot potential applications in the education sector:**

The use of Artificial Intelligence (AI), especially through Chatbots, has brought a tremendous change to education. This technology is growing fast and has a lot of potential to change the way we teach and learn. Chatbots in education have different uses:

- ✓ **Personalized Learning Experiences:**

Chatbots can make learning personal by giving each student customized content, practice questions, and assessments based on what they need. This makes learning more effective and tailored to each student.

- ✓ **Enhanced Student Interaction:**

In today's online world, Chatbots give students a way to interact better. Available on mobile apps, these talking bots help students by giving instant answers, guiding them, and providing information about their studies.

- ✓ **Administrative Support for Educators:**

Chatbots make life easier for teachers by handling routine tasks. They can share details about courses, assignment deadlines, directions on campus, and more. This support lets educators focus on important things like creating the curriculum and doing research.

- ✓ **24/7 Accessibility to Learning Resources:**

Chatbots are available all the time, making it easy for students to get help and access study materials whenever they need them. This flexibility is great for students with different schedules and learning preferences.

- ✓ **Subject-Specific Learning and Support:**

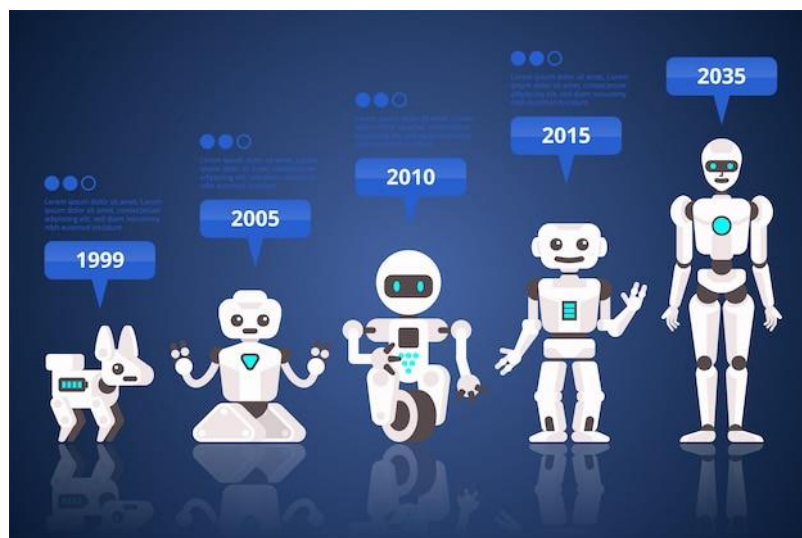
Chatbots have been utilized in various subject domains, including answering specific questions, aiding in understanding complex concepts like Computer Programming, and delivering assessments to gauge students' performance abilities.

✓ Individualized Learning Experiences:

Unlike traditional ways of communication like email or talking to the teacher, Chatbots make learning personal. They give tailored responses to each student, creating a more comfortable and interesting learning space.



b. Historical Evolution of Chatbots:



The captivating journey of chatbots can be traced back to 1950 when Alan Turing, a pioneer in computer science, contemplated the intriguing idea of whether a computer program could engage in conversations with a group of people without them realizing it is not a real person. This question, known as the Turing test, started the whole idea of chatbots. Let us see how chatbots evolved through the years:

✓ ELIZA (1966):

The first major development happened in 1966 when ELIZA, the initial chatbot created by Joseph Weizenbaum, came into existence. ELIZA played the part of a psychotherapist, answering user sentences with questions. Although ELIZA had limited communication abilities, it served as a source of inspiration for future improvements in chatbot technology.

✓ PARRY (1972):

In 1972, another step in developing chatbots took place with PARRY. PARRY was a chatbot that pretended to be a person with schizophrenia. It was seen as more advanced than ELIZA because it had a kind of "personality" and formulated its responses using a set of assumptions and emotional cues.

✓ Jabberwacky (1988):

In 1988, a big leap in chatbot history happened with the arrival of Jabberwacky. It was created using CleverScript, a language built on spreadsheets. Jabberwacky used contextual pattern matching to respond. Even though it was a noteworthy progress, it had some difficulties with speed and capacity.

✓ Chatterbot Term Introduction (1991):

The term "Chatterbot" was first used in 1991 to talk about a TINYMUD artificial player made for chatting (Mauldin, 1994). In 1992, Dr. Sbaitso was created as a basic chatbot to highlight what sound cards could do.

✓ ALICE (1995):

In 1995, a noteworthy event occurred with the introduction of ALICE (Artificial Linguistic Internet Computer Entity). Drawing inspiration from ELIZA, ALICE used patterns to engage in conversations on the web, standing out for its enhanced ability to discuss diverse topics (Wallace, 2009). During this time, a new language called Artificial Intelligence Markup Language (AIML) was introduced, setting ALICE apart from earlier chatbots.

✓ SmarterChild (2001):

Around the year 2001, a chatbot named SmarterChild emerged on Messengers such as AOL and MSN. It was a pioneering chatbot, offering helpful daily assistance by providing information on movie times, sports scores, stock prices, news, and weather. This marked a meaningful change in what chatbots could do.

✓ Smart Personal Voice Assistants (2010 onwards):

In the next ten years, we saw the popularity of smart personal voice assistants grow, and Siri took the lead in 2010. Siri, created by Apple, was a pioneer in letting people talk to their devices. It answered questions and changed how it talked based on how people spoke and what they liked (Siri, 2020). Other important voice assistants, like Google Now (2012), Google Assistant (2016), Microsoft Cortana (2014), and Amazon Alexa (2014), also made voice-activated technology more widespread.

✓ IBM Watson (2011):

In 2011, IBM Watson was born, proving the power of Artificial Intelligence (AI) by winning a quiz competition. Over time, it transformed into a useful tool for businesses and healthcare, showing how chatbots can be versatile in different areas.

✓ Social Media Chatbots (2016):

In 2016, something important happened. Social media platforms started using chatbots, and this allowed developers to make chatbots for specific brands. These chatbots could then talk to customers in messaging apps. This change led to chatbots becoming more common in areas like Marketing, Support, Healthcare, Entertainment, Education, and Cultural Heritage.

✓ Microsoft Xiaolce (2016):

In 2016, Microsoft introduced Xiaolce, a chatbot that really made a mark. What set Xiaolce apart was its emotional intelligence (IQ–EQ) and friendly personality. It formed lasting emotional bonds with users, considering cultural differences and ethical considerations.

✓ Modern Conversational Abilities:

Modern chatbots converse in a way that is quite different from older versions. They can talk about personal matters, handle family-related topics, be both clear and puzzling, and even act in a way that resembles human conversation.

The use of chatbots has been growing significantly, especially after 2016, as shown in **Figure 1**. When we look at **Figure 2**, the United States has the highest interest in researching chatbots, with the United Kingdom and Japan following, but contributing less than one-third of the research compared to the USA.

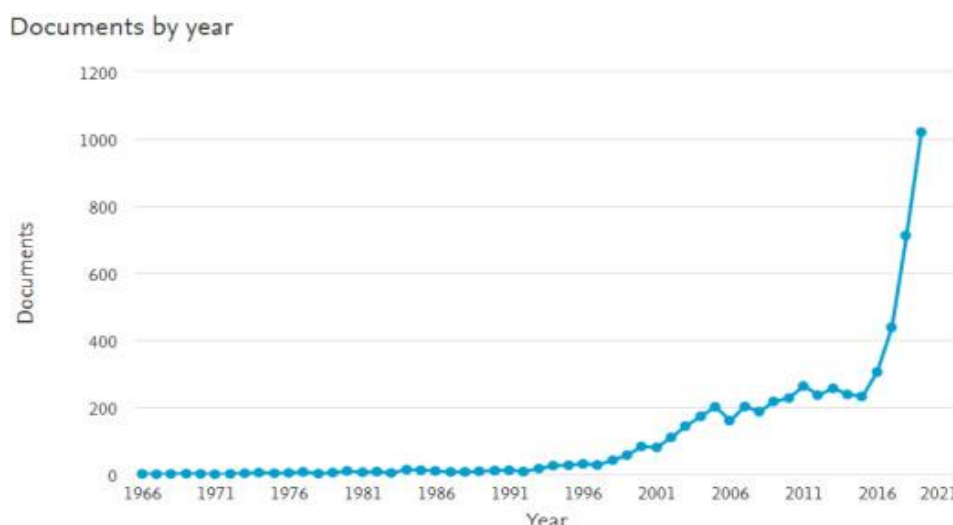


Fig. 1: Search Results in Scopus ([Scopus preview—Scopus—Welcome to Scopus, 2020](#)), from 1966 to 2019 for the keywords “chatbot” or “conversation agent” or “conversational interface”.

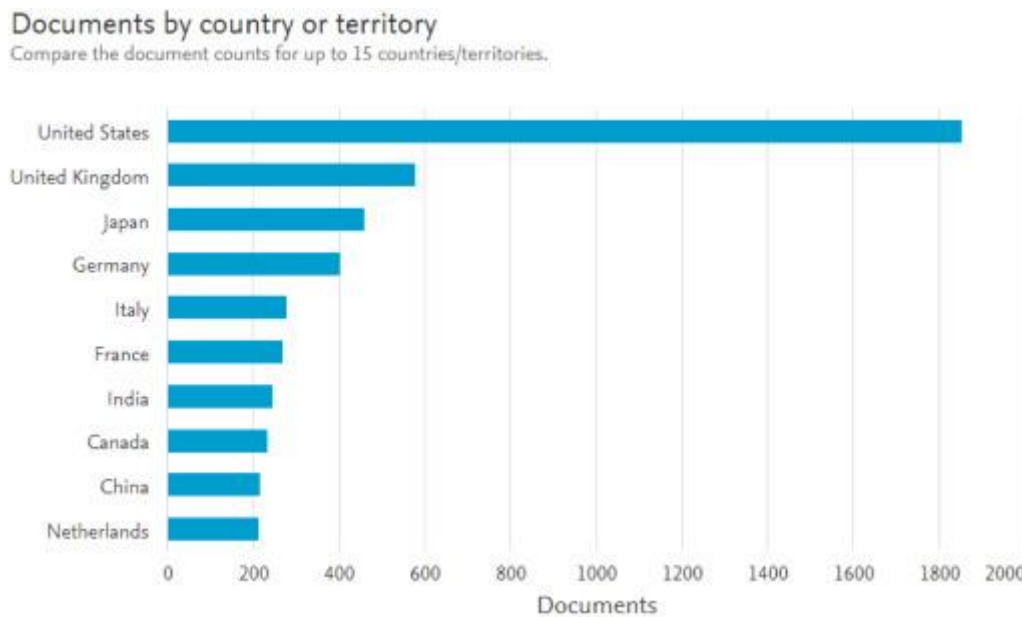


Fig. 2: Search results in Scopus ([Scopus preview—Scopus—Welcome to Scopus, 2020](#)) from 1966 to 2019 for the keywords “chatbot” or “conversation agent” or “conversational interface” by country or territory.

c. Types of Educational Chatbots:

Let us dive into the world of educational chatbots and discover the several types that are making a difference in learning. These chatbots go beyond just answering questions; they can be your tutors, information friends, and even help you learn a new language. Let us explore these diverse types and see how they are shaping the future of education.

✓ Tutoring Bots:

Imagine having a personal tutor available anytime you need help with your studies. Tutoring bots do just that. They assist you in understanding complex concepts, provide extra practice problems, and offer guidance on various subjects. These bots adapt to your pace, making learning a personalized experience.

✓ Information Providers:

Need quick answers or information on a specific topic? Information provider bots have gotten you covered. These bots act as virtual encyclopedias, offering details on a wide range of subjects, from historical events to scientific facts.

✓ Language Learning Bots:

Learning a new language becomes more engaging with language learning bots. These bots help you practice vocabulary, grammar, and conversational skills, making language learning interactive and fun.

▪ Examples of Successful Implementations:

✓ Duolingo:

Duolingo, a well-known language learning platform, uses a chatbot interface to make language learning engaging and effective. It provides short lessons, interactive conversations, and fun challenges.



✓ Squirrel AI:

Squirrel AI is an AI-powered tutoring system that tailors learning paths for students. It assesses strengths and weaknesses, adjusting content delivery to enhance learning outcomes.



✓ IBM Watson Education:

IBM Watson Education uses AI to create personalized learning experiences. Its chatbots help students grasp complex topics, offer extra resources, and provide guidance, creating a more customized educational experience.



d. Benefits and Challenges of Chatbots in Education

Chatbots act as digital helpers, bringing both positive aspects and a few challenges. In this section, we will delve into the benefits and challenges associated with chatbots.

▪ Chatbot Benefits:



Some cool things about chatbots are quite clear. They are always ready to help, making our lives easier by doing tasks while we talk. Let us see some benefits:

✓ Entertainment:

Chatbots can do many tasks, like being a personal fashion advisor, trip advisor, or even an interactive music player. They make things more personal, and you do not always need a person to chat with; chatbots can handle it. Examples of this include Google Assistant, Siri, or Cortana, which are easy to use and can answer almost any request you make.

✓ Industry:

Chatbots are like new smart interfaces for modern devices. They make it easier for people to connect with businesses, especially in customer service, content distribution, or affiliate marketing. They help save money in call centers and let humans focus on more important things.

✓ Education:

One of the most important things chatbots do is help with education. They are useful if they act in a friendly way to assist users in reaching their goals and solving problems. As people need more skills for work, chatbots bring new ways of learning alongside traditional methods and teachers. They play a significant role in teaching all kinds of subjects. For instance, Alexa instructs kids about recycling and gives adults advice on health and wellness. With the right knowledge and technology boost, chatbots can help prevent many problems.

▪ Chatbot Challenges:

Chatbots act as friendly guides, adding an interesting touch to learning. While they bring some positive aspects to the table, there are also a few challenges we should discuss.



✓ Implementation Concerns:

Setting up chatbots in schools can be a bit like figuring out a puzzle. Making sure they work smoothly and help students learn requires some careful planning.

✓ Privacy and Security:

Some people might worry about their confidential information when they talk to chatbots. Making sure that information stays safe and does not get into the wrong hands is something to think about.

✓ Technical Issues:

Just like a computer might have a small glitch now and then, chatbots can sometimes have tiny problems too. They do not understand everything or need a little fixing up.

✓ People Acceptance:

Not everyone may like the idea of using chatbots for learning or obtaining information. Some may find it unusual and prefer traditional ways of learning or gathering information.

2. Overview of recent advancements in LLMs and their use in conversational chatbots:

a. Introduction to Large Language Models (LLMs):

▪ Large Language Models (LLMs) Definition:

A large language model, often called LLM, is like a smart computer program that is good with words. It uses a special kind of learning called deep learning, which helps it do many language-related tasks effortlessly. Picture it as a language whiz that can recognize, translate, predict, or even create text and other content.

These language models, also known as neural networks, are like computer brains inspired by how our human brains work. They have layers of interconnected nodes, like how our brain neurons connect.

Apart from teaching computers human languages, LLMs can be trained to do different jobs like understanding the shapes of proteins or writing computer code. Just like how we learn, these models go through two steps: first, they get a general understanding, and then they fine-tune themselves to be efficient at specific tasks like answering questions, summarizing documents, or creating new text. It is like training a superhero to solve various kinds of language problems.

These language models are not simply good for talking. They can also be super helpful in areas like healthcare, finance, and entertainment. In these fields, they do all sorts of language-related jobs, from translating languages to chatting with you like a friend. You might have heard of chatbots or AI assistants – these are all part of what LLMs can do.

Large language models also have large numbers of parameters, which are akin to memories the model collects as it learns from training. Think of these parameters as the model's knowledge bank.



- **How LLMs Operate:**

Large language models (LLMs) function as digital linguists, and understanding their operation involves delving into the intricacies of artificial neural networks, specifically transformers.

- ✓ **Learning Through Practice:**

Think of LLMs as diligent students immersed in a vast library of language examples. They employ artificial neural networks, particularly transformers, undergoing training via self-supervised learning and semi-supervised learning. During this phase, they become adept at predicting the next word or token in a sentence.

- ✓ **The Transformer Model:**

An LLM, rooted in the transformer model, operates by receiving an input, encoding it, and decoding it to generate an output prediction. However, before it can process text input and deliver accurate predictions, it undergoes a dual-phase preparation: training and fine-tuning.

- ✓ **Training for General Functions:**

For general language prowess, LLMs undergo pre-training using extensive textual datasets sourced from platforms like Wikipedia or GitHub. These datasets, comprising trillions of words, play a pivotal role in shaping the model's language understanding. Unsupervised learning characterizes this phase, allowing the LLM to grasp word meanings, understand relationships between words, and discern contextual nuances.

- ✓ **Fine-Tuning for Specific Tasks:**

To excel in tasks like translation, LLMs embark on the fine-tuning process. This phase optimizes the model's performance, aligning it with the intricacies of targeted activities.

- ✓ **Prompt-Tuning: Tailoring to Tasks:**

Prompt-tuning, akin to fine-tuning, hones the model's skills for specific tasks. It achieves this through either a few-shot prompting or zero-shot prompting. Few-shot prompting involves training the model with examples to predict outputs based on given instructions. For instance, in sentiment analysis, providing positive and negative examples helps the model understand sentiments.

✓ **Zero-Shot Prompting: Task without Examples:**

On the other hand, zero-shot prompting challenges the LLM to respond to inputs without explicit examples. It formulates tasks through direct instructions without guiding problem-solving instances. For example, when asked, "The sentiment in 'This plant is so hideous' is...", the model infers the task without specific examples.

▪ **Importance of LLMs in Natural Language Processing:**

The importance of LLMs in natural language processing is groundbreaking. They function as a connection between human language and computers, enabling machines to understand the intricacies of syntax, semantics, and even the underlying "ontology" found in human language data. LLMs, including ones like GPT-3, not only comprehend language but can also generate relevant text in various contexts, making them incredibly useful in applications like chatbots and text creation tasks. However, it is crucial to recognize that while LLMs are language experts, they may also inherit inaccuracies and biases from the data they learn, emphasizing the need for careful consideration in their use.

▪ **Historical evolution of Large Language Models:**

✓ **Early Days (1950s-2000s):**

Large language models (LLMs) have been around since the mid-20th century. People first tried to make computers talk like humans in the 1950s. But back then, computers had some trouble keeping up because they were not as powerful and making them understand human language was a bit tricky.

✓ **Neural Networks Rise (2010s):**

Things got exciting in the 2010s when smart folks started using deep learning and neural networks. They started creating fancy models like transformers, making language models smarter. During this time, models like Word2Vec and GloVe became the cool kids on the block.

✓ **GPT and BERT Time (2018-2019):**

In 2018, OpenAI made a big deal with their Generative Pre-trained Transformer (GPT). It showed off by learning a lot from huge sets of data. Around the same time, Google introduced BERT, a model that paid attention to words from both sides, making it super good at understanding language.

✓ GPT-3 Takes Over (2020s):

GPT-3 showed up in 2020, and it was like the superhero of LLMs. With a whopping 175 billion tricks up its sleeve (parameters), it could do amazing things like chat with people and create content. It became a big deal in many industries.

✓ Keeping It Going (2020s-2021s):

After GPT-3, the race was on to make even bigger and smarter models. OpenAI said, "Hey, here's GPT-4!" Other companies and brainy folks joined in, all trying to make LLMs that really get language, making them better at understanding and saying things.

✓ Diverse Applications (2022 Onward):

In 2022, big language models became the heroes of many jobs. They help virtual assistants do their thing and make content, among other cool stuff. They are like superheroes of language understanding. People are still figuring out more ways these models can be even cooler in the future.



b. Introduction to Natural Language Processing (NLP):

▪ **Natural Language Processing Definition:**

Natural Language Processing (NLP) stands at the intersection of artificial intelligence (AI), linguistics, and computer science. It is a smart way for computers and people to chat and understand each other using human language. NLP techniques make it possible for computers to dive into our spoken words and written text, comprehend them, and craft meaningful responses. It is like teaching computers to talk with us in our own language.

NLP is a specific field within computational linguistics, where scientists blend computer science, linguistics, and AI to unravel the computational aspects of human language. It is a bit like decoding the secrets of how we naturally communicate.



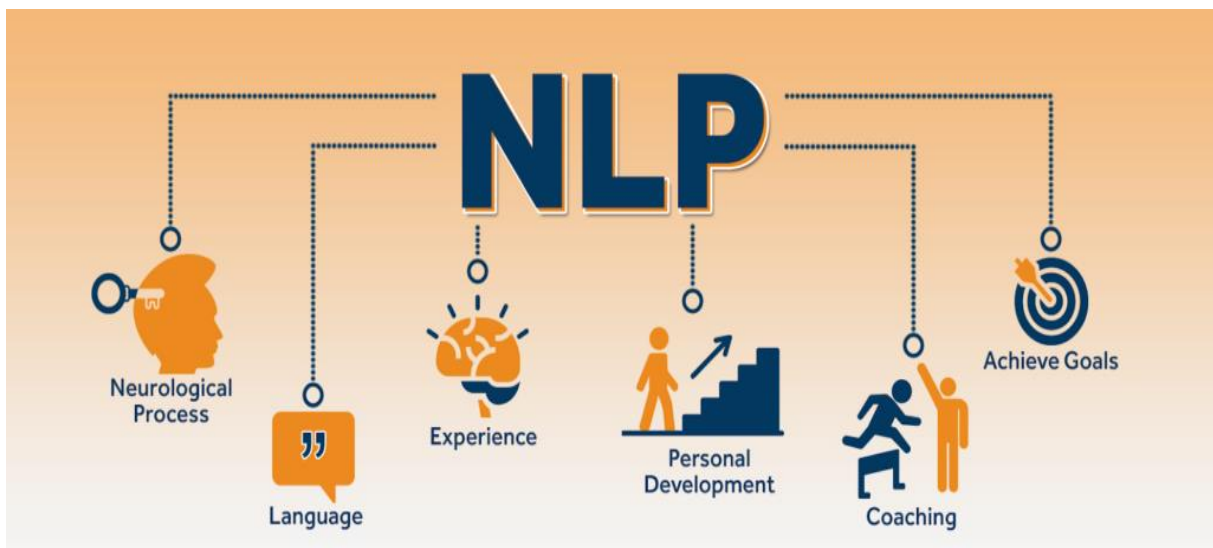
- **History of Natural Language Processing (NLP):**

The journey of Natural Language Processing began back in the 1950s when clever minds in computer science set out to teach machines the art of understanding and generating human language. Think of it as the pioneers trying to bridge the gap between human speech and machine-generated speech. In 1950, Alan Turing, a mathematician, proposed the Turing Test, a clever challenge to see if machines could sound as lifelike as humans.

The initial phase of NLP research, during the 1960s, leaned on rule-based methods. Scientists got creative, using semantic analysis, parts-of-speech tagging, and parsing to decode human language. They even created the first corpora, large documents filled with linguistic information, to train NLP algorithms. It was like giving machines a linguistic crash course.

By the 1970s, statistical NLP stepped in as an alternative to rule-based approaches. This method uses statistical models to understand and generate natural language text, giving machines a more flexible way to grasp our words.

As the 1980s rolled in, NLP researchers got serious about efficient algorithms. They wanted models that could learn better and be more accurate. This sparked the rise of machine learning in NLP, where machines dig into large piles of data to spot patterns and make predictions.



▪ **Understanding Natural Language Processing:**

Natural Language Processing (NLP) is a fascinating field that allows computers to comprehend and generate human language. Let us explore how it works using different approaches.

✓ **AI-Based Approach (Chatbots):**

The most common method involves Artificial Intelligence (AI) and machine learning algorithms. These algorithms learn from vast amounts of data to process, understand, and create human-like language. Think of it as the computer learning to talk by studying how we communicate.

Step 1: Data Prep - Making Text Understandable:

Before the computer can understand language, we need to prepare the text. It is like tidying up a messy room before using it. We clean the text using methods like breaking it into smaller parts (tokenization), removing familiar words that do not help much (stop word removal), and simplifying words to their basic forms (stemming and lemmatization). We also identify the roles of words in a sentence (part-of-speech tagging) and understand how words connect (parsing).

Step 2: Algorithms in Action - Extracting Meaning:

Now that our text is neat and tidy, we apply NLP algorithms to extract useful information. Some common tasks include:

- Sentiment Analysis: Figuring out if the text is positive, negative, or neutral.
- Named Entity Recognition: Identifying specific things like names, locations, dates, and organizations.
- Topic Modeling: Grouping similar words to find the main themes in a bunch of text.
- Machine Translation: Teaching computers to automatically translate text between languages.
- Language Modeling: Predicting what words might come next in each context, like predicting the next word when you are typing.

✓ Hybrid NLP - The Best of All Worlds:

Sometimes, we use a mix of these approaches to get the best results. It is like having a recipe that combines unusual flavors for a delicious dish.

✓ Branches of NLP - Understanding and Generating:

In the NLP world, there are two important branches. Natural Language Understanding (NLU) focuses on teaching computers to understand language like humans do, considering context, emotions, and intent. Natural Language Generation (NLG) works on making computers generate language that humans can easily understand.

▪ **Benefits of Natural Language Processing**

✓ Elevated Communication:

Natural Language Processing (NLP) enhances communication by allowing more natural interactions with search applications. It adapts to assorted styles and sentiments, resulting in more user-friendly experiences.

✓ Efficiency:

NLP automates numerous tasks traditionally performed by people. Tasks like text summarization, monitoring social media and emails, spam detection, and language translation become more efficient with NLP.

✓ Content Curation:

NLP identifies the most relevant information for users based on their preferences. Understanding context and keywords enhances customer satisfaction, and making data more searchable improves the effectiveness of search tools.

▪ **Challenges of Natural Language Processing**

✓ Navigating Human Speech:

NLP faces challenges in dealing with irregular and ambiguous human speech, with meanings often dependent on context. Programmers must teach applications to these intricacies from the beginning.

✓ Homonyms and Syntax Confusion:

Homonyms and syntax can confuse datasets, and even the best sentiment analysis struggles with sarcasm and irony. Learning these nuances takes humans years, and detecting tone in text messages or emails remains challenging.

✓ Multilingual Complexity:

Text is published in various languages, but NLP models are trained in specific languages. Language identification is necessary before feeding data into NLP to sort information by language.

✓ Data Specificity:

Unspecific and overly general data limits NLP's ability to accurately understand and convey the meaning of text. Industries requiring highly specific and up-to-date information may face challenges, and ongoing research like the ELSER model aims to address this limitation.

✓ Privacy Concerns:

Processing personal data raises privacy concerns, particularly in sensitive fields like healthcare. Extracting information from patient files for form filling and health issue identification raises ethical and security challenges, making NLP implementation difficult in certain domains.



c. Introduction to Transformers:

▪ **Definition and Evolution of Transformers in Deep Learning:**

Transformers, a revolutionary deep learning model, emerged in 2017, transforming the landscape of natural language processing (NLP) and gaining prominence across various machine learning and artificial intelligence tasks.

✓ **Origins and Key Contributors:**

In the seminal paper "Attention is All You Need" (2017), Ashish Vaswani, the Google Brain team, and the University of Toronto introduced the transformer model. This breakthrough publication marked a pivotal moment in the field, leading to the widespread adoption of transformers in applications like training Large Language Models (LLMs).

✓ **Versatility in Applications:**

Transformers highlight their power in real-time language translation, enabling tourists to communicate seamlessly and aiding researchers in DNA understanding and drug design acceleration. In finance and security, these models detect anomalies and prevent fraud, while vision transformers contribute to computer vision tasks.

✓ **Role in Text Generation:**

Notably, OpenAI's ChatGPT, a widely used text generation tool, harnesses transformer architectures, specifically the "generative pre-trained transformer" (GPT) framework. Versions like GPT-2 and GPT-3 empower ChatGPT to excel in prediction, summarization, and question answering, as transformers facilitate focusing on the most relevant segments of input text.

✓ **BERT and Its Impact:**

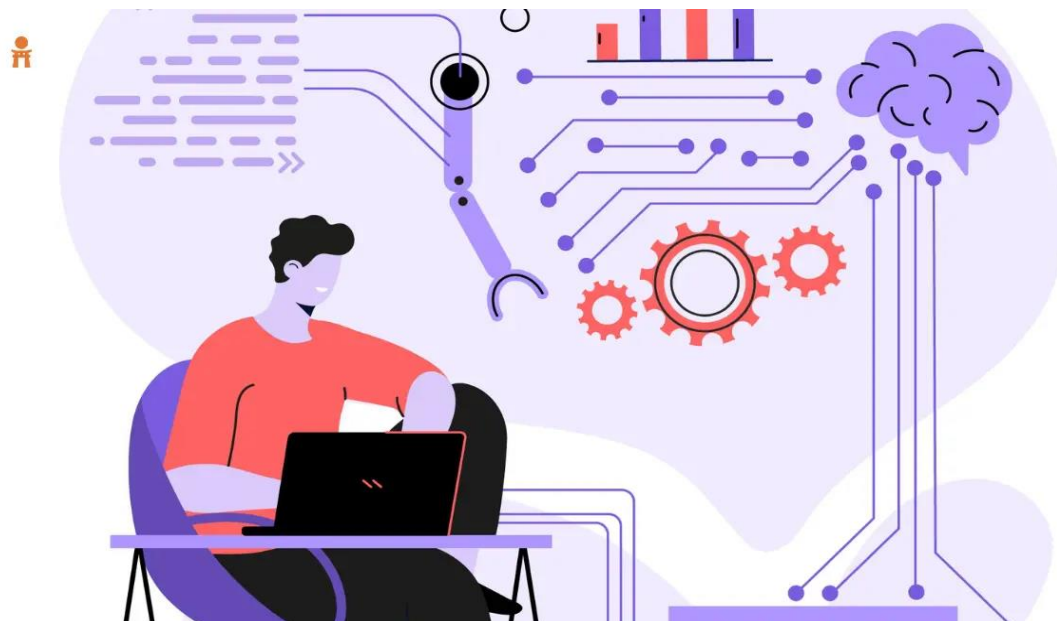
The Bidirectional Encoder Representations from Transformers (BERT) model introduced based on transformer architecture, revolutionized Google search results in multiple languages by 2019. BERT's influence extended to over 70 languages, highlighting the global impact of transformer-based architectures.

✓ Deep Learning Innovation:

The transformer architecture, driven by the multi-head attention mechanism, sets itself apart by eliminating recurrent units, significantly reducing training time compared to previous neural architectures like LSTM. This innovation, described in the 2017 paper, has found extensive applications beyond NLP, encompassing computer vision, audio processing, and multi-modal tasks.

✓ Historical Context:

Although the transformer paper was published in 2017, the roots of the softmax-based attention mechanism can be traced back to 2014, emphasizing its gradual evolution. The transformer's far-reaching influence extends to pre-trained systems such as Generative Pre-trained Transformers (GPTs) and BERT, solidifying its position as a cornerstone in modern deep learning.



- **Transformer models architecture and functionality:**

Neural networks, the powerhouse behind AI tasks like image recognition and natural language processing (NLP), have evolved over time. The traditional ones use an encoder/decoder pattern to handle sequences of data, like sentences.

- ✓ Traditional way: Sequential Processing:

In the past, these networks processed data step by step, analyzing one part at a time. It is like reading a book word by word, losing some details along the way, especially over long distances.

- ✓ Self-attention Mechanism:

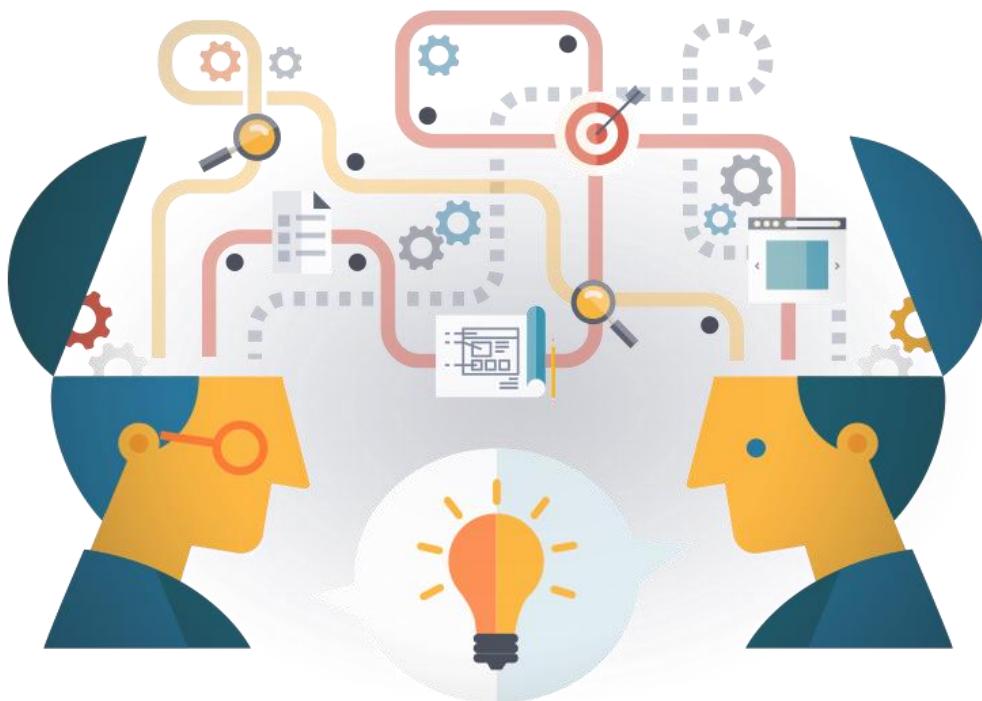
Transformers shook things up by introducing a self-attention mechanism. Picture yourself in a bustling room, focusing on one conversation amidst the noise—similarly, self-attention helps the model concentrate on crucial information all at once. This makes transformers faster, more efficient, and better at grasping context, crucial for understanding lengthy texts.

- ✓ How transformers operate (key steps):

Imagine you want to translate an English sentence into French using a transformer model. Here is how it works:

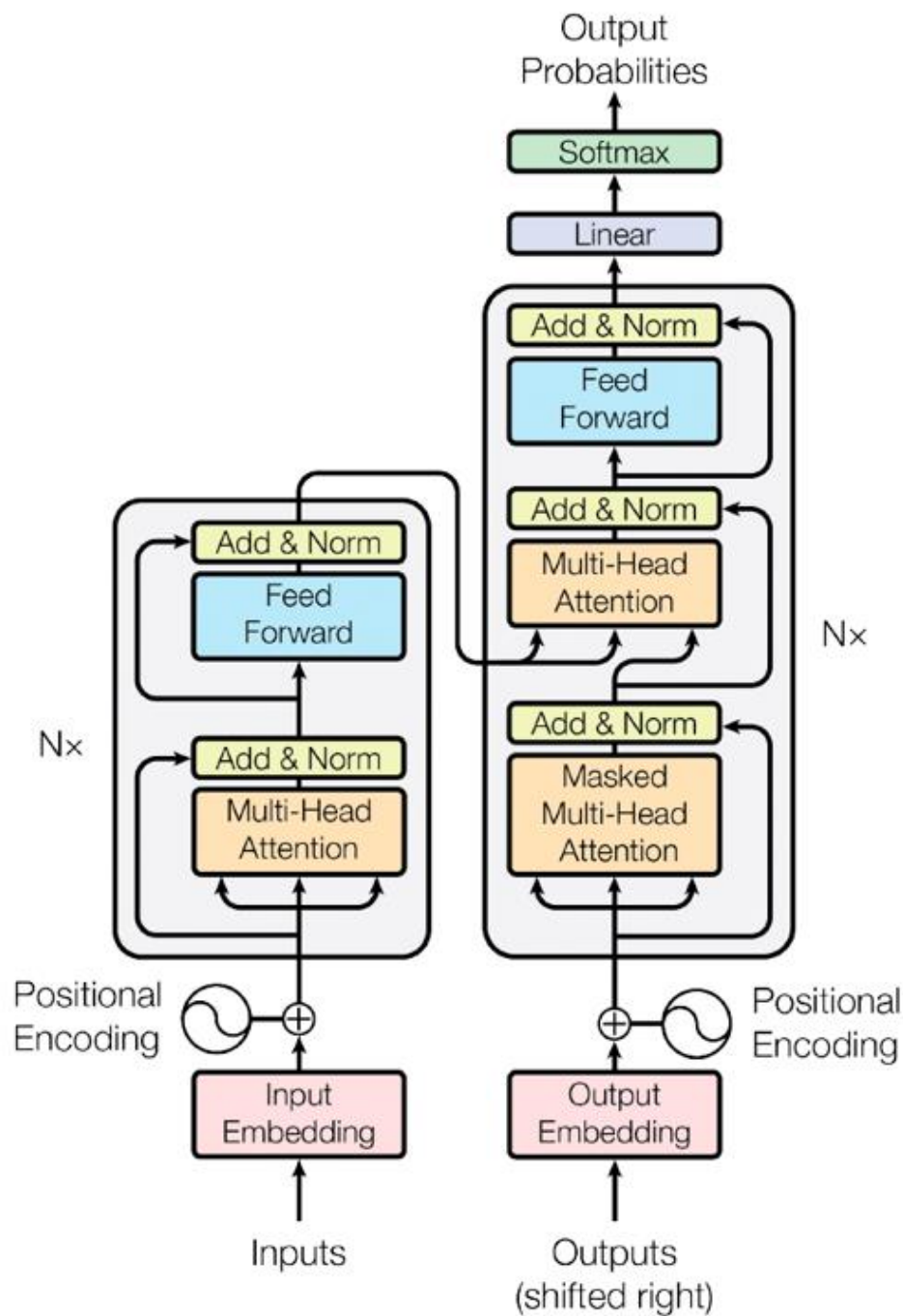
1. **Input Embeddings:** Transform the sentence into numerical representations called embeddings, capturing the meaning of each word.
2. **Positional Encoding:** Add positional information patterns to embeddings, ensuring the model understands word order.
3. **Multi-Head Attention:** Utilize multiple attention heads to consider various relationships between words using SoftMax functions.
4. **Normalization and Residual Connections:** Stabilize and speed up training using layer normalization and residual connections.

5. Feedforward Neural Networks: Apply non-linear transformations to token representations, enabling the model to grasp complex patterns.
6. Stacked Layers: Use multiple layers to process information hierarchically, capturing abstract features.
7. Output Layer: For tasks like translation, add a decoder to generate the output sequence.
8. Training: Train the model using supervised learning, minimizing the difference between predictions and ground truth.
9. Inference: After training, use the model on new data, passing the sequence through the pre-trained model to generate predictions.



▪ **Transformer architecture components:**

The transformer neural network, a powerful tool for language processing, comprises several key parts that collaborate to generate meaningful results. The following image shows the components of transformation architecture, as explained in the rest of this section.



✓ **Input Embeddings:**

Input embeddings transform human language into a computer-friendly format. Breaking down sentences into tokens (like words), embeddings convert them into mathematical vectors. These vectors, representing semantics and syntax, are learned during training. Imagine these vectors as coordinates, simplifying complex attributes. This numeric representation helps the model understand and process words efficiently, forming a crucial bridge between human language and computational understanding.

✓ **Positional Encoding:**

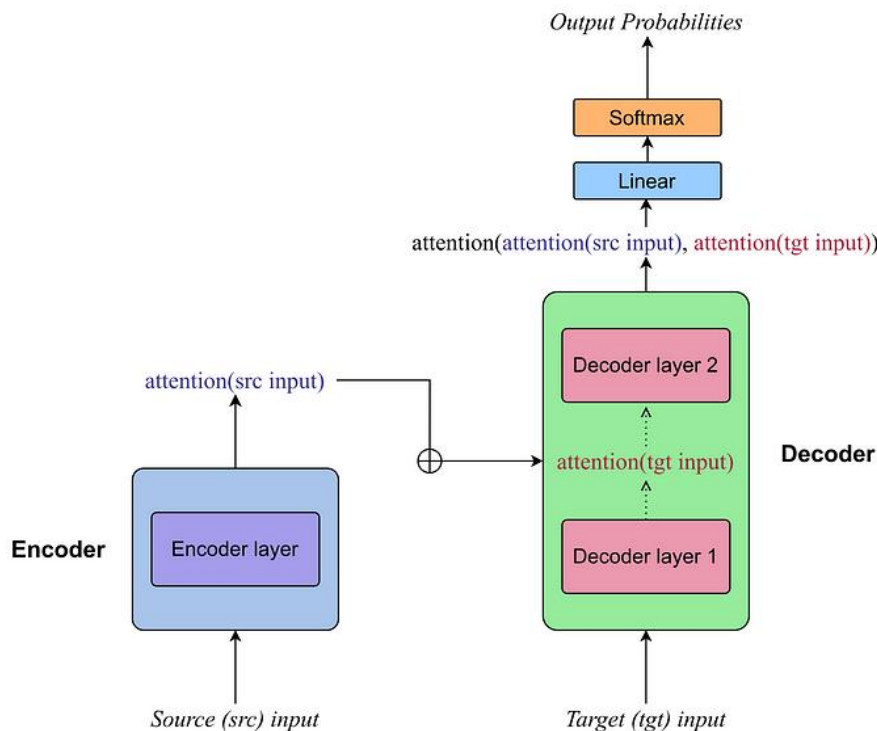
In the transformer architecture, handling sequential data order is vital, as the model does not inherently grasp it. To address this, we employ positional encoding, a key element. This technique enhances each token's embedding by incorporating information about its position in the sequence. Through unique positional signals added to each token's embedding, the model can now retain the order of tokens, comprehending the contextual sequence. Positional encoding thus ensures the model understands the sequential structure of the input data.

✓ **Transformer Block:**

Transformer layers are integral components in the architecture of language models, playing a crucial role in both encoding and decoding information.

-Encoder-Decoder Architecture:

The transformer model utilizes an encoder-decoder architecture, consisting of encoding and decoding layers. Encoders process input tokens sequentially, generating contextualized token representations. In contrast, decoders process both encoder output and their own tokens iteratively during inference. Each encoder and decoder layer includes a feed-forward neural network, residual connections, and layer normalization steps.



-Encoder:

The encoder comprises two key components: a self-attention mechanism and a feed-forward neural network. The self-attention mechanism evaluates the relevance of input encodings to each other, creating output encodings. These output encodings, incorporating positional information and input sequence embeddings, are then passed to subsequent encoders and decoders. The bidirectional nature of the encoder allows attention on tokens both before and after the current one, using tokens to account for polysemy.

-Decoder:

Decoders consist of three essential elements: a self-attention mechanism, an attention mechanism over encodings, and a feed-forward neural network. Like encoders, the decoder processes input sequentially but introduces an encoder-decoder attention mechanism. This mechanism draws pertinent information from encodings generated by encoders. The decoder, with its own positional information and output sequence embeddings, ensures autoregressive text generation by masking the output sequence partially. The final decoder is succeeded by a linear transformation and SoftMax layer, producing output probabilities over the vocabulary.

✓ Linear Block:

To bring the model's complex internal representations into tangible predictions, the linear block, a fully connected layer, plays a vital role. It maps the vector space back to the original

input domain, generating scores (logits) for potential tokens. This decision-making layer transforms intricate representations into interpretable predictions.

✓ **SoftMax Block:**

The SoftMax function, acting as the last stage, takes these logits and transforms them into a probability distribution. Each element in the SoftMax output reflects the model's confidence in a specific class or token, simplifying the interpretation of predictions.

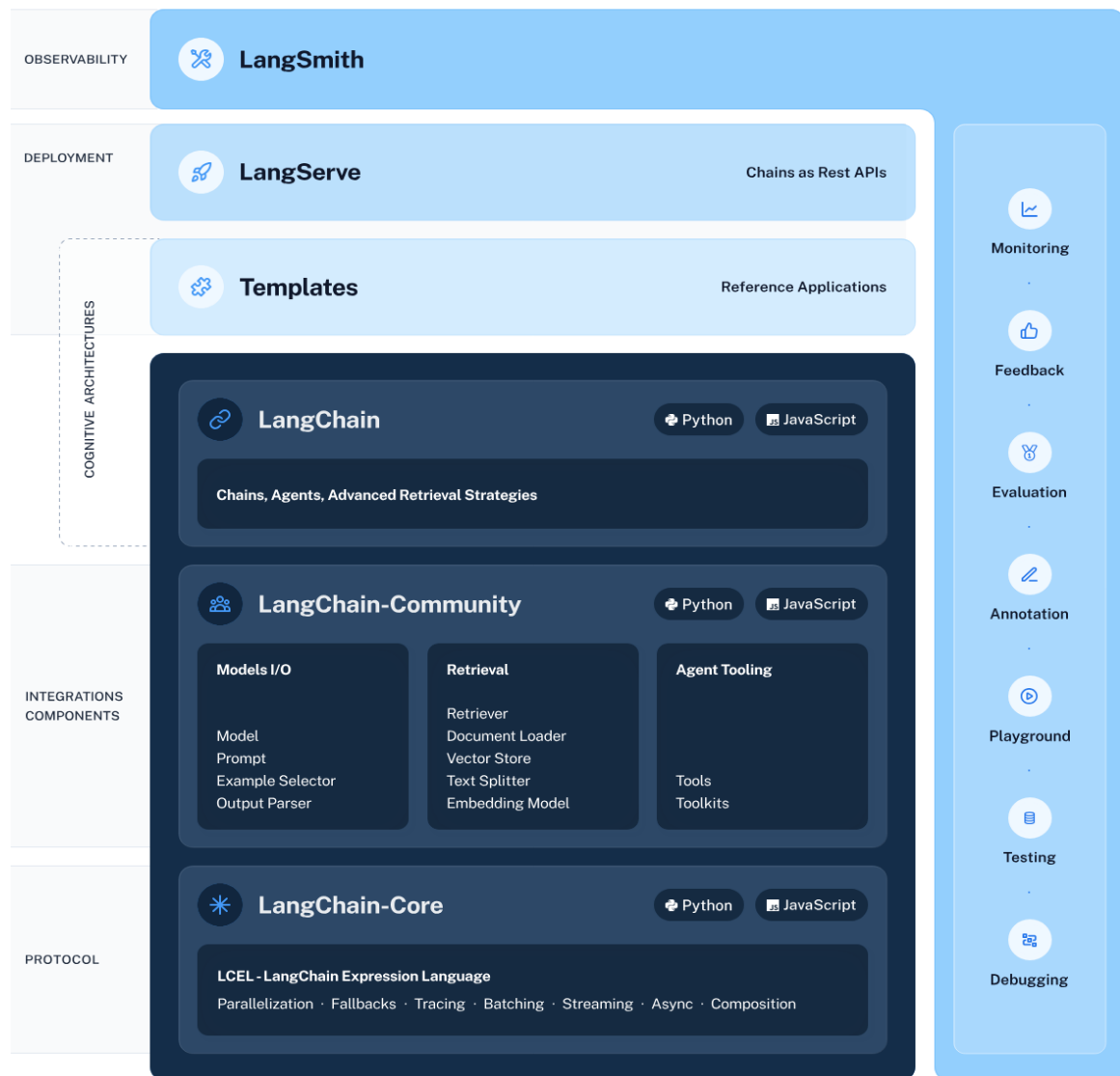
d. **Introduction to Lang chains:**

LangChain is a framework designed for crafting language model-powered applications. This framework excels in creating applications that possess two key attributes:

- ✓ **Context-Awareness:** It connects language models to various contextual sources, such as prompt instructions, few-shot examples, or relevant content, enhancing the model's responses.
- ✓ **Reasoning Abilities:** Lang Chain empowers applications to leverage language models for reasoning, aiding in decision-making based on provided context.

▪ **LangChain comprises essential components:**

- ✓ **LangChain Libraries:** These Python and JavaScript libraries provide interfaces, integrations, and a runtime for building chains and agents. They also offer pre-built implementations for various tasks.
- ✓ **LangChain Templates:** A repository of easily deployable reference architectures catering to diverse tasks.
- ✓ **LangServe:** A library facilitating the deployment of Lang Chain chains as REST APIs.
- ✓ **LangSmith:** A developer platform facilitating debugging, testing, evaluation, and monitoring of chains built on any Large Language Model (LLM) framework. It seamlessly integrates with LangChain.



- **Together, these products simplify the entire application lifecycle:**
 - ✓ **Develop:** Write applications in LangChain/LangChain.js, kickstarting your project using Templates as a reference.
 - ✓ **Productionize:** Utilize LangSmith to inspect, test, and monitor chains, ensuring continuous improvement and confident deployment.
 - ✓ **Deploy:** Transform any chain into an API effortlessly with LangServe, ensuring seamless integration into your application architecture.

- **LangChain Expression Language (LCEL):**

LCEL, a declarative composition language, simplifies chain building without code changes. Designed for prototypes to smoothly transition into production, LCEL supports a range of chains, from basic "prompt + LLM" to intricate configurations.

Explore LCEL:

- Overview: Understanding LCEL and its benefits.
- Interface: Standard LCEL object interface.
- How-to: Key features and functionalities of LCEL.
- Cookbook: Example code for common tasks.

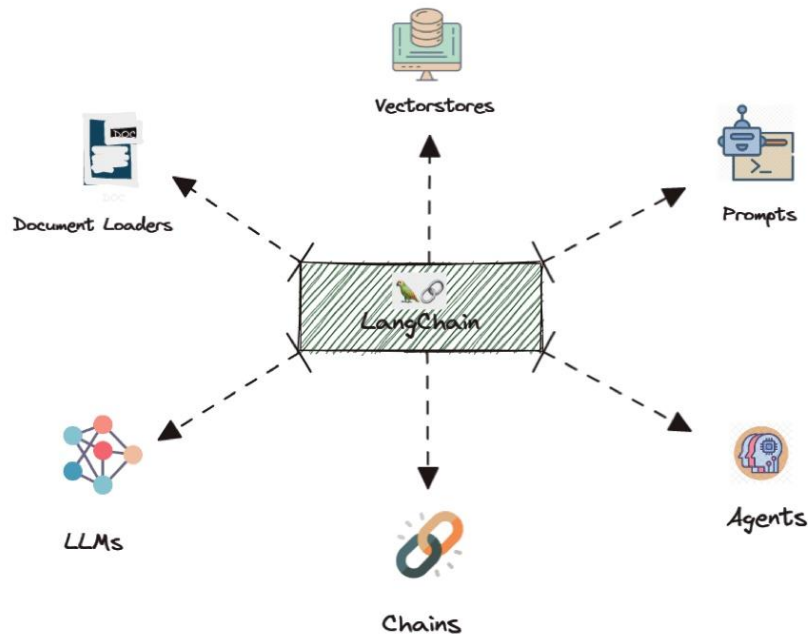
- **Modules in LangChain**

LangChain offers standard, extendable interfaces and integrations for key modules:

- Model I/O: Interface with language models.
- Retrieval: Interface with application-specific data.
- Agents: Allow models to choose tools based on high-level directives.



e. LangChain Components:



▪ Prompts in LangChain:

Prompts are like giving instructions to a language model. It is a set of directions or things you tell the model to help it respond better. This way, it understands what you are talking about and can give you useful information or have a conversation. In LangChain, there are tools to help with prompts.

Key Features:

- ✓ Prompt Templates: These are like pre-made patterns for telling the model what to do. It is like having a form where you just fill in the blanks.
- ✓ Example Selectors: These are tools to help pick out good examples to include in your prompts. It is like choosing the best examples to teach the model what you want.

▪ Large Language Models (LLMs) in LangChain:

Large Language Models (LLMs) play a crucial role in the LangChain framework. Unlike providing its own LLMs, LangChain serves as a platform that offers a standardized interface for seamless interaction with various LLM providers.

Numerous LLM providers, including OpenAI, Cohere, and Hugging Face, are compatible with LangChain. The LLM class within LangChain is specifically designed to establish a uniform interface across all these providers.

For this walkthrough, we will focus on demonstrating functionalities using an OpenAI LLM wrapper. It is important to note that the features highlighted in this demonstration are applicable to all types of LLMs integrated with LangChain.

▪ Document Loaders in LangChain:

Document loaders in LangChain serve as a fundamental component for extracting data from various sources and organizing it into structured documents. A document, in this context, refers to a piece of text along with its associated metadata.

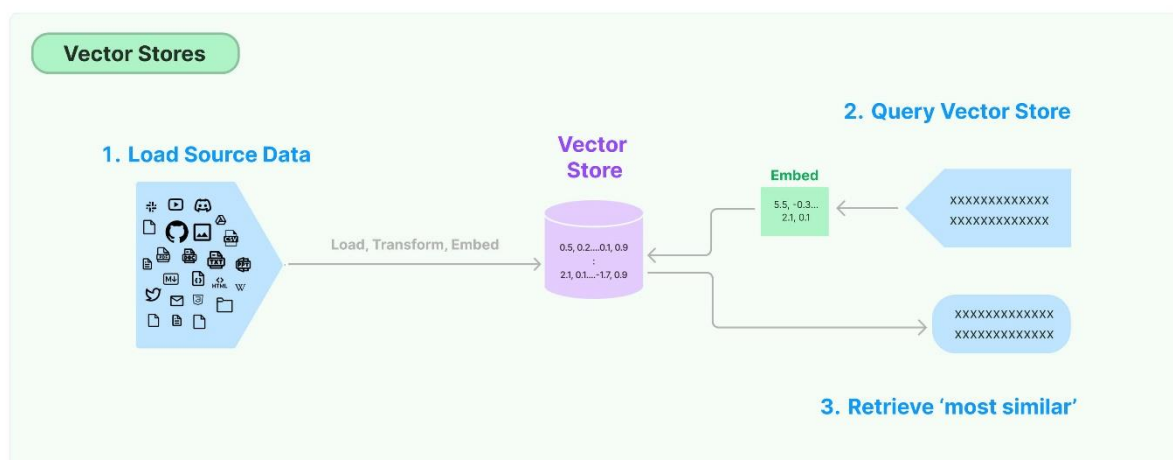
For instance, document loaders can handle diverse data sources such as simple .txt files, text contents of web pages, or transcripts of YouTube videos. They offer a standardized "load" method, allowing users to retrieve data as documents from a configured source.

Additionally, document loaders may feature a "lazy load" option, enabling the gradual loading of data into memory as needed. This flexibility ensures efficient data retrieval and processing, making document loaders a crucial element in LangChain's toolkit for information extraction.

▪ Document Transformers:

After you have brought in your documents, you might need to tweak them to fit your application better. Let us say you have a lengthy document, and you want to break it down into smaller parts that your model can handle. LangChain comes with handy built-in document transformers to help you split, merge, filter, and adjust your documents as needed. These transformers make it a breeze to get your documents ready for effective processing in your language model.

▪ Vector Stores:



In LangChain, vector stores play a crucial role in handling unstructured data. They employ a common method of embedding and storing data as vectors, enabling efficient search operations. Here is how it works:

- ✓ **Embedding Data:** Unstructured data is transformed into embedding vectors. These vectors capture the semantic and contextual information of the data, representing it in a numerical format.
- ✓ **Storage:** The embedded data is then stored in a vector store, organizing it for easy retrieval and subsequent processing. This method provides an efficient means of managing large volumes of unstructured information.
- ✓ **Vector Search:** During query time, LangChain can embed an unstructured query and search for the most similar embedding vectors in the vector store. This approach facilitates quick and accurate retrieval of relevant data.

▪ **Agents in LangChain:**

Agents serve as intelligent decision-makers in LangChain, leveraging language models to determine sequences of actions. Unlike traditional chains with hardcoded actions, agents use a language model as a reasoning engine to decide on actions dynamically.

✓ **Agent:**

- Responsible for deciding the next step in a sequence.
- Powered by a language model and a prompt.
- Inputs include tool descriptions, user objectives, and previous action-tool pairs.
- Outputs the next action(s) or the final response (AgentActions or AgentFinish).
- Different agents have distinct prompting styles, input encoding, and output parsing.

✓ **Agent Types:**

Zero-shot ReAct:

- Determines the tool to use based solely on tool descriptions.
- General-purpose action agent.

Structured input ReAct:

- Handles multi-input tools using a structured action input.
- Useful for complex tool usage, such as browser navigation.

OpenAI Functions:

- Works with fine-tuned OpenAI models (e.g., gpt-3.5-turbo-0613, gpt-4-0613).
- Explicitly detects when a function should be called.

Conversational:

- Designed for conversational settings, using ReAct framework and memory.
- Prompts are crafted to make the agent helpful and conversational.

Self-ask with search:

- Utilizes a single tool, Intermediate Answer, for factual answers to questions.
- Equivalent to the original self-ask with search paper.

ReAct document store:

- Interacts with a docstore using the ReAct framework.
- Requires Search and Lookup tools for document retrieval and term lookup.

▪ **Chains in LangChain:**

When dealing with more complex applications, utilizing a single Large Language Model (LLM) may not suffice. In such cases, chaining LLMs becomes essential, either by connecting them with each other or with other components.

LangChain offers two main frameworks for creating these chains. The traditional method involves using the Chain interface, while the modern approach employs the LangChain Expression Language (LCEL). For new applications, we recommend adopting LCEL for chain composition. However, we continue to support several pre-built Chains in both frameworks. It is worth noting that Chains can also be integrated into LCEL, providing flexibility in application design.

Understanding and implementing these chaining frameworks is crucial for developing intricate applications that demand the collaboration of multiple language models or components.

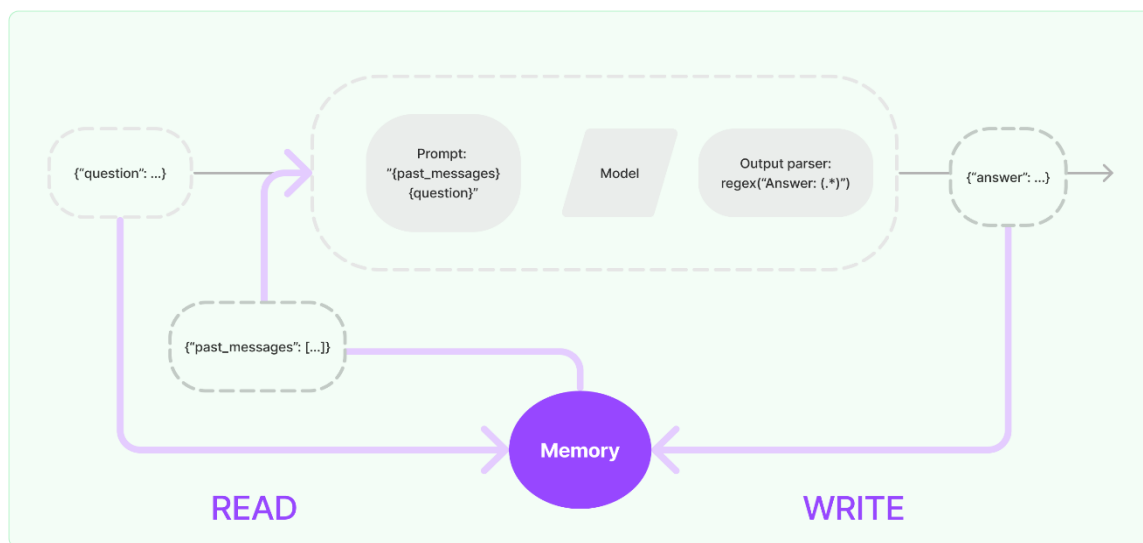
▪ **Memory in LangChain:**

In LangChain, memory is a crucial component for applications with conversational interfaces. Its role is to enable systems to refer to earlier information in a conversation. At the basic level, a conversational system should access a window of past messages. For more sophisticated systems, a constantly updating world model is essential, allowing the retention of information about entities and their relationships.

✓ Memory Functions:

LangChain offers utilities to seamlessly incorporate memory into a system. A memory system must support two fundamental actions: reading and writing. Every chain, which defines core execution logic, interacts with its memory system twice in each run.

- **Reading from Memory (Before Core Execution):** After receiving initial user inputs, a chain reads from its memory system to augment the user inputs.
- **Writing to Memory (After Core Execution):** After executing core logic but before returning the answer, a chain writes the inputs and outputs of the current run to memory. This allows referencing in future runs.



✓ Types of Memory in LangChain:

LangChain provides various memory types tailored to unique needs:

- 1. Conversation Buffer Memory:** Stores messages and extracts them when needed.
- 2. Conversation Buffer Window Memory:** Maintains a list of recent interactions, using the last K interactions to avoid overwhelming the buffer.
- 3. Entity Memory:** Remembers facts about specific entities, extracting and building knowledge over time.
- 4. Conversation Knowledge Graph Memory:** Uses a knowledge graph to recreate memory.
- 5. Conversation Summary Memory:** Creates a summary of the conversation over time, condensing information for longer interactions.
- 6. Conversation Summary Buffer Memory:** Combines recent interactions into a summary, using token length to determine when to flush interactions.
- 7. Conversation Token Buffer Memory:** Keeps a buffer of recent interactions, flushing based on token length rather than the number of interactions.

▪ Chatbots and LangChain:

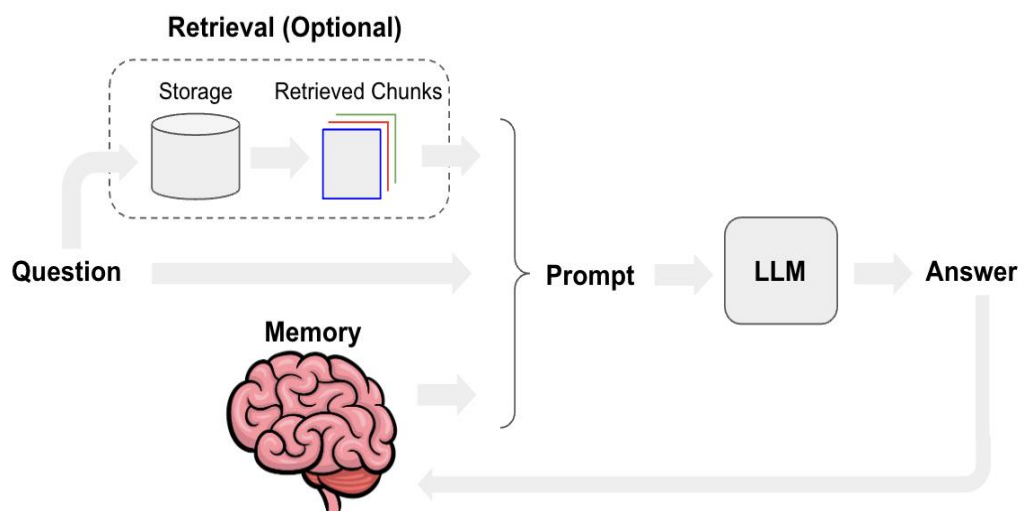
✓ Practical Application:

Chatbots stand out as a primary application of Large Language Models (LLMs). Their key attributes include the ability to engage in extended conversations and access information relevant to user inquiries.

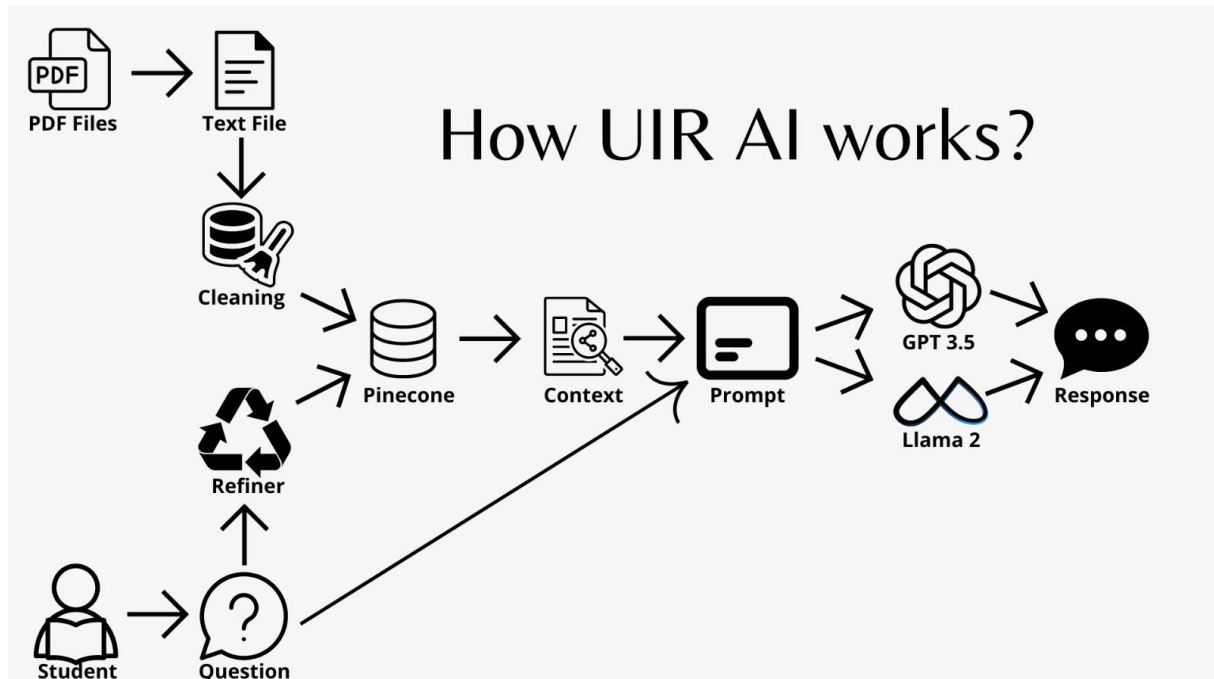
✓ Essential Elements in Chatbot Development:

Beyond fundamental prompting and LLM integration, two critical components define an effective chatbot: memory and retrieval. Memory empowers the chatbot to recall previous interactions, fostering continuity in conversations. Simultaneously, retrieval ensures the chatbot has access to the latest and domain-specific information, enriching the user experience.

Incorporating these components not only enhances the chatbot's conversational capabilities but also contributes to its adaptability and responsiveness. The synergy of these elements within the LangChain framework makes it a valuable tool for optimizing chatbot functionality and user interactions.



III. Methodology



- a) Detailed explanation of the method used for collecting and preparing UIR's data.

PyPDF2



The method used for collecting and preparing UIR's data involved a systematic process. Initially, we obtained data by downloading PDFs from the official website of UIR (www.uir.ac.ma). These PDFs contained program details from the College of Engineering and Architecture. Subsequently, we utilized the widely used Python library, PyPDF2, to extract text from these PDFs, employing its Pdfreader module.

However, due to limitations in accuracy, particularly when dealing with images within the PDFs, we performed a cleaning process on the extracted text. This involved rectifying missing words, repositioning certain words for coherence, and supplementing absent information. As the downloaded PDFs lacked some necessary details, we referenced and supplemented the missing information by cross-referencing with the web pages dedicated to each program directly from the College's website.

After refining the extracted data and incorporating the missing information, we consolidated all the gathered data into a single text file for further analysis and processing.

b) Description of the process for embedding storage using Pinecone



In this step, we utilized Google Colab, a hosted Jupyter Notebook service, to leverage its hassle-free setup and free computing resources. Initially, we installed essential libraries - Sentence_transformers, langchain, and pinecone client.

The choice of Sentence_transformers with the 'all-MiniLM-L6-v2' model for embedding was driven by its popularity and impressive speed of 14200 sentences per second. This specific model stands out for its exceptional balance between speed and quality, which is not commonly found in other models that typically prioritize either speed or quality.

For document handling, we employed langchain's document loaders and RecursiveCharacterTextSplitter from langchain.textsplitter. This text splitter is optimized for generic text, aiming to keep semantically related pieces like paragraphs, sentences, and words together. We configured it with a chunk size of 500 tokens and an overlap of 20 tokens between chunks to maintain quality within the model's 512 token limit and ensure continuity between adjacent chunks.

Post chunking the document, we generated embeddings using the 'all-MiniLM-L6-v2' model. Subsequently, we initialized the Pinecone database, a platform capable of quick similarity searches by representing data as vectors. While considering alternatives like Chroma DB, we found Pinecone to be more user-friendly, offering a free plan that suits our needs without

requiring an upgrade to a paid plan. Pinecone's real-time search capability and ease of access via API were especially advantageous for our use case, which involved multiple language models (LLMs) like llama2, GPT3.5, among others.

After storing our vectors in Pinecone, for similarity searches within our chatbot, we use the same 'all-MiniLM-L6-v2' model for embeddings. Pinecone employs cosine similarity to compare user queries with stored data vectors, allowing us to specify the number of closest neighbors we want to retrieve using the 'top-k' parameter, utilizing the k-nearest neighbors algorithm.

- c) Steps involved in integrating OpenAI GPT-3.5 Turbo and Llama 2 into the chatbot.



For GPT-3.5 Turbo, OpenAI offers a free trial plan that provides a \$5 credit for newly created accounts. We specifically chose GPT-3.5 Turbo because of its pricing structure, which charges \$0.0010 per 1K tokens for input and \$0.0020 per 1K tokens for output. Although GPT-4 is available, its pricing is notably higher at \$0.03 per 1K tokens for input and \$0.06 per 1K tokens for output.

After creating the account, we imported 'ChatOpenAI' from 'langchain.chat_models' and accessed our chat model using the following code:

```
llm = ChatOpenAI(model_name="gpt-3.5-turbo", openai_api_key="APIKEY")
```

Here, 'gpt-3.5-turbo' refers to the model's name, and 'APIKEY' represents the API key obtained from the created account's settings during the free trial.

We incorporated the 'llm' model into a chain that comprises the 'gpt-3.5-turbo' model, a prompt template, and memory to retain conversation history. Further details about these parameters will be discussed in the upcoming chapter IV.



For Llama 2, an open-source language model that does not require payment for use, we initially utilized the Llama 2 7-billion model due to limitations in Google Colab's resources (RAM, GPU, and storage). Unfortunately, even the smallest model available did not meet our requirements as it exhibited significant hallucinations and limitations.

After extensive research, we discovered Together AI, the fastest cloud platform for building and executing generative AI. While it also operates on a pay-as-you-go system, it provides a free trial plan offering \$25 credit. The platform hosts several renowned language models, including all Llama 2 and Falcon models, and functions similarly to the OpenAI API. Its pricing model charges \$0.0002 per 1000 tokens for usage.

The Python library 'together' facilitates API integration. Setting the API key is done using:

```
together.api_key = os.environ["TOGETHER_API_KEY"]
```

Accessing the Llama 2 model is achieved through:

```
together.Models.start("togethercomputer/llama-2-70b-chat").
```

Configuring our LLM involves the following code:

```
llm = TogetherLLM(
    model="togethercomputer/llama-2-70b-chat",
    temperature=0.3,
    max_tokens=512
)
```

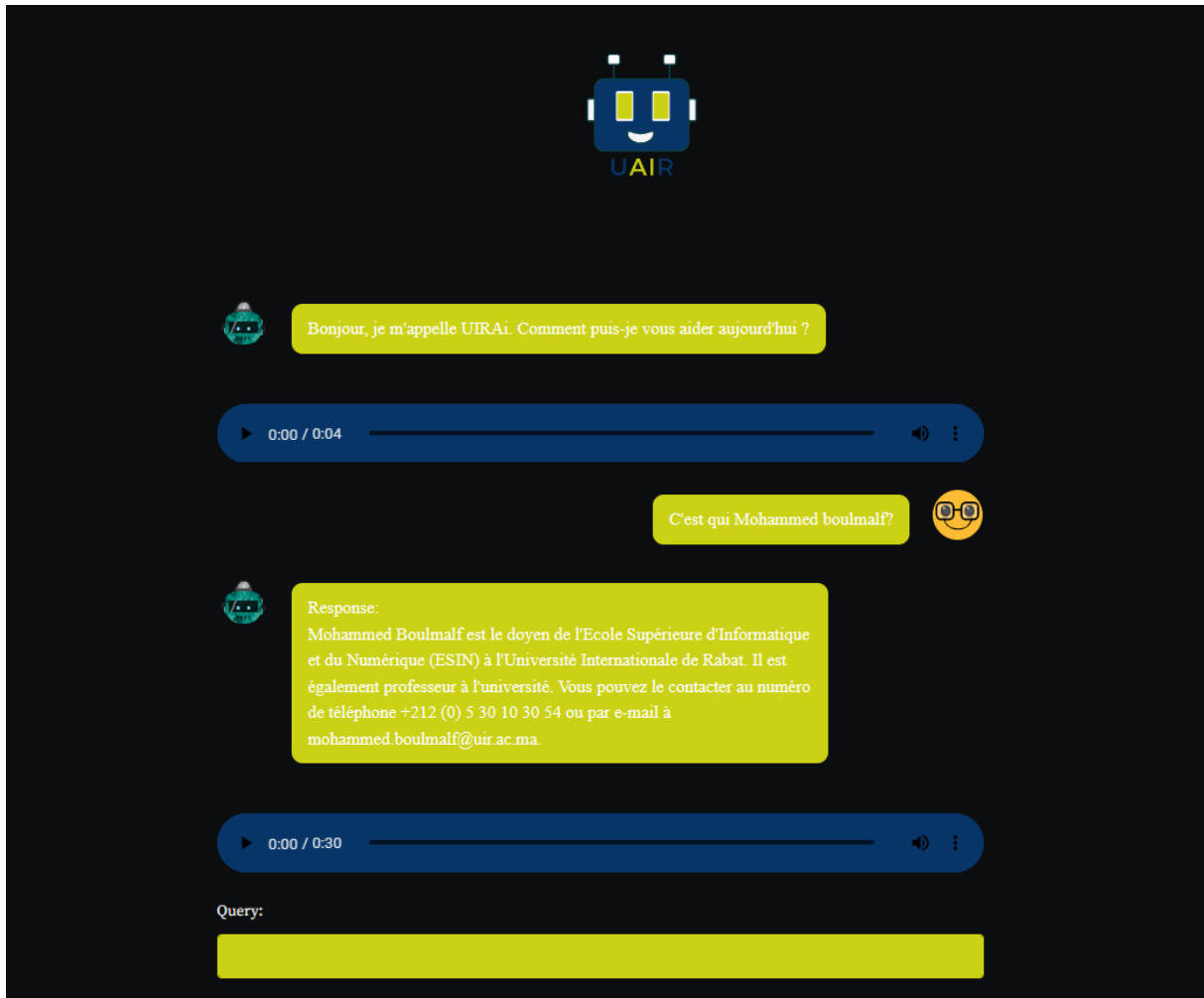
The 'temperature' parameter regulates the creativity of the bot's generated output. Higher temperatures tend to produce more diverse and creative responses.

In the upcoming chapter, we will delve into the details of the specific chain configuration utilized for Llama 2 and discuss the implementation of the prompt template.

IV. Chatbot Design and Development

a) Design (Streamlit)

1. User Interface



This is how the UI appears using Streamlit. We integrated our brand colors (yellow, blue, and white) with their corresponding hex codes. The logo was created using Canva. To customize this UI, we utilized Streamlit components and defined our theme in the **config.toml** file as follows:

1. `[theme]`
2. `primaryColor = "#073567"`
3. `backgroundColor = "#0c0d0e"`
4. `secondaryBackgroundColor = "#cbd115"`
5. `textColor = "#ffffff"`
6. `font = "serif"`

This configuration allowed us to set specific colors for primary and background elements, define the text color, and choose a serif font for the UI.

2. Streamlit Components and Their Usage:



Streamlit

- ***st.set_page_config:***

Purpose: Configures Streamlit's page settings.

Usage: Sets the page title and icon for the Streamlit app.

- ***Containers (st.container()):***

Purpose: Organizes different sections of the chatbot UI.

Usage:

- response_container: Manages the display of chatbot responses and user queries.
- textcontainer: Encapsulates the text input box and processing logic for user queries.

- ***st.image:***

Purpose: Displays an image in the Streamlit app.

Usage: Shows an image related to the chatbot, a logo or visual identifier.

- ***st.text_input:***

Purpose: Creates a text input box for users to enter queries.

Usage: Allows users to input text queries that trigger interactions with the chatbot.

- ***st.spinner:***

Purpose: Displays a spinner to indicate loading or processing.

Usage: Provides visual feedback while the chatbot processes a user query.

- ***st.code:***

Purpose: Displays code in the Streamlit app.

Usage: Likely used for debugging or displaying intermediate processing steps.

- **st.audio:**

Purpose: Renders audio playback functionality.

Usage: Plays audio generated from text-to-speech conversion for chatbot responses.

- **st.markdown:**

Purpose: Renders Markdown-formatted text in the Streamlit app.

Usage: Likely used for styling or displaying formatted text in the interface.

- **st.session_state:**

Purpose: Manages session-specific state information.

Usage: Stores and manages the history of user queries and chatbot responses during the session.

3. Streamlit's Design in the Code:

- **Structuring the UI:**

Streamlit components are used to organize different sections of the chatbot UI, such as chat history display (response_container) and the user input section (textcontainer).

- **User Interaction Handling:**

Components like st.text_input and processing logic within containers (textcontainer) handle user queries and chatbot interactions.

- **Visual Feedback and Representation:**

Utilization of spinners (st.spinner) provides visual cues during processing or loading.

- **Dynamic Content Display:**

Content such as chat messages and audio playback are dynamically updated based on user interactions and chatbot responses.

- **State Management:**

st.session_state maintains session-specific data (e.g., chat history) throughout the user's interaction with the chatbot.

b) Development (Langchain)

1) Llms

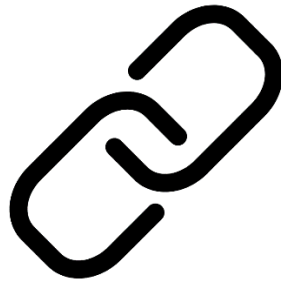


The context we utilize is derived from Pinecone, employing the same embedding technique used for storing university information in the vector database.

For our first chatbot, we utilize GPT 3.5 Turbo as an LLM. To achieve this, we imported the class ChatOpenAI from langchain.chat_models. The ChatOpenAI class offers more chat-related methods enhancing its user-friendliness when building chatbot-related applications.

For our second chatbot, we utilized the LLama 2 70b model as an LLM. To achieve this, due to limitations in both Google Colab and our computers for handling the LLama 2 70b model, we opted for Together AI (previously discussed in prior chapters). To facilitate this integration, we utilized a class called TogetherLLM. This class accepts parameters such as the endpoint of the model to be used, which is set as "togethercomputer/llama-2-70b-chat," the Together API key, and a method named "_call." The "_call" method is responsible for making API calls to the Together AI endpoint, using specified parameters to generate text based on a provided prompt. This is how we incorporated both LLama 2 (70b) and GPT 3.5 into our chatbot.

2) Chains

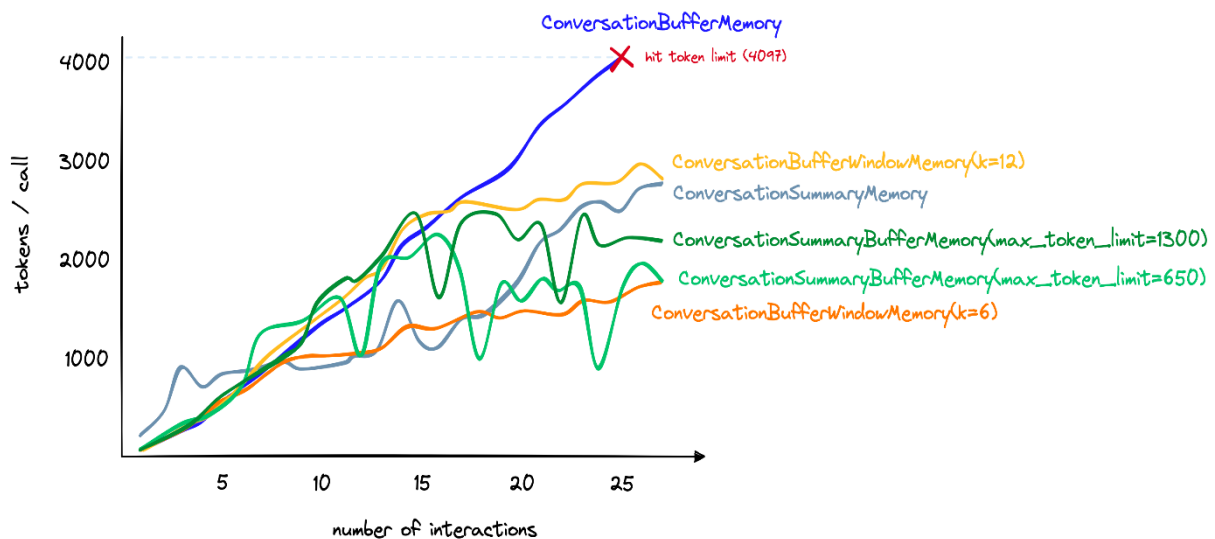


ConversationChain

After configuring our LLMs, we needed to set up our chains. There are several types of chains in langchain, but we used ConversationChain. By default, the ConversationChain has a simple type of memory that remembers all previous inputs/outputs and adds them to the context passed to the LLM. The chain takes memory as a parameter, prompt template, and LLM (already initiated by TogetherLLM for Llama 2 ChatOpenai for GPT 3.5 turbo).

3) Memory

Regarding memory, there are several types:



and we chose ConversationBufferWindowMemory because it has a parameter called K that allows us to set the exact number of previous interactions the chatbot will remember. If we chose ConversationBufferMemory, as the number of interactions increases, the number of tokens will increase, eventually reaching the limit (4096 tokens for GPT 3.5 Turbo and LLama 2 70b).

4) Prompt Template



For the prompt template, we divided it into a system prompt template and a human prompt template.

For GPT 3.5 model

- *System prompt template:*

"You are a very friendly AI bot working for Université Internationale de Rabat. Your name is UAIR, and you are just 2 months old. Remember that the user is a future student at the university you are working with, and he is providing his name in the first query, so you need to call him by his name always. Your creators are Marouane Benbrahim and Ayman Jouhari. Always answer questions in the user's language and only using the provided context. If the answer is not contained within the context below and it seems related to the university, redirect the user without adding anything to the university's website, which is www.uir.ac.ma. If The question is completely off-topic in relation to the university, say that it's not your job to answer that."

- *Human prompt template:*

we included only a variable "{input}," signifying the query the user will write.

LLama 2 70b model.

- *System prompt template:*

"<<SYS>> You are a very friendly AI bot working for Université Internationale de Rabat. Your name is UAIR, and you are just 2 months old. Remember that the user is a future student at the university you are working with. Your creators are Marouane Benbrahim and Ayman Jouhari. Always answer questions in the user's language and only using the provided context. If the answer is not contained within the context below and it seems related to the university, redirect the user without adding anything to the university's website, which is www.uir.ac.ma. If The question is completely off-topic in relation to the university, say that it's not your job to answer that."

- *The human message template:*

"<</SYS>>[INST]{input}[/INST]."

- *Final template.*

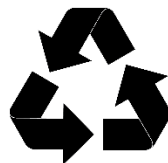
We combined both human and system templates into one template using:

```
prompt_template =
ChatPromptTemplate.from_messages([system_msg_template,
MessagesPlaceholder(variable_name="history"), human_msg_template]).
```

The MessagesPlaceholder is used to store the history on a variable called history.

To distinguish between LLama 2 prompt templates and GPT 3.5 prompt templates, LLama 2 employs <<SYS>><</SYS>> and [INST]/[/INST]. Within the <<SYS>> brackets, the prompt for the system should be placed, while inside the [INST] brackets, the prompts for the users are inserted. Failure to include these distinctions for LLama 2 may result in the model's inability to discern between user and system prompts. After conducting several research, we discovered that this differentiation arises from the training methodology of LLama 2, contrasting with GPT 3.5, which inherently comprehends the distinctions between user and system prompts.

5) Query Refiner



We have implemented an integrated function called 'query refiner,' aimed at enhancing user queries, particularly addressing grammatical errors to ensure their correctness. This function is especially crucial when conducting similarity searches on Pinecone, significantly improving the precision of these searches.

For this purpose, we utilized OpenAI's text-davinci-003 model for GPT-3.5 and the llama 2 13b model for llama 2 70b. These models were instructed to formulate a relevant French question based on a given student query and conversation log. The text-davinci-003 model, known for its newer and more adept capabilities, excels in instruction-following tasks and delivers accurate responses, even in zero-shot scenarios. It is also cost-effective compared to the GPT 3.5 LLM and performs well in tasks like translation and text correction.

Despite Pinecone storing data in French, the prompt guided the models to craft questions in French. However, in our initial LLM prompts, we specified that if the query is in French, the response should be in the user's language. The 'query refiner' function plays a pivotal role in extracting context from Pinecone, refining this context before providing it to the initial LLMs (LLama 2 and GPT) for further processing.

6) gTTS



We imported gTTS (Google Text-to-Speech), a Python library and CLI tool that interfaces with the Google Translate text-to-speech API. This provides us with an option to listen to the chatbot's response using the voice of Google Translator

V. Evaluations, results, and Analysis

a) GPT Builder

For our analysis, we include GPT Builder for comparison with GPT-3.5 Turbo and Llama GPT Builder, developed by OpenAI, enables users to create custom ChatGPT versions (GPTs) tailored to specific needs. It is user-friendly and requires no coding skills. Users can customize GPTs for personal, company, or public use, defining functionalities such as web search, image creation, or data analysis. GPT Builder is available through a paid plan. We provided it with the same data and prompt templates as GPT-3.5 and Llama 2.



b) Evaluation Procedure:

- **Question Set:** Prepare a set of 10 questions covering diverse topics and complexity levels.
- **Chatbot Interaction:** Ask each chatbot the same set of questions, recording their responses.

Metrics Evaluation:

- **Accuracy of Responses:** Judge the correctness and relevance of answers.
 - **Response Time:** Record the time taken by each chatbot to respond to each question.
 - **Understanding of Context:** Assess how well the chatbots maintain the conversation flow and context.
 - **Handling Ambiguity:** Observe how each chatbot deals with unclear questions or requests for clarification.
 - **User Satisfaction:** Collect feedback from students on their satisfaction level with each chatbot's responses.
 - **Language and Grammar:** Analyze responses for grammar, language style, and appropriateness.
- **Scoring:** Use a scoring system (e.g., from 1 to 10) for each metric.
- **Average Score Calculation:** Sum up the scores for each metric for all chatbots and divide by the number of metrics to calculate the average score for each chatbot.
- **Judging by GPT-4 (ChatGPT Online Platform):** Utilize GPT-4 to provide a score for each metric.

c) Presentation of achieved results.

We created 10 entirely distinct questions and directed each of the models (Llama 2, GPT-3.5 Turbo, GPT Builder) to answer them.



Questions:

1. Quels sont les critères d'admission à l'École supérieure d'architecture de Rabat (ESAR) ?
2. Quelles les partenariats académiques de l'école d'ingénierie aérospatiale et automobile (SAAE) ?
3. Quels sont les débouchés de l'école d'ingénierie aérospatiale et automobile (SAAE) ?
4. Pouvez-vous me donner des informations sur le diplôme d'architecte ?
5. Quels sont les frais de formation de l'école Supérieure d'Architecture de Rabat (ESAR) ?
6. Qui est Mohammed Boulmalf ?
7. Pouvez-vous m'expliquer le schéma des études de l'École Supérieure de l'Ingénierie de l'Énergie (ECINE) ?
8. Pouvez-vous me donner les Coordonnées bancaires de l'UIR ?
9. Pourriez-vous me donner la présentation générale de l'École d'ingénierie aérospatiale et automobile (SAAE) ?
10. Esin ?

Question 1 :

QUELS SONT LES CRITÈRES D'ADMISSION À L'ÉCOLE SUPÉRIEURE D'ARCHITECTURE DE RABAT (ESAR)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	4	3	7
RESPONSE TIME	8	9	2
UNDERSTANDING OF CONTEXT	4	3	8
HANDLING AMBIGUITY	6	5	7
USER SATISFACTION	5	4	7
LANGUAGE AND GRAMMAR	9	9	9
AVERAGE SCORE (GPT 4)	6	5.5	6.67

1. Accuracy of Responses:

GPT 3.5: 4/10 - The response lacks key details from the provided context, missing specific steps in the application process and fee structure.

Llama 2: 3/10 - Like GPT 3.5, it gives a detailed but inaccurate response, not aligning with the specific provided data.

GPT Builder: 7/10 - More accurate in terms of the admission process, closely aligning with the context but still missing some details.

2. Response Time:

GPT 3.5: 8/10 - Fast with a 7.63-second response time.

Llama 2: 9/10 - Slightly faster than GPT 3.5 with a 7.44-second response time.

GPT Builder: 2/10 - Significantly slower with a 61.70-second response time.

3. Understanding of Context:

GPT 3.5: 4/10 - Partially understands the context but fails to mention specific steps and the fee structure correctly.

Llama 2: 3/10 - Shows an understanding of the general context but lacks depth in specific details.

GPT Builder: 8/10 - Demonstrates better grasp of the context, although not perfect.

4. Handling Ambiguity:

GPT 3.5: 6/10 - Provides a structured answer but does not fully align with the given data.

Llama 2: 5/10 - Addresses the question but misses specific details from the data.

GPT Builder: 7/10 - Manages to provide a more relevant response to the provided data.

5. User Satisfaction:

GPT 3.5: 5/10 - The response is structured and clear but inaccurate in details, which might affect user satisfaction.

Llama 2: 4/10 - While the response is clear, the inaccuracies might lead to lower satisfaction.

GPT Builder: 7/10 - Despite the slower response, the detailed and accurate information provided could lead to higher satisfaction.

6. Language and Grammar:

GPT 3.5: 9/10 - Well-structured language and correct grammar.

Llama 2: 9/10 - Good language use and grammar.

GPT Builder: 9/10 - Well-written response with good language and grammar.

Question 2 :

QUELLES LES PARTENARIATS ACADÉMIQUES DE L'ÉCOLE D'INGÉNIERIE AÉROSPATIALE ET AUTOMOBILE (SAAE)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	8	7	9
RESPONSE TIME	6	8	2
UNDERSTANDING OF CONTEXT	8	7	9
HANDLING AMBIGUITY	8	7	8
USER SATISFACTION	8	7	9
LANGUAGE AND GRAMMAR	9	8	9
AVERAGE SCORE (GPT 4)	7.85	7.4	7.6

1. Accuracy of Responses:

- GPT 3.5: 8/10 (Provides correct partners and additional details about the partnership and its benefits.)
- LLaMA 2: 7/10 (Correctly identifies partners but lacks the additional detail provided by GPT 3.5.)
- GPT Builder: 9/10 (Provides detailed and accurate information about the partners, specifically mentioning the university's name and the nature of the partnership.)

2. Response Time:

- GPT 3.5: 6/10 (5.18 seconds)
- LLaMA 2: 8/10 (3.9 seconds)
- GPT Builder: 2/10 (57.23 seconds)

3. Understanding of Context:

- GPT 3.5: 8/10 (Correctly identifies the partners and their relevance, showing good contextual understanding.)
- LLaMA 2: 7/10 (Identifies partners but does not show as much depth in contextual understanding as GPT 3.5.)
- GPT Builder: 9/10 (Demonstrates an excellent understanding of the context by providing specific and relevant details.)

4. Handling Ambiguity:

- GPT 3.5: 8/10 (Effectively addresses the question without needing further clarification.)
- LLaMA 2: 7/10 (Addresses the question well but with less detail than GPT 3.5.)
- GPT Builder: 8/10 (Provides a clear and detailed response, showing good handling of potential ambiguity.)

5. User Satisfaction:

- GPT 3.5: 8/10 (The response is informative and relevant, leading to high user satisfaction.)
- LLaMA 2: 7/10 (Adequate response, but slightly less detailed, which might affect user satisfaction.)
- GPT Builder: 9/10 (Despite the slower response, the detailed and accurate information provided would lead to high user satisfaction.)

6. Language and Grammar:

- GPT 3.5: 9/10 (Well-structured response with appropriate language and grammar.)
- LLaMA 2: 8/10 (Clear and grammatically correct response, but less detailed.)
- GPT Builder: 9/10 (Excellent language use and grammar, with a detailed and well-structured response.)

Question 3 :

QUELS SONT LES DÉBOUCHÉS DE L'ÉCOLE D'INGÉNIERIE AÉROSPATIALE ET AUTOMOBILE (SAAE)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	9	8	10
RESPONSE TIME	8	9	5
UNDERSTANDING OF CONTEXT	9	8	10
HANDLING AMBIGUITY	9	8	9
USER SATISFACTION	8	7	9
LANGUAGE AND GRAMMAR	9	9	9
AVERAGE SCORE (GPT 4)	8.7	8.2	8.7

1. Accuracy of Responses:

- GPT 3.5: 9/10. The response is accurate and aligns well with the context provided, but it misses the mention of "Compagnies pétrolières" for the automobile sector.
- LLaMA 2: 8/10. The response is generally accurate but adds a statement about the school's mission which is not related to the career opportunities asked.
- GPT Builder: 10/10. This response is fully accurate and aligns completely with the provided context.

2. Response Time:

- GPT 3.5: 8/10. Response time is good, but not the fastest.
- LLaMA 2: 9/10. The fastest response among the three.
- GPT Builder: 5/10. Significantly slower than the others.

3. Understanding of Context:

- GPT 3.5: 9/10. Shows good understanding of the context but lacks some details.
- LLaMA 2: 8/10. Good understanding, but the addition of unrelated information slightly detracts.
- GPT Builder: 10/10. Excellent understanding of the context with complete alignment to the provided data.

4. Handling Ambiguity:

- GPT 3.5: 9/10. The response is clear and to the point.
- LLaMA 2: 8/10. Clear, but includes some additional information that may not be necessary.
- GPT Builder: 9/10. Direct and focused on the question without ambiguity.

5. User Satisfaction:

- GPT 3.5: 8/10. Informative and clear, but slightly less detailed.
- LLaMA 2: 7/10. While informative, the extra information might confuse or be irrelevant to some users.
- GPT Builder: 9/10. Detailed and directly aligned with the provided data, leading to high user satisfaction.

6. Language and Grammar:

- GPT 3.5: 9/10. Well-structured and grammatically correct.
- LLaMA 2: 9/10. Good language use, though the additional information might slightly detract.
- GPT Builder: 9/10. Good language and grammar, appropriate for the context.

Question 4 :

POUVEZ-VOUS ME DONNER DES INFORMATION SUR LE DIPLOME D'ARCHITECT?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	9	10	8
RESPONSE TIME	7	10	4
UNDERSTANDING OF CONTEXT	8	10	7
HANDLING AMBIGUITY	9	9	9
USER SATISFACTION	8	10	7
LANGUAGE AND GRAMMAR	9	9	9
AVERAGE SCORE (GPT 4)	8.4	9.7	7.4

1. Accuracy of Responses

- GPT 3.5: The response is accurate, detailed, and directly addresses the diploma's recognition and the focus of the program. Score: 9/10
- Llama 2: The response is accurate and includes additional details about the career opportunities and further studies, which aligns well with the context. Score: 10/10
- GPT Builder: Accurate response with a specific reference to the decree, but less detail about the program compared to others. Score: 8/10

2. Response Time

- GPT 3.5: 10.86 seconds - Moderately fast. Score: 7/10
- Llama 2: 6.5 seconds - Fastest among the three. Score: 10/10
- GPT Builder: 30.73 seconds - Significantly slower. Score: 4/10

3. Understanding of Context

- All responses understand the context of the question about the architecture diploma at ESAR. However, Llama 2 provided more comprehensive information relevant to the context.
- GPT 3.5: Score: 8/10
- Llama 2: Score: 10/10
- GPT Builder: Score: 7/10

4. Handling Ambiguity

- The question was straightforward without much ambiguity, so all did well in this regard.
- GPT 3.5: Score: 9/10
- Llama 2: Score: 9/10
- GPT Builder: Score: 9/10

5. User Satisfaction

- GPT 3.5: Detailed and informative. Score: 8/10
- Llama 2: Most comprehensive and informative. Score: 10/10
- GPT Builder: Less detailed, which might not satisfy users looking for more information. Score: 7/10

6. Language and Grammar

- All responses seem well-structured and grammatically correct in French.
- GPT 3.5: Score: 9/10
- Llama 2: Score: 9/10
- GPT Builder: Score: 9/10

Question 5 :

QUELS SONT LES FRAIS DE FORMATION DE L'ÉCOLE SUPÉRIEURE D'ARCHITECTURE DE RABAT (ESAR)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	10	10	10
RESPONSE TIME	7	8	4
UNDERSTANDING OF CONTEXT	10	5	10
HANDLING AMBIGUITY	10	5	10
USER SATISFACTION	9	6	8
LANGUAGE AND GRAMMAR	10	9	10
AVERAGE SCORE (GPT 4)	9.4	7.2	8.6

1. Accuracy of Responses:

- GPT 3.5: 10/10 - Provided accurate information and added a suggestion to visit the university's website for more details.
- Llama 2: 10/10 - Accurately provided the fee details. However, it included additional information about Mohamed Boulmalf, which was not requested.
- GPT Builder: 10/10 - Accurately responded with the fees.

2. Response Time:

- GPT 3.5: 7/10 - Reasonable response time (4.41 seconds).
- Llama 2: 8/10 - Faster response time (3.33 seconds).
- GPT Builder: 4/10 - Significantly slower response time (19.35 seconds).

3. Understanding of Context:

- GPT 3.5: 10/10 - Fully understood and responded to the context.
- Llama 2: 5/10 - Responded correctly but included unrelated information, indicating a partial misunderstanding of the context.
- GPT Builder: 10/10 - Understood and responded accurately to the context.

4. Handling Ambiguity:

- GPT 3.5: 10/10 - Handled the direct question well with no ambiguity.
- Llama 2: 5/10 - The inclusion of unrelated information suggests issues with handling the specific query.
- GPT Builder: 10/10 - Directly answered the question without ambiguity.

5. User Satisfaction:

- GPT 3.5: 9/10 - High satisfaction due to accurate and helpful information, but the response time could affect the rating slightly.
- Llama 2: 6/10 - The additional, unsolicited information about Mohamed Boulmalf might confuse users.
- GPT Builder: 8/10 - Accurate and to the point, but the slow response time could affect user satisfaction.

6. Language and Grammar:

- GPT 3.5: 10/10 - Fluent and grammatically correct.
- Llama 2: 9/10 - Good language and grammar, but the inclusion of unrelated information affects this score.
- GPT Builder: 10/10 - Language and grammar were on point.

Question 6 :

QUI EST MOHAMMED BOULMALF ?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	7	7	8
RESPONSE TIME	9	9	5
UNDERSTANDING OF CONTEXT	7	7	8
HANDLING AMBIGUITY	8	8	7
USER SATISFACTION	7	7	8
LANGUAGE AND GRAMMAR	9	9	8
AVERAGE SCORE (GPT 4)	7.84	7.84	7.34

1. Accuracy of Responses:

- GPT 3.5: 7/10 (Provides detailed information but includes incorrect role and contact information).
- Llama 2: 7/10 (Similar to GPT 3.5, detailed but with some inaccuracies).
- GPT Builder: 8/10 (More accurate about the role but lacks contact details).

2. Response Time:

- GPT 3.5: 9/10 (3.47 seconds is quite fast).
- Llama 2: 9/10 (3.45 seconds is similarly fast).
- GPT Builder: 5/10 (14.75 seconds is slower).

3. Understanding of Context:

- GPT 3.5: 7/10 (Provides a response that seems relevant but includes inaccuracies).
- Llama 2: 7/10 (Similar to GPT 3.5 in context understanding).
- GPT Builder: 8/10 (Seems to focus more on the role, which is closer to the context data).

4. Handling Ambiguity:

- GPT 3.5: 8/10 (Addresses the question directly with detailed information).
- Llama 2: 8/10 (Similar approach to GPT 3.5).
- GPT Builder: 7/10 (Less detailed but still addresses the question).

5. User Satisfaction:

- GPT 3.5: 7/10 (Good, but the inaccuracies might affect user satisfaction).
- Llama 2: 7/10 (Similar to GPT 3.5).
- GPT Builder: 8/10 (Less detail but more accurate role description).

6. Language and Grammar:

- GPT 3.5: 9/10 (Clear and well-structured response).
- Llama 2: 9/10 (clear and well-structured).
- GPT Builder: 8/10 (Clear but less detailed and slightly less engaging).

Question 7 :

POUVEZ-VOUS M'EXPLIQUER LE SCHÉMA DES ÉTUDES DE L'ÉCOLE SUPÉRIEURE DE L'INGÉNIERIE DE L'ÉNERGIE (ECINE)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	7	6	9
RESPONSE TIME	9	7	4
UNDERSTANDING OF CONTEXT	7	6	9
HANDLING AMBIGUITY	8	8	9
USER SATISFACTION	8	7	9
LANGUAGE AND GRAMMAR	10	10	10
AVERAGE SCORE (GPT 4)	8.2	7.34	8.34

1. Accuracy of Responses:

- GPT 3.5: The response is mostly accurate but misses some details about the two distinct cycles (preparatory and engineering) and specific elements like stages and dual degree options. Score: 7/10
- LLaMA 2: This response includes relevant details but lacks the depth and specifics about the two cycles and stages. Score: 6/10
- GPT Builder: This response is quite accurate, mentioning the two distinct cycles and their focus, aligning well with the provided context data. Score: 9/10

2. Response Time:

- GPT 3.5: 7.03 seconds. Score: 9/10
- LLaMA 2: 9.90 seconds. Score: 7/10
- GPT Builder: 38.15 seconds. Score: 4/10

3. Understanding of Context:

- GPT 3.5: Shows a good understanding but misses some nuances of the program structure. Score: 7/10
- LLaMA 2: Adequate understanding but generalized. Score: 6/10
- GPT Builder: Incredibly good understanding, capturing the essence of the two cycles. Score: 9/10

4. Handling Ambiguity:

- GPT 3.5: Provided a structured answer without needing further clarification. Score: 8/10
- LLaMA 2: Similarly handled the query well without seeking clarifications. Score: 8/10
- GPT Builder: Detailed response, addressing potential ambiguities in the question. Score: 9/10

5. User Satisfaction:

- GPT 3.5: Likely to satisfy most users with its balance of detail and clarity. Score: 8/10
- LLaMA 2: Might leave some users wanting more specific information. Score: 7/10
- GPT Builder: Detailed and thorough, potentially very satisfying for users seeking in-depth information. Score: 9/10

6. Language and Grammar:

All responses are in French and seem grammatically correct and well-structured.

- GPT 3.5: Clear and correct language usage. Score: 10/10
- LLaMA 2: Similarly clear and grammatically sound. Score: 10/10
- GPT Builder: Well-structured and grammatically accurate. Score : 10/10

Question 8 :

POUVEZ-VOUS ME DONNER LES COORDONNÉES BANCAIRES DE L'UIR?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	10	10	10
RESPONSE TIME	9	10	5
UNDERSTANDING OF CONTEXT	10	10	10
HANDLING AMBIGUITY	10	10	10
USER SATISFACTION	9	9	8
LANGUAGE AND GRAMMAR	10	10	10
AVERAGE SCORE (GPT 4)	9.7	9.84	8.84

1. Accuracy of Responses

- GPT 3.5: The response is accurate and matches the context data. Score: 10/10
- LLaMA 2: The response is accurate and closely aligns with the context data. Score: 10/10
- GPT Builder: Accurate response, in line with the provided context data. Score: 10/10

2. Response Time

- GPT 3.5: 4.81 seconds. Fast. Score: 9/10
- LLaMA 2: 3.40 seconds. The fastest among the three, indicating a better performance in this aspect. Score: 10/10
- GPT Builder: 19.16 seconds. Significantly slower compared to the others. Score: 5/10

3. Understanding of Context

- GPT 3.5: The response shows good understanding of the context. Score: 10/10
- LLaMA 2: Shows a clear understanding of the context. Score: 10/10
- GPT Builder: Demonstrates an understanding of the context like the other two. Score: 10/10

4. Handling Ambiguity

- Since the question is straightforward and lacks ambiguity, this metric might not be as relevant for this specific query. However, based on their responses:
- GPT 3.5: Handled the direct question well. Score: 10/10
- LLaMA 2: Also handled direct questions effectively. Score: 10/10
- GPT Builder: Appropriately responded to the direct question. Score: 10/10

5. User Satisfaction

- GPT 3.5: Clear and detailed response, likely to satisfy users. Score: 9/10
- LLaMA 2: Also, clear, and detailed, with a friendly tone. Score: 9/10
- GPT Builder: Clear, but slower response time might affect user satisfaction. Score: 8/10

6. Language and Grammar

- GPT 3.5: Effective use of language and grammar. Score: 10/10
- LLaMA 2: Good language use and grammar, with a slightly more conversational tone. Score: 10/10
- GPT Builder: Language and grammar are appropriate. Score: 10/10

Question 9:

POURRIEZ-VOUS ME DONNER LA PRÉSENTATION GÉNÉRALE DE L'ÉCOLE D'INGÉNIERIE AÉROSPATIALE ET AUTOMOBILE (SAAE)?			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	9	8	7
RESPONSE TIME	8	8	5
UNDERSTANDING OF CONTEXT	9	8	7
HANDLING AMBIGUITY	9	8	7
USER SATISFACTION	9	8	7
LANGUAGE AND GRAMMAR	10	10	10
AVERAGE SCORE (GPT 4)	9	8.4	7.2

1. Accuracy of Responses:

- GPT 3.5: 9/10. The response is comprehensive and aligns closely with the context provided, covering key aspects of the SAAE.
- LLaMA 2: 8/10. The response is accurate and relevant but slightly less detailed compared to GPT 3.5.
- GPT Builder: 7/10. While accurate, the response focuses more on partnerships and less on the overall presentation of SAAE, missing some context details.

2. Response Time:

- GPT 3.5: 8/10. Good response time at 7.23 seconds.
- LLaMA 2: 8/10. Like GPT 3.5 with a 7.3-second response time.
- GPT Builder: 5/10. Significantly slower at 25.25 seconds.

3. Understanding of Context:

- GPT 3.5: 9/10. Demonstrates a good understanding of the context.
- LLaMA 2: 8/10. Shows understanding but with slightly less depth than GPT 3.5.
- GPT Builder: 7/10. Understands the context but focuses more on specific aspects (partnerships) rather than a general overview.

4. Handling Ambiguity:

- Since the question was quite specific, it is hard to evaluate how well each chatbot handles ambiguity in this case. However, based on their ability to address the specifics of the question:

- GPT 3.5: 9/10
- LLaMA 2: 8/10
- GPT Builder: 7/10


5. User Satisfaction:

- GPT 3.5: 9/10. Likely to satisfy users with its detailed and relevant response.
- LLaMA 2: 8/10. Provides a clear and accurate response, though slightly less detailed.
- GPT Builder: 7/10. While informative, the focus on partnerships might not fully satisfy users looking for a general overview.


6. Language and Grammar:

- GPT 3.5: 10/10. No noticeable language or grammar issues.
- LLaMA 2: 10/10. Similarly, no language or grammar issues.
- GPT Builder: 10/10. Well-written with proper language and grammar

Question 10 :



ESIN?



LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	8	9	8
RESPONSE TIME	9	10	4
UNDERSTANDING OF CONTEXT	8	9	7
HANDLING AMBIGUITY	7	8	7
USER SATISFACTION	8	9	7
LANGUAGE AND GRAMMAR	9	9	6.84
AVERAGE SCORE (GPT 4)	8.2	9	7.5

1. Accuracy of Responses:

- GPT 3.5: 8/10. The response is accurate, mentioning various roles and positions that align with the data provided.
- Llama 2: 9/10. It provides a comprehensive overview, touching on the mission, vision, and academic aspects, which are well-aligned with the provided context.
- GPT Builder: 8/10. The response is detailed and aligns well with the key aspects of ESIN, including its mission, vision, and academic offerings.

2. Response Time:

- GPT 3.5: 9/10. With a response time of 5.23 seconds, it is quite efficient.
- Llama 2: 10/10. The fastest response was at 4.49 seconds.
- GPT Builder: 4/10. A significantly longer response time of 65.55 seconds.

3. Understanding of Context:

- GPT 3.5: 8/10. It provides relevant information but lacks some depth compared to Llama 2.
- Llama 2: 9/10. It shows a good understanding of the context by addressing various aspects of ESIN, including its departments and partnerships.
- GPT Builder: 7/10. While informative, it seems slightly more generic and less tailored to the specific context of ESIN.

4. Handling Ambiguity:

- GPT 3.5: 7/10. It provides a general but relevant response without asking for clarification.
- Llama 2: 8/10. It effectively addresses potential ambiguities by offering a broad overview.
- GPT Builder: 7/10. Like GPT 3.5, it gives a general response but does not address ambiguity directly.

5. User Satisfaction:

- GPT 3.5: 8/10. The response is helpful and informative, leading to user satisfaction.
- Llama 2: 9/10. Its detailed and contextual response would satisfy most users.
- GPT Builder: 7/10. While informative, the long response time might affect user satisfaction.

6. Language and Grammar:

- GPT 3.5: 9/10. The response is well-written with proper language and grammar.
- Llama 2: 9/10. Also demonstrates excellent language and grammar usage.
- GPT Builder: 8/10. Good language use, but slightly less natural or conversational compared to the others.

d) Final Score

FINAL SCORE			
LLMS	GPT3.5	LLAMA2	GPT BUILDER
ACCURACY OF RESPONSES	8.1	7.8	8.6
RESPONSE TIME	8	8.8	4
UNDERSTANDING OF CONTEXT	8	7.2	8.5
HANDLING AMBIGUITY	8.4	7.6	8.3
USER SATISFACTION	7.9	7.4	7.9
LANGUAGE AND GRAMMAR	9.4	9.2	9.08
FINAL SCORE	8.3	8	7.73

- **Accuracy of Responses:** GPT Builder leads with a score of 8.6, followed by GPT-3.5 (8.1) and LLama 2 (7.8). This suggests GPT Builder is generally more accurate in its responses.
- **Response Time:** LLama 2 is the fastest (8.8), while GPT Builder is significantly slower (4). GPT-3.5 is moderately quick (8).
- **Understanding of Context:** GPT Builder scores highest (8.5), indicating its superior understanding of context, followed by GPT-3.5 (8) and LLama 2 (7.2).
- **Handling Ambiguity:** GPT-3.5 excels (8.4), closely followed by GPT Builder (8.3) and LLama 2 (7.6).
- **User Satisfaction:** GPT-3.5 and GPT Builder are tied (7.9), with LLama 2 slightly behind (7.4).
- **Language and Grammar:** GPT-3.5 leads (9.4), followed closely by LLama 2 (9.2) and GPT Builder (9.08).

Final Scores: GPT-3.5 averages the highest overall score (8.3), followed by LLama 2 (8) and GPT Builder (7.73).

Conclusion: GPT-3.5 emerges as the most balanced model, excelling in language and grammar, handling ambiguity, and user satisfaction. While GPT Builder shows strength in accuracy and context understanding, its slow response time is a notable drawback. LLama 2, despite its fast response time, lags slightly in accuracy, context understanding, and user satisfaction. Therefore, GPT-3.5 is the best all-rounder, but the choice between these models may depend on specific requirements such as speed, accuracy, or contextual understanding.

e) Strengths and Limitations:

GPT-3.5 Turbo:

- **Strengths:**

- Powerful capabilities in generating high-quality responses.
- Access to a vast amount of pre-trained data (175 billion parameters).
- Availability of a refined and diverse dataset until September 2021.
- Suitable for a wide range of tasks and applications.

- **Limitations:**

- Restricted by a token limit of 4096, limiting context length.
- Requires payment for unrestricted usage.
- Not updated with the most recent data after September 2021.

LLama 2 (70b):

- **Strengths:**

- Open-source and free for use, offering accessibility to all users.
- Access to a considerable dataset despite the parameter size (70 billion).
- Provides access to recent data, updated until September 2022.
- Suitable for various applications with appropriate hardware.

- **Limitations:**

- Relies on highly advanced hardware to efficiently run the model.
- Token limit of 4096, like GPT-3.5 Turbo, constraining context length.
- While it has the advantage of recent data, it is still limited by its parameter size compared to GPT-3.5 Turbo.

- **Collaborative Approach:**

- Utilizing Together AI, which hosts the largest models, addresses hardware constraints.

Overall Assessment:

LLMS	GPT3.5 	LLAMA2 
OPEN SOURCE		
PARAMETER SIZE	175B	70B
CONTEXT LENGTH	4096 TOKEN	4096 TOKEN
LATEST DATA TRAINING	SEPTEMBER 2021	SEPTEMBER 2022
TRAINED DATA	570 GB	UNKNOWN

Very Important: For GPT-3.5 and Llama 2, the maximum token limit of 4096 applies to the total combined tokens of both the input prompt and the model's generated output. The input text itself could be shorter than 4096 tokens, leaving room for the model's generated response within the limit of 4096 tokens for both input and output combined. Exceeding this token limit could result in truncation or an incomplete response. Therefore, it is essential to consider the length of both the input prompt and the generated output to ensure it stays within the 4096 token limit for GPT-3.5 and Llama 2.

- The comparison between GPT-3.5 Turbo and Llama 2 (70b) highlights their distinct advantages and limitations. GPT-3.5 Turbo excels in parameter size and historical data availability but requires payment and lacks recent updates. Conversely, Llama 2 (70b) is free, more up to date, yet demands advanced hardware for efficient execution. Their individual characteristics cater to specific needs and usage scenarios.

VI. Conclusion

In conclusion, this study presents a comprehensive comparison of two AI chatbots, one powered by GPT-3.5 Turbo and the other by Llama 2, with additional insights from the GPT Builder version. The project demonstrates the strengths and limitations of each model, highlighting GPT-3.5 Turbo's expansive parameter size and historical data availability, and Llama 2's open-source accessibility and recent data updates. The comparison reveals that while GPT-3.5 Turbo necessitates payment and lacks recent data, Llama 2 demands advanced hardware for effective operation but offers free access.

This analysis underscores the significance of choosing the right chatbot model based on specific application needs and constraints. For future work, it is recommended to explore the integration of these models with more sophisticated user interfaces and the potential of combining their strengths to create a hybrid model that maximizes performance and minimizes limitations. Further research could also investigate the impact of ongoing updates and improvements in AI models on chatbot effectiveness in various domains.

References:

Hugging Face NLP Course: An introductory guide to Natural Language Processing (NLP) using PyTorch. (<https://huggingface.co/learn/nlp-course/chapter1/4?fw=pt>)

Transformer (Machine Learning Model): Overview of the Transformer model in machine learning. ([https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)))

Transformers in AI: Discusses the role of transformers in artificial intelligence. (<https://aws.amazon.com/fr/what-is/transformers-in-artificial-intelligence/>)

IBM - Transformer Model: Insights into the transformer model and its applications. (<https://www.ibm.com/topics/transformer-model>)

Langchain Documentation: Introduction and getting started guide for Langchain. (https://python.langchain.com/docs/get_started/introduction)

CEEOL Article: Scholarly article on AI and NLP technologies. (<https://www.ceeol.com/search/article-detail?id=668455>)

ScienceDirect Article (1): Research paper on advancements in AI and machine learning. (<https://www.sciencedirect.com/science/article/pii/S2666827020300062>)

ScienceDirect Article (2): Study on the latest developments in AI technologies. (<https://www.sciencedirect.com/science/article/pii/S2666920X21000278>)

Langchain Documentation (Main) : Comprehensive documentation on Langchain. (<https://python.langchain.com/docs/>)

Pinecone Documentation: Documentation for Pinecone, a database for vector search. (<https://docs.pinecone.io/docs>)

Hugging Face - Sentence Transformers: Overview of Sentence-Transformers for efficient sentence embeddings. (<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>)

Together.ai Documentation: Documentation on Together.ai platform.

(<https://docs.together.ai/docs>)

OpenAI Platform Documentation: Comprehensive guide on using OpenAI's platform.

(<https://platform.openai.com/docs/>)

Streamlit Documentation: Guide on building web applications for machine learning and data science using Streamlit. (<https://docs.streamlit.io/>)

