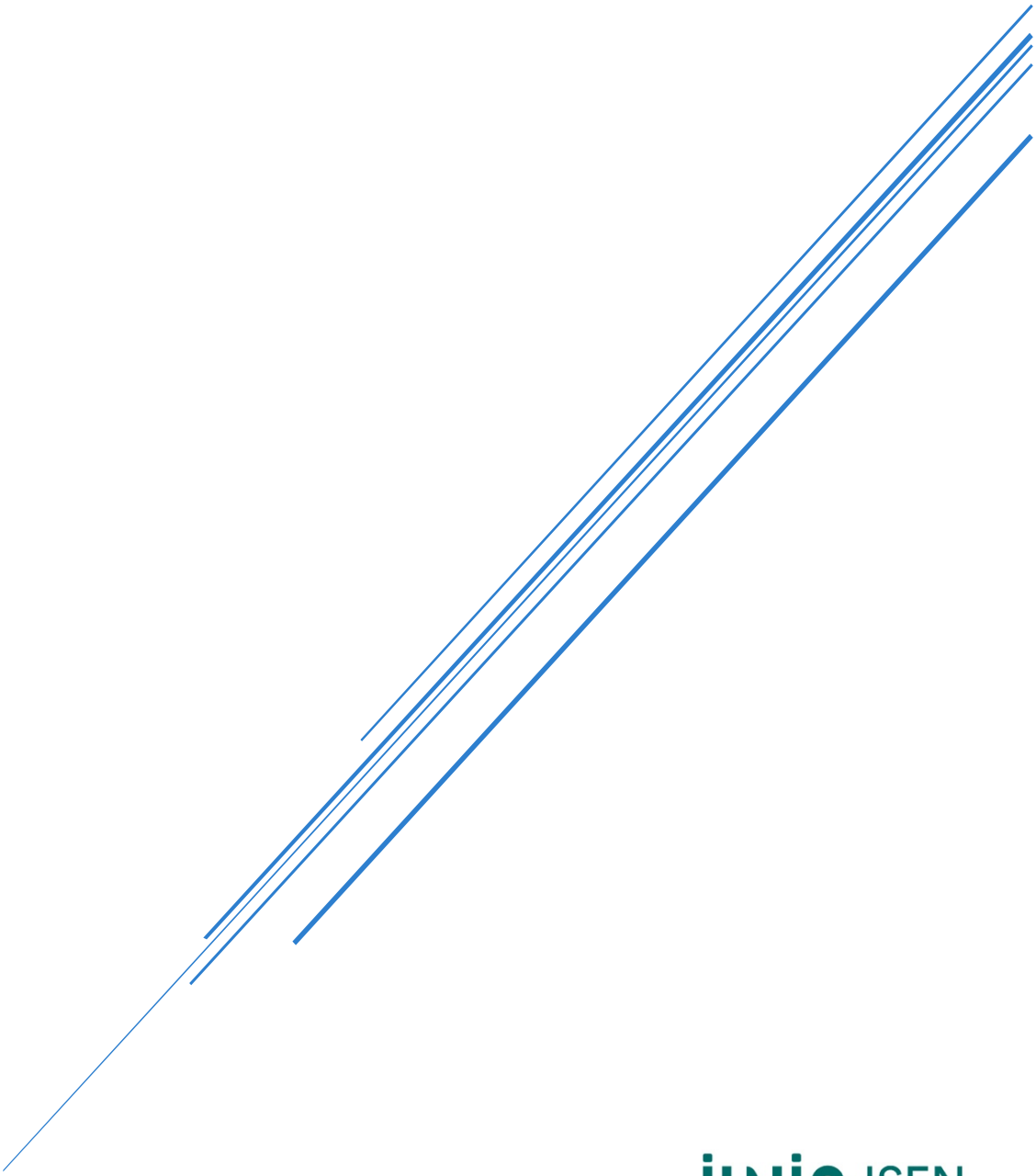
INCLUSIVE LLM

INCLUSION ORIENTED Large Language Models

Gabin CHARLEMAGNE
Romain DUJARDIN
Hugo CORNE
Simon BEUVAIN
Titouan GLANGETAS
Vazha SAGINADZE



JUNIA ISEN
M1 Project 2024-2025

Table of contents

Introduction

Large-scale language models (LLMs) are revolutionizing the way we interact with technology, facilitating complex tasks such as text generation, translation and contextual analysis. However, their universal accessibility remains limited. These models do not always respond optimally to the specific needs of marginalized groups, such as people with disabilities. For example, when a user requests a recommendation for an accessible restaurant, a classic LLM may ignore essential criteria such as the accessibility of the premises, revealing a lack of adaptability.

In this context, making LLMs more inclusive becomes a priority to ensure that their benefits reach a diverse population, regardless of physical or cognitive abilities.

Project Objectives

The Inclusive Multichannel Chatbot project aims to create a virtual assistant, based on an LLM, capable of adapting to the specific needs of people with disabilities, whether visual, hearing, cognitive or motor impairments, with a flexible architecture and a working prototype.

The idea is to build on the limitations of existing chatbots to offer a more inclusive and accessible solution, using the capabilities of advanced language models (LLM).

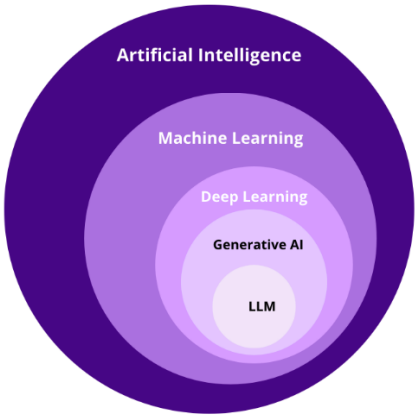
The goal is to develop an innovative prototype that meets the challenges of accessibility and inclusion in today's digital tools.

- Analyze the limitations of current chatbots and their solutions for people with disabilities.
- Adapt an LLM to better meet inclusion needs.
- Propose a technical architecture dedicated to accessible and adapted interfaces
- Design a functional prototype (POC) using existing AI models
- Integrate functionalities adapted to different disabilities (visual, auditory, cognitive, motor)
- Explore tools and technologies to facilitate accessibility
- Test chatbot accessibility and effectiveness with real-life users
- Document the solutions developed to facilitate their deployment and improvement

1. State of the art

1.1 What is an LLM

A Large Language Model (LLM) is an advanced artificial intelligence system designed to process and generate human language. At its core, an LLM is a machine learning model that leverages deep neural networks to analyze text and predict patterns in language. These models are based on transformer architectures and trained on vast text datasets, allowing them to perform tasks like text generation, translation, and contextual understanding. [2][3]



The primary function of a language model is to compute the probability of a word appearing after a given input. For instance, given the sentence fragment *"The sky is ___,"* the model predicts the most probable continuation, such as *"blue."* Through this training process, the model learns to identify patterns in text, resulting in what is known as a **pre-trained language model**.

What distinguishes LLMs from smaller models is their scale. They contain a vast number of parameters—for example, GPT-3 has over 175 billion—trained on immense datasets using **self-supervised learning**. This method enables models to learn directly from unlabeled text, avoiding the costly requirement of manually annotated data. Consequently, the size of an LLM's architecture correlates with its ability to capture complex language patterns and make more accurate predictions.

These models are also adaptable beyond text analysis, including domains like code generation or multimodal tasks involving images and language. Their scale and adaptability have revolutionized fields ranging from customer service to scientific research.

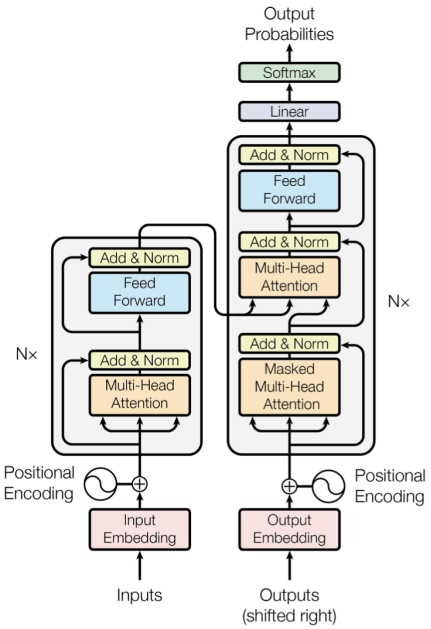
1.2 How LLMs work

LLMs operate using **deep learning**, a subset of machine learning designed to extract patterns from unstructured data. They rely on artificial neural networks composed of multiple layers, including an input layer, hidden layers, and an output layer. Each layer processes information sequentially, refining predictions at each stage. [4]

1.2.1 Architecture

LLMs are typically based on the **Transformer** architecture, which uses self-attention mechanisms to process and generate text. The model consists of multiple layers of neural networks, each containing [1] :

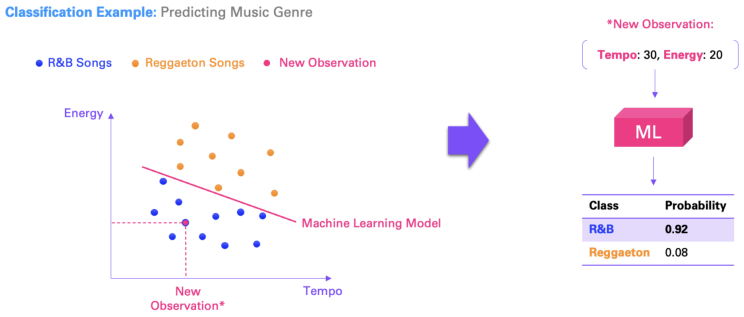
- **Self-attention layers**: These allow the model to weigh the importance of different words in the input when processing each word.
- **Feed-forward layers**: These process the output of the attention layers and introduce non-linearity.



1.2.2 Fundamentals

Machine learning

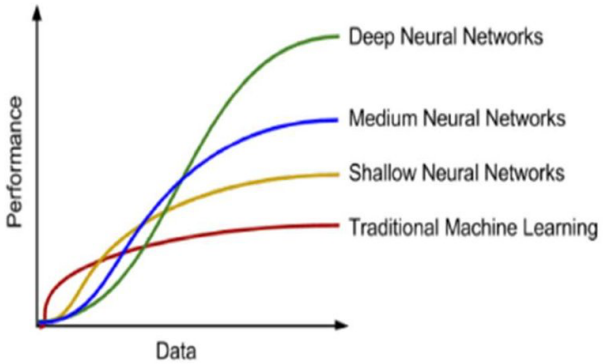
Machine learning aims to discover patterns in data. For example, to distinguish between two genres of music (reggaeton and R&B), a model can be trained to recognize patterns in the tempo and energy metrics of the songs. This model can then predict the genre of a new song based on these patterns.



Deep Learning: Neural Networks

Neural networks are powerful models that can model complex relationships. They consist of layers of "neurons" that process input data to predict an output. Neural networks are particularly effective at processing unstructured for analyze natural language or data such as text and images and generate language like Produce fluid and contextually coherent text.

Neural networks outperform traditional algorithms by effectively modeling unstructured data such as text, where patterns and relationships are less explicit. A recent study showed that a fine-tuned LLM achieved lower error rates compared to conventional machine learning approaches, including feed-forward neural networks, random forests, and linear regression, when predicting electromagnetic spectra.



A neural network in an LLM is made up of interconnected layers as i say in the structure, each playing a specific role in data processing:

The different layers [4][5][8] :

- **Input Layer:** Receives the raw data (in the case of LLMs, text that has been transformed into tokens, digital units corresponding to words or parts of words).
- **Hidden Layers:** These layers perform complex calculations to extract patterns and analyze relationships in text. Modern LLMs (like those based on Transformers) can contain hundreds of layers. These layers include **Self-Attention** Layers: Allow the model to give weight to important words in a sentence. For example, in "The black cat is on the carpet", the word "cat" may be more relevant in predicting the next word than "carpet". **Feed-Forward** Layers: Process data by applying mathematical transformations to improve learning of complex linguistic patterns.
- **Output Layer:** Generates a result in the form of probabilities for each possible word, allowing the model to choose the next word in a sequence.

Neurons: Each layer is made up of neurons (or units). These neurons:

- Receive input in the form of numbers (weights).
- Perform a mathematical operation (such as a weighted sum followed by an activation function).
- Pass the result to the next layer.
-

1.2.3 LLM Training Phases

- Pre-training: LLMs are initially trained on vast amounts of text data from diverse sources. This process involves [8][11] :
    1. **Tokenization**: Breaking down input text into smaller units (tokens).
    2. **Masked Language Modeling**: The model learns to predict missing words in a sentence.
    3. **Next Sentence Prediction**: The model learns to understand the relationship between sentences.
- Fine-tuning of Instructions: The model is fine-tuned to follow specific instructions and respond in a useful way.
- Reinforcement from Human Feedback (RLHF): The model is trained to align its responses with human values and preferences.

Pre-training provides the model with a broad understanding of language. Fine-tuning focuses on specific tasks, while RLHF ensures the model aligns with human preferences and ethical considerations.

1.2.4 Large Language Models: Predicting the Next Word

LLMs are trained to predict the next word in a sequence of words. For example, given a sentence, the model tries to predict the next word. This task is called language modeling. LLMs

use massive neural networks and huge amounts of data to learn to make these predictions accurately [11][13][14].

Although the task of predicting the next word may seem basic, LLMs learn intricate linguistic patterns and even factual knowledge through this process, enabling them to perform complex tasks like summarization or question answering.

Here is a simplified view of how it works step by step:

- **Input**: The text is converted into digital tokens. For example, the sentence "The sky is blue" could be encoded as [12, 85, 24, 67].
- **Data propagation in layers**: Tokens pass through hidden layers where each neuron calculates and transmits a value. Self-attention layers calculate which parts of the text are important. For example, to predict the next word, the model may give more weight to the word "sky."
- **Word prediction**: After several calculation steps, the network generates a probability distribution for each word in the vocabulary. For example: "blue": 90% "gray": 5% "green": 3% etc. The word with the highest probability ("blue") is selected.
- **Repetition to generate sentences**: Once a word is predicted, it is added to the sequence, and the process begins again to generate the next word, until a stopping condition is reached (e.g., a full stop).

1.2.5 Inference

When given a prompt or query, the LLM [8] :

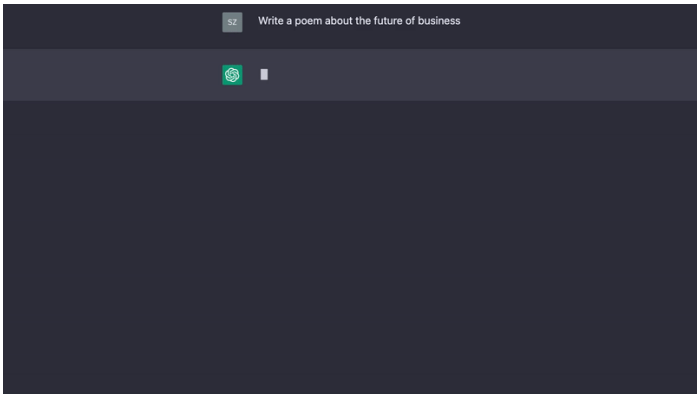
1. Tokenizes the input.
2. Processes it through its layers, attending to relevant parts of the input.
3. Generates a probability distribution over its vocabulary for the next token.
4. Selects the most likely token and adds it to the output.
5. Repeats steps 2-4 until a stop condition is met (special character or a probability threshold)

1.3 Applications of LLMs

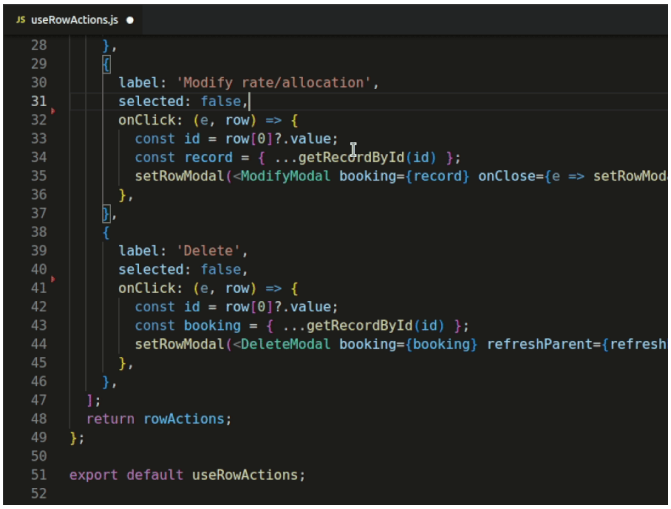
LLMs are transforming various domains by providing innovative solutions [6][9][10] :

- **Chatbots and customer support**: LLMs enable virtual assistants to interact in natural language, helping e-commerce platforms and service providers address customer concerns efficiently.

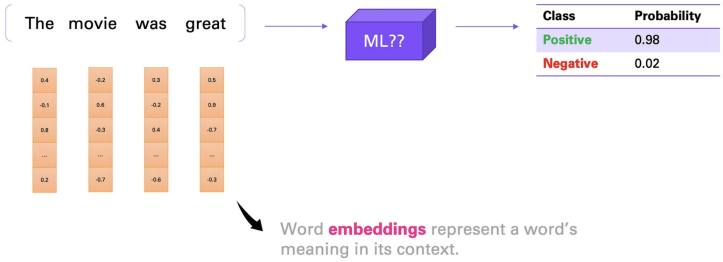
- **Writing and content generation**: Tools like ChatGPT can draft articles, create marketing campaigns, or generate creative content like poetry and screenplays.



- **Assisted programming**: LLMs, such as GitHub Copilot, can assist developers by suggesting code snippets or debugging scripts, significantly speeding up software development.



- **Complex data analysis**: In research-heavy fields like genomics, LLMs analyze large datasets to uncover patterns or correlations. Similarly, they perform sentiment analysis on customer reviews, aiding brands in understanding consumer behavior.



Notable examples of LLMs include **ChatGPT (OpenAI)**, **Bard (Google)**, **Llama (Meta)**, and **Copilot (GitHub)**, each tailored for specific use cases and other cases of use like **Medical** with Assisted diagnosis or clinical documentation assistance. **Education** with Personalized educational tools for students. **Finance** with Market analysis using textual data processing.

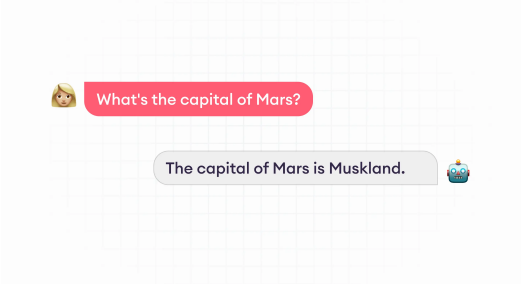
1.4 Advantages and limitations

So, we can list some advantages of these LLMs Advantages [7][10] :

- **Flexibility:** LLMs can handle a wide range of tasks, from simple queries to complex problem-solving, adapting to varied and unpredictable requests.
- **Contextual learning:** By leveraging vast datasets, LLMs excel at understanding concepts in context and identifying nuanced relationships in language, improving accuracy and relevance.
- **Wide range of applications:** LLMs are versatile tools applied in fields as diverse as education, content generation, programming assistance, and technical data analysis.
- **Pre-training Advantage:** Pre-training on massive datasets enables LLMs to generalize across multiple tasks, requiring less task-specific data during fine-tuning.

However, LLMs have several limitations, here are some of them:

- **Dependence on training data:** LLMs may reproduce biases or generate false information ("hallucinations") if the training data contains inaccuracies or biases. **Mitigation**: Providing reliable and relevant context—such as embedding a specific Wikipedia article—can help the model deliver accurate responses to targeted questions. [12]



- **Resource intensity and complexity**: Training and deploying LLMs demand significant computational power, energy, and technical expertise. This can limit accessibility for smaller organizations or individuals.
- **Bias**: Models may reflect biases present in their training data.
- **Lack of True Understanding**: Despite their impressive outputs, LLMs don't truly "understand" in the way humans do.
- **Security and Privacy:** Without proper safeguards, user input data could be misused or inadvertently expose sensitive information, raising concerns about privacy and data protection.

1.5 Limits of current systems

There are several systems, platforms and tools on the market today to meet the needs of people with disabilities. Here is a summary of the various existing solutions and their limitations.

1.5.1 Databases and Existing Tools

Solution Name	Description	Strengths	Limitations
Accesslibre **[18]**	French public platform providing data on accessibility (ramps, toilets, Braille menus).	Clear and structured data Government-backed platform	Static information, lacks interaction Outdated data
[18]	Collaborative map assessing wheelchair-accessible public places.	Community-driven Easy overview	Focused on reduced mobility Limited service integration
AccessNow	Interactive accessibility map of public/private spaces where users can call on the community to obtain reports.	Interactive and user-friendly Crowdsource-driven data	Less suited for specific environments Limited verification
Handicap.fr	French information portal on rights, services and places accessible to people with disabilities. It lists accessible establishments and local initiatives.	Comprehensive resource hub Covers many topics	Primarily human-contributed data Less detailed than tools

1.5.2 Specific Solutions for Accessibility

Application	Description	Strengths	Limitations
Ava	Mobile app for real-time audio transcription, designed for people who are deaf or hard of hearing.	Real-time transcription Easy group conversation support	Covers only auditory impairments Lacks multichannel interaction mechanisms
Be My Eyes	Connects visually impaired people with volunteers or agents via video for human assistance in navigation and tasks.	Human interaction for tailored support Effective for tasks	Requires human availability No AI-driven autonomous interactions
Seeing AI	App leveraging computer vision for visually impaired users (text reading, object description).	Effective for visual tasks Supports text/object recognition	Functionality limited to simple tasks Requires network connection
Transkriptor	Automated tool for transcribing meetings or verbal interactions.	Useful for documenting content Automated and timesaving	Limited to transcription No interaction support

1.5.3 Existing Solutions Using LLMs or Related Technologies for Accessibility

Solution Name	Description	Strengths	Limitations
Microsoft Azure AI and Bot Framework	Platform for creating chatbots combining AI and cloud integrations. Includes Cognitive Services (speech-to-text, text-to-speech, translation).	Real-time voice conversion Integration with multiple services Adaptive Cards for interface flexibility	Lack of specialization for complex disabilities Costly fine-tuning for specific needs Limited support for non-standard environments
Google Dialogflow	Multilingual and multichannel chatbot (voice, text) based on NLP models integrated with Google Cloud. Compatible with major communication platforms.	Multilingual capabilities Strong integration with Google tools Good for managing user intents	Lack of tools for specific cognitive or motor impairments Requires external layers for Braille or adapted speech synthesis Limited customization without technical expertise
ChatGPT/OpenAI	Versatile and general-purpose language model capable of answering a wide range of questions, adaptable via APIs or fine-tuning.	Versatile and general-purpose Wide range of applications API available for customization	Not specialized in accessibility aspects Challenges in tailoring responses to emotional/contextual signals Limited tactile navigation support
IBM Watson Assistant	Cognitive solution for designing complex virtual assistants with NLP integration. Includes multichannel support and third-party tool integration.	Multichannel experience support NLP-based customization Integration with third-party tools	Less intuitive for specific accessibility cases Lacks native Braille or gesture recognition integrations Complex configuration process

The "Inclusive Chatbot" project stands out by combining conversational AI with accessibility data sources to address various disabilities. Unlike static databases such as Accesslibre or AccessNow, this chatbot offers dynamic and personalized interactions, ensuring proactive assistance tailored to the user's needs. For example, it can provide customized suggestions for accessible restaurants or offer step-by-step navigation adapted to both visual and auditory impairments. Additionally, it simplifies complex information for users with cognitive disabilities, ensuring inclusivity across all interfaces.

This project bridges gaps in existing solutions by integrating real-time enriched data and providing multichannel support, including voice, text, and tactile feedback. It also introduces innovative features like emotional and contextual recognition to adapt interactions dynamically. Partnerships with platforms such as Acceslibre and Tobii Dynavox could further enhance its utility, creating a comprehensive tool capable of supporting diverse user profiles. Moreover, the chatbot's modularity allows for integration with emerging technologies, such as augmented reality, to extend its capabilities in the future.

1.6 Adaptation of LLMs to proprietary data

1.6.1 Latent Representations in AI

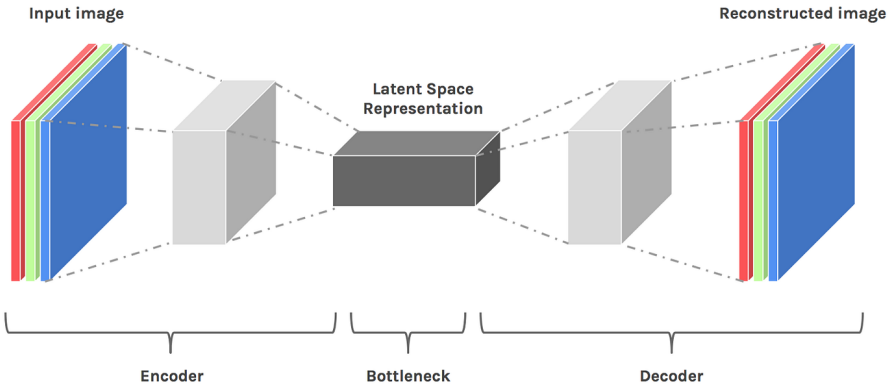
Latent representations refer to the hidden features or patterns that machine learning models extract and encode from data. These representations allow AI systems to abstract complex, high-dimensional data (like images, text, or audio) into simpler, meaningful structures within a vector space. They are pivotal in many AI applications, such as clustering, recommendation systems, and natural language processing.

Key Aspects of Latent Representations:

Dimensionality reduction is a key aspect of latent representations, where techniques like Principal Component Analysis (PCA) or autoencoders reduce the complexity of data while retaining critical information. Semantic understanding is another crucial feature, particularly in natural language processing, where models like BERT or GPT encode the semantic meaning of words or phrases into latent vectors. Latent representations also enable transfer learning, allowing their reuse across different tasks and reducing computational effort for new problems. By representing data as vectors, latent representations facilitate clustering and similarity identification using metrics like cosine similarity or Euclidean distance.

In text understanding, embeddings such as Word2Vec or Sentence Transformers represent words, sentences, or documents in a meaningful vector space. Image recognition relies on convolutional neural networks (CNNs), which generate feature maps as latent representations to identify objects in images. Recommender systems utilize latent factors in collaborative filtering to match users with content based on preferences.

These latent representations not only enhance AI's efficiency but also bridge the gap between complex data and actionable insights.



1.6.2 Vector Databases and Their Role in AI

Vector databases store and manage high-dimensional vector data, enabling efficient similarity search and retrieval. They are critical in applications involving latent representations, especially when integrating retrieval-augmented methods like RAG or HyDE.

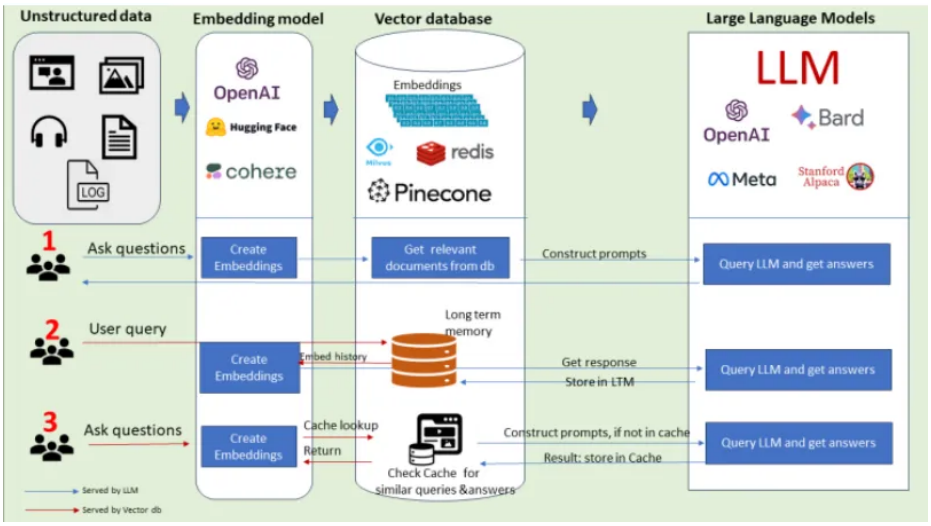
Key Features of Vector Databases:

One of the primary advantages of vector databases is their ability to conduct efficient similarity searches. Algorithms such as Approximate Nearest Neighbor (ANN) enable rapid retrieval of similar vectors, even within large datasets. Additionally, these databases are designed for scalability, handling millions or billions of vectors without significant performance trade-offs.

Indexing techniques play a crucial role in the efficiency of vector databases. Methods such as Facebook AI Similarity Search (FAISS), Hierarchical Navigable Small World graphs (HNSW), and Inverted File Index (IVF) optimize query processing. Furthermore, these databases integrate seamlessly with AI pipelines, connecting with frameworks like LangChain to enhance retrieval-augmented applications.

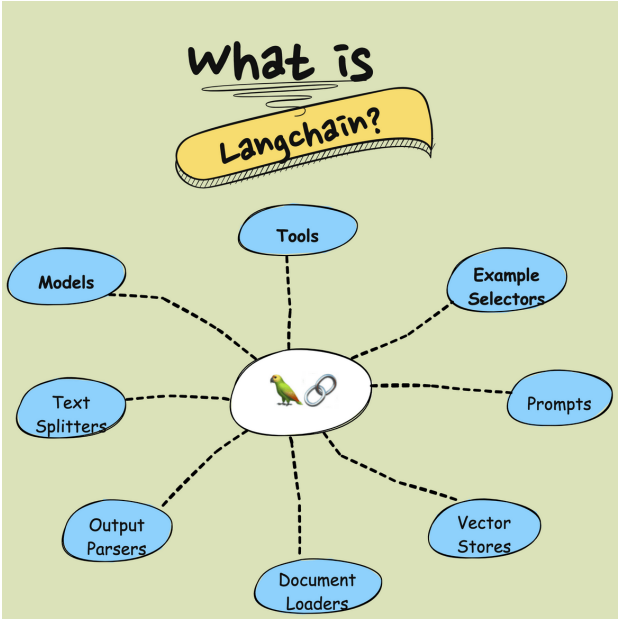
Applications of vector databases are diverse. Recommendation engines benefit by matching users with content or products through vector comparisons. Search systems leverage these databases for semantic search, retrieving results based on contextual similarity rather than simple keyword matching. Finally, RAG pipelines use vector databases to store embeddings of proprietary documents, providing precise and context-aware responses.

By combining high-dimensional data storage with advanced retrieval capabilities, vector databases form an essential component of modern AI infrastructure.



1.6.3 LangChain

LangChain is a versatile library designed to simplify the development of applications based on language models. By seamlessly integrating various steps such as text generation, information retrieval, and tool or database interaction, LangChain provides developers with a framework to build sophisticated pipelines tailored to specific use cases. The library's modular design allows for the flexible combination of LLM capabilities with external resources, enhancing both efficiency and adaptability. [19][25]



Key Features of LangChain :

1. Chain Building: Enables the creation of complex workflows where the output of one model step can serve as input for another. This chaining mechanism is ideal for multi-step reasoning or task-specific applications.
2. Integration with External Tools: Supports connections to APIs, databases, and search systems to enrich the language model's outputs with external, real-time data.
3. Memory Management: Provides memory modules to maintain context over extended interactions, ensuring coherent and contextually aware responses.
4. Customizability: Offers developers the ability to fine-tune chains, integrate custom modules, and adapt the framework to proprietary requirements.

Applications of LangChain in Proprietary Data Adaptation:

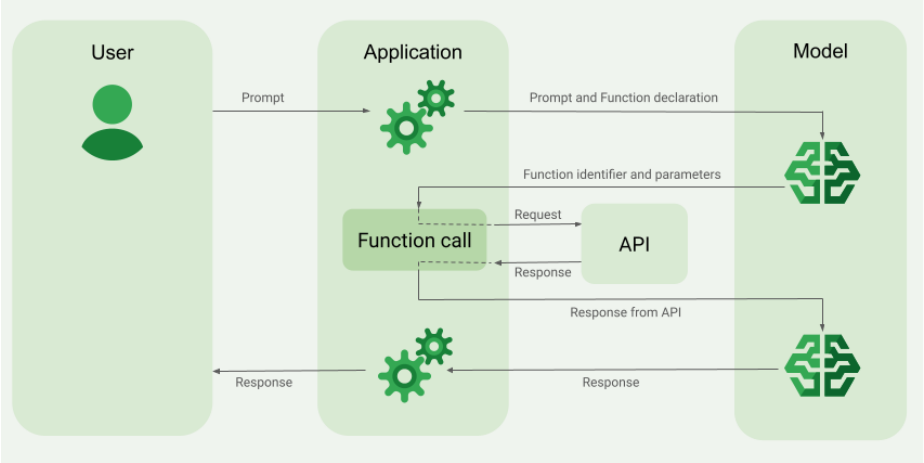
- Dynamic Knowledge Bases: Creating chatbots that interact with live databases to fetch up-to-date and domain-specific information.
- Content Generation Pipelines: Automating processes such as drafting, editing, and summarization of proprietary documents.
- Interactive Assistants: Enhancing user interactions by leveraging tools like calculators, calendars, or proprietary APIs.

Challenges :

- Complexity in Design: Building efficient chains requires careful planning to avoid unnecessary computational overhead.
- Tool Dependency: Reliance on external tools or databases may introduce latency or availability issues.

1.6.4 Prompt Chaining and Function Calling in LangChain

Prompt chaining and **function calling** are transformative techniques that enhance the capabilities of language models, particularly when integrated into modular frameworks like LangChain. **Prompt chaining** involves breaking down complex tasks into sequential steps, where the output of one model prompt becomes the input for the next. This structured approach allows for efficient multi-step reasoning and problem-solving. On the other hand, **function calling** enables language models to interact with predefined functions, executing specific tasks such as querying APIs, performing calculations, or retrieving contextual information dynamically. [20][22][23]

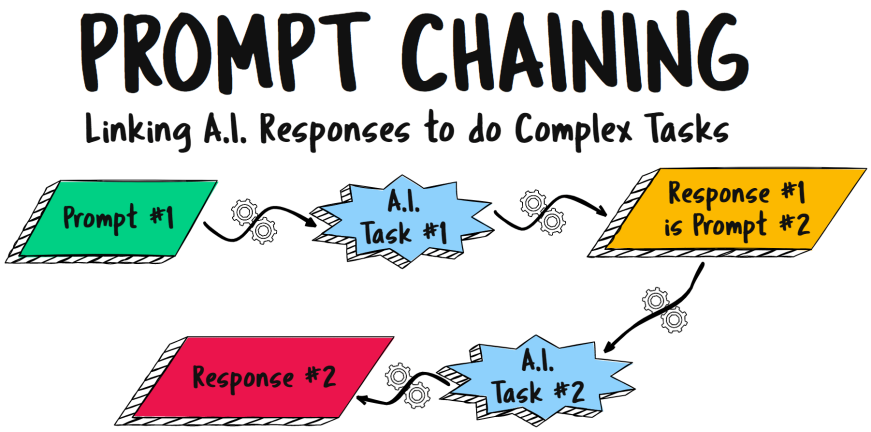


LangChain leverages these techniques to build robust AI pipelines. Through **function calling**, for instance, a conversational assistant can dynamically fetch real-time data from external APIs or databases to enrich its responses. When combined with **prompt chaining**, LangChain facilitates workflows where language models alternate between textual reasoning and task execution, all while maintaining contextual awareness.

Applications :

- **Intelligent Assistants**: Constructing workflows that seamlessly integrate information retrieval and analysis to answer complex queries.
- **Automation**: Enabling models to autonomously execute tasks such as booking appointments or analyzing documents through external function calls.
- **Enhanced Retrieval and Generation**: Merging prompt chaining with retrieval-augmented generation (RAG) techniques to produce contextually accurate and enriched responses.

By integrating these methods, LangChain provides developers with tools to create dynamic, efficient, and highly adaptable AI systems, expanding the potential of language models beyond static text generation.



1.6.5 Langraph: Simplifying AI Workflow Visualization and Debugging

Langraph is an innovative tool designed to simplify the creation, visualization, and debugging of complex workflows involving language models. Built to work seamlessly with frameworks like LangChain, Langraph provides a user-friendly interface for mapping out intricate AI pipelines, making it easier for developers to design and troubleshoot multi-step processes.

Langraph's core strength lies in its ability to translate programmatic workflows into clear, interactive graphs. These visualizations not only improve understanding but also enable real-time monitoring and optimization of tasks such as prompt chaining, function calling, and data retrieval. [24]

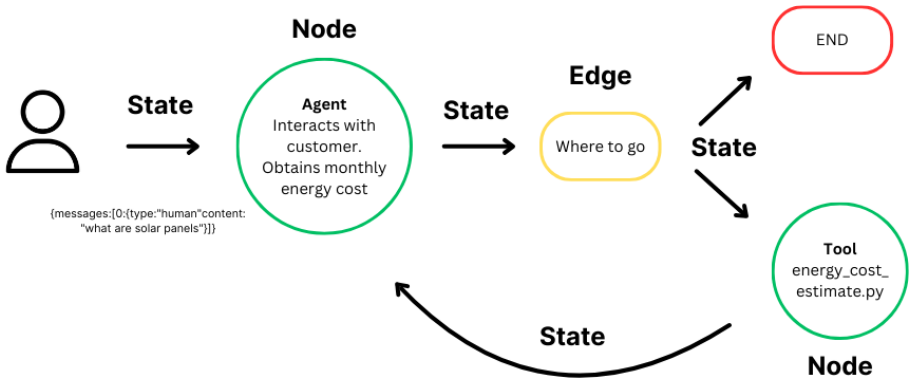
Key Features:

- **Workflow Visualization**: Automatically generates interactive graphs to display the sequence of prompts, function calls, and decision branches in a pipeline.
- **Debugging Tools**: Highlights errors or bottlenecks in the workflow, enabling quick iteration and optimization.
- **Seamless Integration**: Compatible with LangChain and other frameworks, allowing for smooth import/export of workflows.
- **Collaboration-Friendly**: Facilitates teamwork by providing a shared visual context for understanding and improving AI pipelines.

Applications:

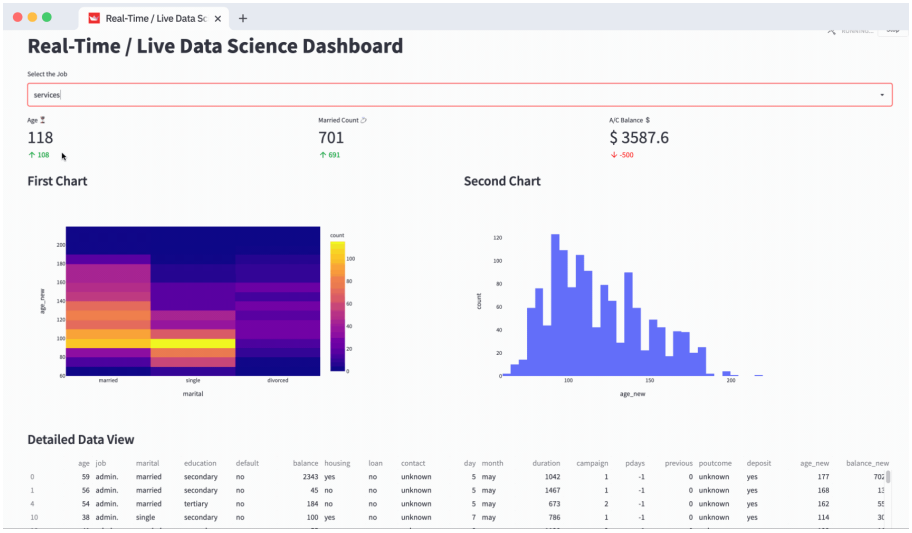
- **Complex AI Systems**: Designing workflows with multiple steps, such as combining retrieval-augmented generation (RAG) with external API calls, becomes more intuitive.
- **Interactive Prototyping**: Rapidly testing and modifying workflows with a graphical interface.
- **Educational Tools**: Providing a visual aid for teaching prompt chaining, function calling, and other advanced concepts in language model-based systems.

Langraph democratizes access to advanced AI workflows by reducing the technical barriers associated with coding-only approaches. Its ability to visually represent and debug pipelines enhances productivity, particularly for teams working on collaborative or large-scale AI projects.



1.6.6 StreamLit

StreamLit is a Python framework that enables the rapid creation of interactive web applications. Its simplicity and efficiency make it a preferred choice for prototyping AI and machine learning tools, as well as for developing dashboards and data visualization interfaces with minimal code. StreamLit's user-friendly design allows developers to focus on functionality rather than the intricacies of web development.



Key Features of StreamLit:

1. **Ease of Use:** Requires only a few lines of Python code to build fully functional web applications.
2. **Interactive Widgets:** Includes built-in tools like sliders, buttons, and file uploaders to create dynamic user interfaces effortlessly.
3. **Seamless Integration:** Works well with popular Python libraries such as Pandas, NumPy, and Matplotlib for data manipulation and visualization.
4. **Rapid Prototyping:** Facilitates quick development cycles, making it ideal for testing and showcasing AI or ML models.

Applications of StreamLit in Proprietary Data Adaptation:

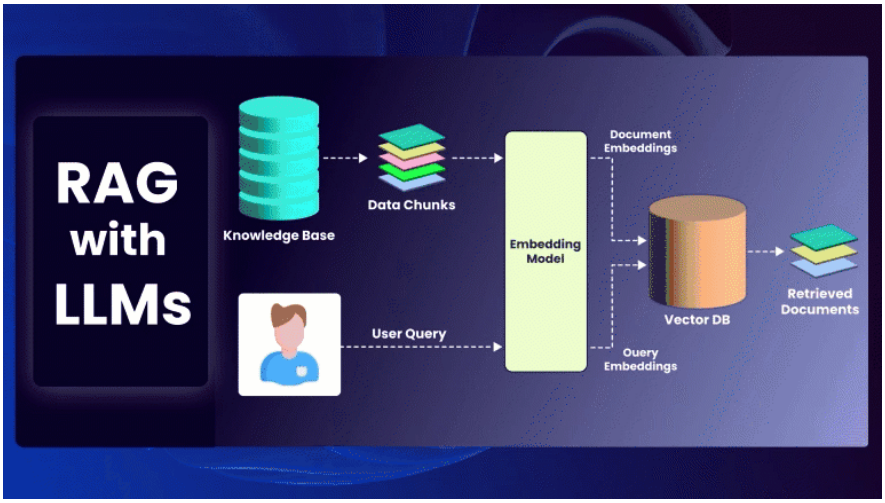
- **AI Model Deployment:** Simplifying the presentation and testing of proprietary models through intuitive interfaces.
- **Data Analysis Dashboards:** Providing real-time insights into proprietary data with interactive visualizations.
- **Prototyping Internal Tools:** Quickly building and iterating on tools tailored to organizational workflows.

Challenges :

- **Scalability:** StreamLit is primarily designed for prototyping and may require additional frameworks for large-scale applications.
- **Customization Limits:** While powerful, its simplicity can limit extensive customization compared to traditional web development frameworks.

1.6.7 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) represents a pivotal technique that bridges the gap between the general knowledge base of large language models (LLMs) and domain-specific requirements. RAG combines a language model with an external retrieval system, such as a search engine or a vector database, to enhance response accuracy and relevance by injecting factual, contextually relevant information during inference. This approach addresses one of the main limitations of LLMs: their reliance on fixed pretraining data, which may be outdated or irrelevant to specialized use cases.



Key Components of RAG:

1. **Retriever:** This module searches and retrieves relevant documents or data from a predefined corpus or external sources. State-of-the-art retrievers often use dense

embeddings and similarity search techniques (e.g., FAISS, Milvus) to efficiently locate the most pertinent information.

2. **Generator:** The language model generates responses by combining its inherent linguistic capabilities with the retrieved context, effectively augmenting its knowledge with up-to-date or domain-specific information.
3. **Feedback Loop:** RAG workflows may include mechanisms for iterative improvement, where retrieved data can refine the generated outputs or guide further searches.

Applications of RAG in Proprietary Data Adaptation:

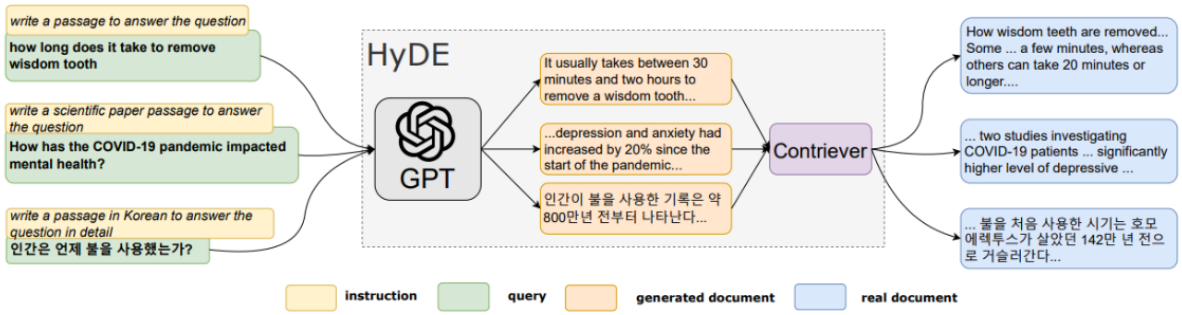
- **Customer Support Chatbots:** Leveraging proprietary FAQs or support documents to provide precise and customized responses.
- **Knowledge Base Expansion:** Enhancing internal knowledge management systems with AI-driven search and response generation.
- **Regulatory and Legal Compliance:** Ensuring AI systems adhere to organizational policies by grounding outputs in company-approved documents.

Challenges:

- **Data Privacy:** Proprietary data integration demands robust security and compliance measures to prevent unauthorized access.
- **Domain-Specific Optimization:** Designing retrievers and generators to handle niche terminologies and formats effectively.
- **Latency:** Real-time applications require optimization of retrieval and generation to minimize delays.

1.6.8 HyDE (Hypothetical Document Embeddings)

Hypothetical Document Embeddings (HyDE) provides an innovative alternative to RAG by introducing a generative step before retrieval. Instead of directly searching for relevant data based on a query, HyDE first generates a "hypothetical document" that represents the most plausible response to the query. This document serves as a reference point to guide a secondary search in a database or corpus, ensuring that the retrieved information aligns closely with the query's intent.



Key Components of HyDE :

1. **Hypothetical Document Generator:** Using a language model, this component generates a synthetic document that hypothesizes a likely answer to the query. This document encapsulates the essential context and intent of the query, serving as a bridge between generation and retrieval.
2. **Retriever:** The generated document is used as a reference to perform a focused search within a predefined corpus or external knowledge base. This process retrieves data that closely matches the hypothetical context.
3. **Refiner:** The final step involves synthesizing the retrieved information with the hypothetical document to create a coherent and accurate response.

Applications of HyDE in Proprietary Data Adaptation:

- **Research Assistance:** Generating preliminary drafts of responses to queries and refining them with domain-specific data.
- **Data-Driven Decision Making:** Assisting in scenarios where partial information exists by hypothesizing and validating potential outcomes.
- **Enhanced Chatbot Capabilities:** Delivering highly contextualized responses in proprietary environments by leveraging both generative and retrieval-based strengths.

Challenges:

- **Synthetic Document Quality:** The effectiveness of HyDE depends heavily on the quality and relevance of the hypothetical document.
- **Retrieval Alignment:** Ensuring that the retrieval process accurately matches the context of the synthetic document.
- **Computational Overhead:** The dual-step process of generating and retrieving can introduce latency.

1.7 Data sources for inclusion and accessibility

1.7.1 Definition of terms

Accessibility primarily concerns devices, infrastructures, and technological interfaces that must be usable by everyone, including people with disabilities or temporary limitations. Inclusion, on the other hand, aims to ensure that all populations, particularly minorities or marginalized groups, are considered in societal decisions and projects. [17]

1.7.2 The different categories of disabilities:

Motor disability: P Esquivel, K Gill, M Goldberg, S. A Sundaram, L Morris and D Ding. Voice Assistant Utilization among the Disability Community for Independent Living: A Rapid Review of Recent Evidence

This type of disability affects a person's mobility. It may involve a total or partial loss of motor function in the limbs, difficulty moving around, or challenges performing tasks requiring physical coordination
Examples: Paralysis, amputation, neuromuscular disorders, multiple sclerosis.

Visual disability:

This refers to an impairment affecting vision, ranging from partial visual loss to total blindness. It can limit access to written information, visual interfaces, or spatial orientation. Examples: Glaucoma, cataracts, congenital blindness.

Hearing disability:

This type of disability involves a partial or total loss of hearing, leading to difficulties in communication, following conversations, or perceiving important sound signals. Examples: Deafness, hearing loss.

Cognitive disability:

This category includes disorders that affect higher mental functions such as memory, concentration, comprehension, or problem-solving.
Examples: Dyslexia, Attention Deficit Hyperactivity Disorder (ADHD), intellectual disability.

Psychiatric disability:

This disability is related to psychiatric or mental disorders that significantly impact daily life. It can involve difficulties in managing emotions, behavior, or social relationships. Examples: Schizophrenia, bipolar disorder, severe anxiety, major depression.

Sensory disability:

In addition to auditory and visual disabilities, this category includes impairments affecting other senses, such as smell, taste, or touch, which may limit interaction with the

environment.
Example: Hypoesthesia (reduced tactile sensitivity).

Disability related to disabling diseases:

Certain chronic or progressive illnesses may be considered disabilities due to their impact on autonomy and quality of life.
Examples: Diabetes, cancer, heart failure, Parkinson's disease. **Error! Reference source not found.**

1.7.3 How could we adapt our AI to make it as inclusive and accessible as possible for these disabilities?

Motor disabilities :

- Voice interfaces: Allow users to control the application or AI through voice commands, reducing the need for physical devices.
- Compatibility with assistive devices: Ensure the AI works with technologies like joysticks, eye trackers, or adaptive switches.

Visual disabilities :

- Speech synthesis: Use text-to-speech technologies to make textual information accessible.
- Screen reader compatibility**:** Ensure the application is fully compatible with existing tools like NVDA, VoiceOver, or JAWS.

Cognitive disabilities or learning disorders:

- Simplified interfaces: Reduce interface complexity to enhance comprehension and usability.
- Contextual assistance: Provide interactive guides or tailored suggestions based on specific needs.
- Content adaptation**:** Offer explanations suited to the user's level of understanding, such as using images or concrete examples.
- Dyslexia: Use accessible fonts, colors and integrate existing APIs to enable a dyslexic-friendly mode.

Speech disabilities :

- Recognition of non-verbal expressions**:** Allow communication through gestures or facial expressions.
  Error! Reference source not found.

1.7.4 Dataset and sources for inclusion

Dataset Name	Dataset Link	Established Area	API link	Accessibility Description	File type
Acceslibre: All information on the accessibility of public places in France	https://acceslibre.beta.gouv.fr/	Entire France 525 000 location	https://api.gouv.fr/les-api/api-acces-libre	Detailed information on each infrastructure's accessibility (e.g., presence of ramps, steps, etc.)	csv excel
Europa: List of Tourism and Handicap Certified Establishments	https://data.europa.eu/data/datasets/https-data-economie-gouv-fr-explore-dataset-liste-des-etablissements-labellises-tourisme-et-handicap-?locale=fr	Entire France		Classification of accessibility by type of disability: [visual], [motor], [auditory], [mental]	csv json excel
Accommodation Accessibility in Île-de-France - Paris je t'aime	https://opendata.paris.fr/explore/dataset/accessibilite-des-hebergements-en-ile-de-france-paris-je-t-aime/export/	Ile de France 536 accommodation locations	https://opendata.paris.fr/explore/dataset/accessibilite-des-hebergements-en-ile-de-france-paris-je-t-aime/api/		csv json excel

Acceslibre : https://acceslibre.beta.gouv.fr/

This dataset could be very useful for us as it lists 525,000 locations in France, describing the accessibility for each type of disability.

Additionally, a free API allows us to query the platform's database: https://api.gouv.fr/les-api/api-acces-libre

API documentation:https://www.data.gouv.fr/fr/dataservices/api-acces-libre/

To maximize our sources and references, it would be ideal to combine multiple datasets. Below is a link to the European Union's site that lists several datasets related to accessible locations for people with disabilities across France:

Data Europa: https://data.europa.eu/data/datasets/https-data-economie-gouv-fr-explore-dataset-liste-des-etablissements-labellises-tourisme-et-handicap-?locale=fr

The site lists all freely accessible datasets from the EU, which could be very helpful if we need additional datasets for the project

2. Discussion

2.1 Current solutions for multichannel interfaces

	Description	Why is it suitable?	Potential obstacles
GenIA chatbots (chatGPT, Gemini, ...)	Advanced AI chatbots to generate coherent text, answer questions and assist with tasks.	Solid foundation for a multi-channel chatbot with multimodal processing (text, image, audio).	Requires a good connection
Seeing AI, Google Lookout, Be my IA	Mobile applications using camera vision to describe objects, text and faces, helping the visually impaired.	Visual recognition and audio description for improved accessibility.	Internet-dependent, variable accuracy depending on lighting and scene complexity.
Alexa/siri/google assistants	Voice recognition-based virtual assistants for commands, responses and control of connected devices.	Audio playback of responses, home automation and connected device compatibility.	Limitations for complex or specific orders.
IRIS by IVèS	3D assistant designed for the deaf to communicate in sign language.	Interactive 3D avatar, adaptable to various media (screens, tablets, web).	Advanced infrastructure for sign recognition, dependent on camera performance.
Proloquo2Go, Livox	AAC applications for people with communication disorders.	Intuitive interface with pictograms, photos and visual customization.	Limited availability of pictograms to express complex or abstract concepts.
Voiceitt	Speech recognition application designed to help people with speech impairments.	AI that learns voice patterns and is compatible with assistants like Alexa.	Requires prior recording of 200 phrases for effective recognition. Requires a good connection

In summary, analysis of current multichannel solutions reveals several elements to be considered for the realization of our project:

The integration of text, image and audio, as in advanced AI chatbots, is essential for an accessible multimodal service.

The need for a reliable connection encourages us to consider offline functionalities.

Interface customization, like that of Proloquo2Go, is crucial to meeting the specific needs of users with disabilities.

The integration of specific technologies, such as sign recognition or adapted text-to-speech, can significantly improve accessibility. [17]

2.1.2 Help windows

Implementation of a small window in the UI to customize the interface display according to the user's difficulty [16] :

- Select disability to automatically customize the interface according to needs
- Language
- Cursor size
- Light/dark mode
- Page colors, contrast modification, for specific visual impairments.
- Font size for the visually impaired.
- Special fonts
- Disable animations, for users with cognitive impairments.
- Spacing and line spacing
- Automatic content playback (audio accessibility)
- Page zoom (magnifier)
- Disable sounds
- Highlighting options
- Enable “simplified” mode
- Visual feedback for clicks and interactions
- Save user settings

(These options are examples, see if they are useful/feasible)

2.1.3 Voice assistance

As soon as the chatbot is used for the first time, a voice asks if the user needs voice assistance, so that he or she is not restricted to searching the interface for this feature.

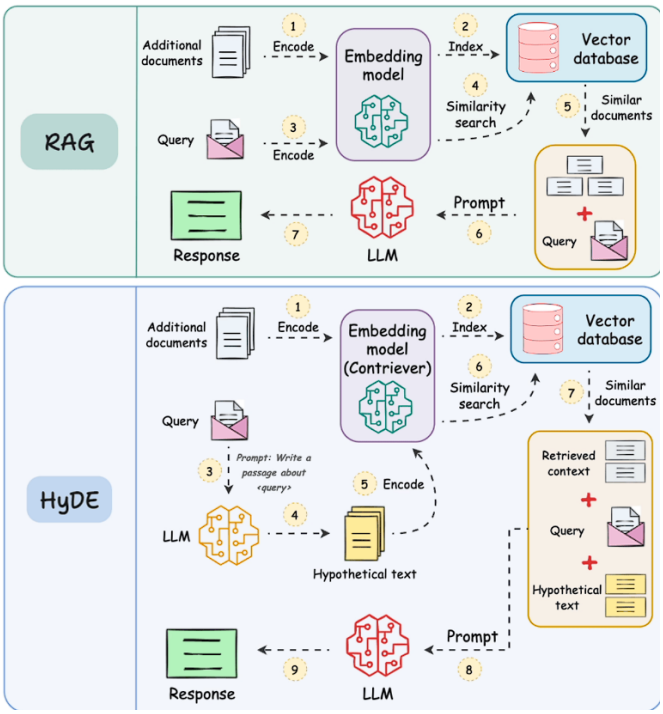
Example of use: allow users to configure the interface by saying voice commands *(“Increase font size”, “Activate dark mode”)*. [18]

2.2 Traditional RAG vs. HyDE: An Innovative Approach to Retrieval

One critical limitation of traditional Retrieval-Augmented Generation (RAG) systems lies in their inability to handle semantic dissimilarities between queries and relevant answers. For instance, when searching for information related to "What is ML?", traditional RAG might incorrectly prioritize results semantically closer to "What is AI?" rather than a more relevant explanation like "Machine learning is fun." This results in the retrieval of less relevant contexts during the search process.

HyDE (Hypothetical Document Embeddings) addresses this challenge by generating a hypothetical document (H) that captures the essence of the query. Using a retriever model, this hypothetical answer is embedded into a vector space and serves as a reference for querying a vector database. This process retrieves relevant contexts that align more closely with the generated document than with the original query. The retrieved context, hypothetical document, and query are then processed by the language model to produce a coherent and accurate final response.

By leveraging contrastive learning techniques, HyDE ensures that the embeddings generated are more representative of real documents, significantly improving retrieval performance. However, this comes at the cost of increased computational overhead and reliance on large language models. This innovative approach highlights a tradeoff between retrieval precision and system latency, making it a valuable tool in specific, high-accuracy applications.



Our Approach for the Multichannel Chatbot

For our multichannel chatbot project, which aims to adapt to users with disabilities, we propose focusing on creating a robust Retrieval-Augmented Generation (RAG) system as the foundation, since implementing Hypothetical Document Embeddings (HyDE) effectively will ultimately require a solid RAG framework. RAG will handle straightforward queries by retrieving precise information from structured databases, ensuring efficiency and reliability. Once this foundation is strong, HyDE can be integrated to tackle more complex or ambiguous queries, generating hypothetical answers to better align with the user's intent.

This step-by-step approach ensures that we fully leverage the strengths of both systems. RAG provides the necessary groundwork for speed and accuracy, while HyDE enhances semantic understanding and context adaptability—critical for users with cognitive or communication disabilities. By first perfecting RAG and then layering in HyDE with tools like LangChain, we can develop a chatbot that is truly inclusive, adaptable, and performant. [15]

2.3 Overview of selected models

To meet the objectives of the project, we prioritize models that ensure long-term sustainability and optimal performance.

- **Sustainable solution** over time, therefore, to have **free** use over time
- The **most efficient** model possible

Model	Provider	Context Window	Open-Source	Price / Million	Price	Quality	Speed
GPT-4o	OpenAI	128k	No	7.5	★☆☆	★★★	★★☆
GPT-4 Turbo	OpenAI	128k	No	15	★☆☆	★★★	★☆☆
GPT-4	OpenAI	8k	No	37.5	★☆☆	★★★	★★☆
GPT-3.5 Turbo	OpenAI	16k	No	0.75	★★☆	★☆☆	★★☆
Gemini 1.5 Pro	Google	1m	No	5.25	★★☆	★★★	★★☆
Gemini 1.5 Flash	Google	1m	No	0.53	★★★	★★☆	★★★
Gemma 7B	Google	8k	Yes	0.2	★★★	★☆☆	★★★
Claude 3 Opus	Anthropic	200k	No	30	★☆☆	★★★	★☆☆
Claude 3 Sonnet	Anthropic	200k	No	6	★☆☆	★★☆	★★☆
Claude 3 Haiku	Anthropic	200k	No	0.5	★★★	★★☆	★★★
Command R +	Cohere	128k	Yes	6	★☆☆	★★☆	★★☆
Command R	Cohere	128k	Yes	0.75	★★☆	★☆☆	★★☆
Llama 3 70B	Meta AI	8k	Yes	0.93	★★☆	★★★	★☆☆
Llama 3 8B	Meta AI	8k	Yes	0.2	★★★	★☆☆	★★★
Code Llama	Meta AI	16k	Yes	0.9	★★☆	★★★	★☆☆
Mistral Large	Mistral AI	32k	No	12	★☆☆	★★★	★☆☆
Mistral Medium	Mistral AI	32k	No	4.05	★★☆	★★☆	★☆☆
Mistral Small	Mistral Ai	32k	No	2.25	★★☆	★☆☆	★★☆
Mixtral 8x22B	Mistral AI	65k	Yes	1.2	★★☆	★★☆	★★☆
Mixtral 8x7B	Mistral AI	32k	Yes	0.5	★★★	★☆☆	★★★
Mistral 7B	Mistral AI	32k	Yes	0.2	★★★	★☆☆	★★☆
DBRX	Databricks	32k	Yes	1.4	★★☆	★★☆	★★★

here is a table summarizing the most well-known LLMs in existence, they are compared on different criteria such as the context window, if the LLM is open source, the quality of response and the speed

- **Context Window:** The context window is the number of tokens (word/letter) that an LLM can consider before giving a response. So, the larger the context window, the larger the prompt that can be sent to it.
- **Open source:** The code to operate or train the model is published. This allows developers to review it, modify it, and even reuse it. The trained parameters, which are essential for using it directly, are also published. However, there are different cases of open source involving different use cases like MIT Licence which allows commercial use or not, there is also CC BY-NC Licence which prohibits commercial use unless authorized and last one License governing ethical or social values who some templates include clauses to ensure that they are not used for harmful, discriminatory, or unethical activities. BUT open-source doesn't mean free API uses !
- **Quality:** the quality of an LLM comes from the answer it gives, this quality is judged on different aspects such as quality, relevance or other. on the LLM arena website, which is a community site, you can find a ranking of LLMs following their answers. the site simply works a question is asked to 2 LLMs who are anonymized and the community votes for the preferred answer, so the classification is done in this way. [35]

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	2	Gemini-Exp-1121	1365	+8/-6	5625	Google	Proprietary
1	1	ChatGPT-4o-latest (2024-11-20)	1361	+4/-5	10658	OpenAI	Proprietary
3	5	Gemini-Exp-1114	1344	+4/-5	12778	Google	Proprietary
4	2	o1-preview	1334	+4/-4	27835	OpenAI	Proprietary
5	7	o1-mini	1308	+3/-4	31992	OpenAI	Proprietary
5	5	Gemini-1.5-Pro-002	1301	+5/-3	27336	Google	Proprietary
7	10	Grok-2-08-13	1289	+4/-3	52102	xAI	Proprietary
7	12	Yi-Lightning	1287	+4/-3	29336	01 AI	Proprietary
7	5	GPT-4o-2024-05-13	1285	+2/-2	111745	OpenAI	Proprietary
8	3	Claude 3.5 Sonnet (20241022)	1282	+4/-3	29454	Anthropic	Proprietary
10	17	Athene-v2-Chat-72B	1274	+8/-6	4354	NexusFlow	NexusFlow

- **Speed:** speed is based on response speed and prompt analysis speed.

But to use LLMs, we must go through providers to use them, openai is the chatGPT provider, some providers offer several LLMs like huggigface which offers mistral or claude but moreover certain LLMs are only available from certain providers like chatGPT at openai. we find several providers which offer API services for the use of LLM, here is a table which summarizes the most interesting for our project:

Providers	Provides limits/notes	Model name	Model limits
Grocq	/	Various models of **Llama**	14,400 requests/day 30,000 tokens/minute or 7,000 requests/day 7,000 tokens/minute
Google AI Studio	Data is used for training (when used outside of the UK/CH/EEA/EU).	Various models of **Gemini**	32,000 tokens/minute 50 requests/day 2 requests/minute or 1,000,000 tokens/minute 1,500 requests/day 15 requests/minute
Lambda Labs	Requires credit card verification.	Various models of **Llama**	/
Mistral	Free tier (Experiment plan) requires opting into data training, requires phone number verification.	Open and Proprietary Mistral models	1 request/second 500,000 tokens/minute 1,000,000,000 tokens/month
HuggingFace	Limited to models smaller than 10GB. Some popular models are supported even if they exceed 10GB.	Various open models	1,000 requests/day (with an account)
SambaNovaCloud	/	Various models of **Llama**	1 or 10 or 20 or 30 requests/minute
Cerebras	Waitlist Free tier restricted to 8K context	Various models of **Llama**	30 requests/minute 60,000 tokens/minute 900 requests/hour 1,000,000 tokens/hour 14,400 requests/day 1,000,000 tokens/day
OVH AI	/	Various models of **Llama** and Mistral	12 requests/minute
Cloudflare	10 000 tokens/day	Various models of **Llama**, **Mistral** and others	/

General recommendations

Provider Recommendation

HuggingFace stands out as the most suitable provider due to its free API usage, extensive library of LLMs, and flexibility in adapting models to specific use cases. Its generous usage limits align perfectly with the scale and duration of our project.

LLM Recommendation

Among the evaluated models, Mistral 7B offers the best combination of response quality, speed, and a large context window, all while being free to use under permissive open-source terms. Its smaller size also ensures faster inference times, making it ideal for real-time chatbot interactions.

While other models like GPT-4 may offer higher response quality, Mistral 7B provides performance that is more than adequate for our target use cases, balancing quality, speed, and cost-efficiency.

Conclusion

In conclusion, for our project, we recommend using the Mistral 7B model via the HuggingFace API.

This combination provides an optimal balance of performance, accessibility, and cost-efficiency, ensuring a robust foundation for building an inclusive and high-quality chatbot.

2.4 Dataset

After evaluating several options, the **Acceslibre** database emerged as the most suitable for our project. This database, managed by the French government, offers several advantages:

1. **Comprehensive Coverage:** Acceslibre includes data on over **525,000 places across France**, ensuring a wide geographic scope. This is essential for providing relevant information to users throughout the country.
2. **Free API Access:** The database provides an easy-to-use, free API, which is crucial for minimizing the operational costs of our project.
3. **Data Reliability:** Being a government-backed initiative, the database guarantees a high level of accuracy and credibility, which is critical for building trust with users.
4. **Format Compatibility:** The data is available in formats (e.g., JSON via REST API) that are easy to integrate into modern applications, making it technically adaptable to our chatbot's architecture.

Comparison with Other Databases

While other databases were considered, they proved less suitable for the following reasons:

- **Smaller Datasets:** Many alternatives lack the extensive coverage offered by Acceslibre, focusing only on specific regions or cities.
- **Geographic Scope:** Some databases are designed for other countries, making them irrelevant to our project's goals.
- **Access and Integration:** Few alternatives provide the same level of API access or are formatted for seamless integration.

Integration in the Project

The Acceslibre database will be used to provide real-time information on the accessibility of public places. This includes responding to user queries about specific locations and enriching the chatbot's responses with reliable, up-to-date information.

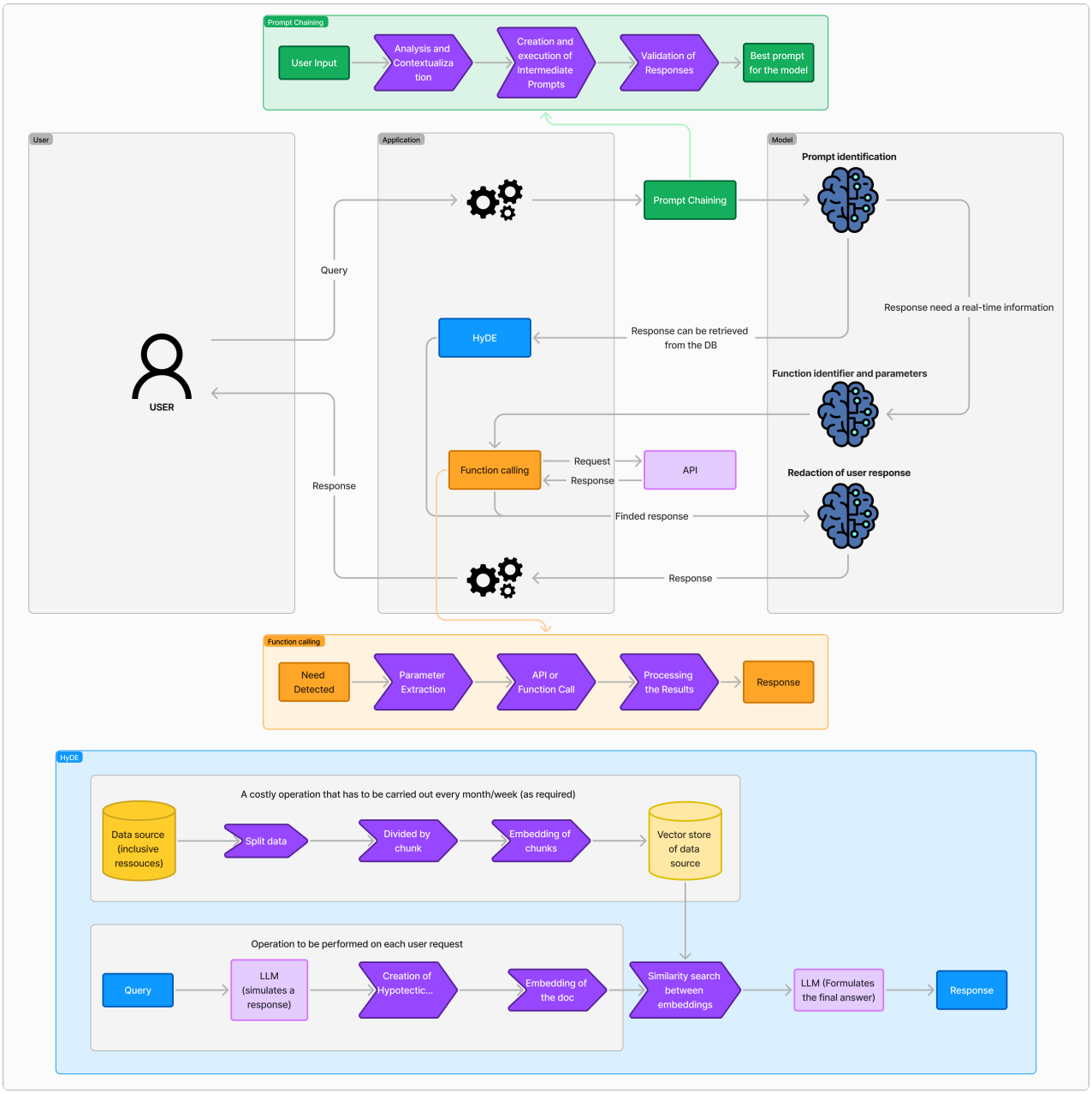
Limitations and Future Enhancements

Although the Acceslibre database is comprehensive, potential gaps or outdated information may arise in certain regions. To address this:

- Additional data sources could be integrated in the future to complement the dataset and provide broader or more granular coverage.

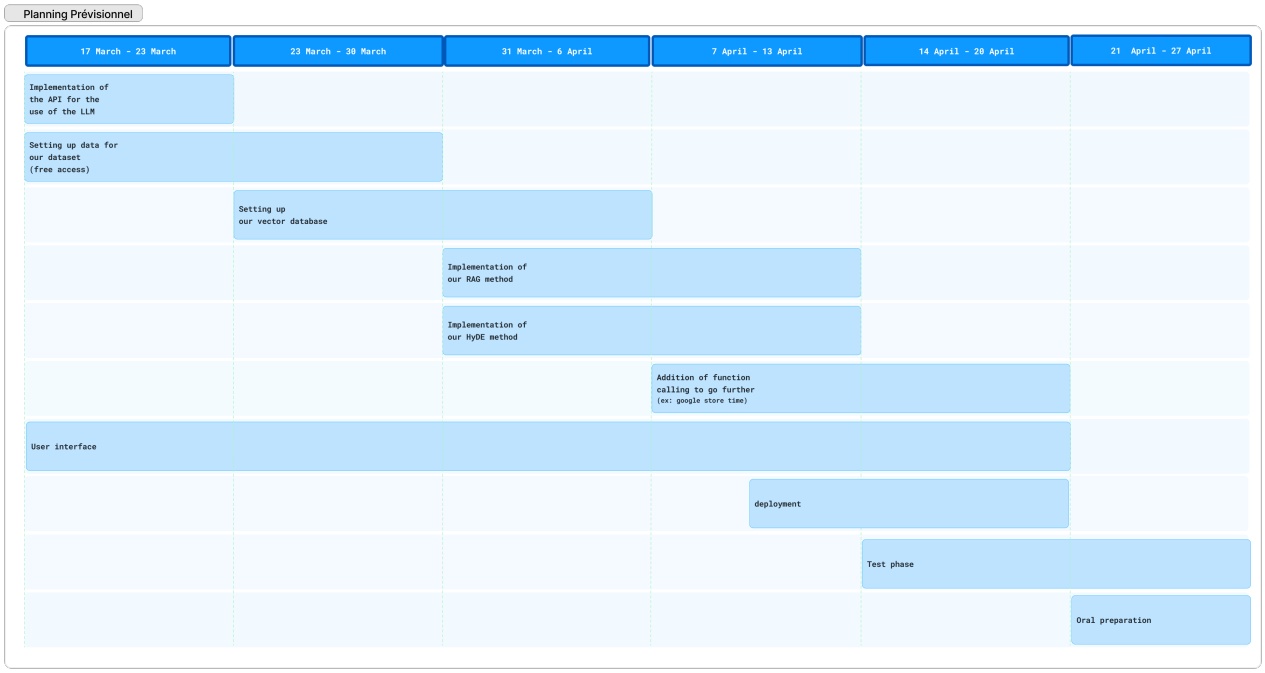
3. Proof Of Concept

3.1 Realization



The diagram above illustrates the operation of an inclusive multi-channel chatbot, designed to adapt to a variety of communication environments to ensure optimal accessibility. The modular architecture enables in-depth analysis and personalized response to user queries.

3.2 Provisional schedule



So, based on what we have seen previously, we can now establish a provisional schedule for the next phase of our project.

4. Conclusion

Despite significant progress in chatbot development, inclusivity remains an underexplored domain. Integrating robust data sources and addressing the diverse needs of users with disabilities are critical for creating impactful solutions. This project aims to address these gaps by developing a multichannel, inclusive chatbot capable of understanding and assisting users across a wide range of scenarios, such as finding accessible restaurants or navigating public spaces.

5. References

[1] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A N. Gomez, Ł Kaiser, and I Polosukhin. Attention Is All You Need

[2] G Yenduri, Ramalingam M, C Selvi G, Supriya Y, G Srivastava, P K R Maddikunta, D Raj G, R H Jhaveri, Prabadevi B, W Wang, A V. Vasilakos, and T R Gadekallu. GPT – A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions

[3] H Naveed, A U Khan, S Qiuc, Muhammad Saqib, S Anwar, M Usman, N Akhtar, N Barnes, A Mian. A Comprehensive Overview of Large Language Models

[4] Z Zhou, J Song, K, Yao, Z Shu, and L Ma. ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning

[5] X Dong, S Wang, D Lin, G K Rajbahadur, B Zhou, S Liu, and A E. Hassan. PromptExp: Multi-granularity Prompt Explanation of Large Language Models

[6] Y Zheng, H Y Koh, J Ju, A T.N. Nguyen, L T. May, G I. Webb, S Pan. Large Language Models for Scientific Synthesis, Inference and Explanation

[7] E Hu, Y Shen, P Wallis, Z Allen-Zhu, Y Li, S Wang, L Wang, and W Chen. LORA: Low-rank adaptation of LLM

[8] L Wang, N Yang, X Huang, L Yang, R Majumder, F Wei. Large Search Model: Redefining Search Stack in the Era of LLMs

[9] A Korinek. Generative AI for Economic Research: Use Cases and Implications for Economists

[10] K M Jablonka, Q Ai, A Al-Feghali, S Badhwar, J D. Bocarsly, A M Bran. Examples of How LLMs Can Transform Materials Science and Chemistry: A Reflection on a Large Language Model Hackathon

[11] F F. Xu, U Alon, G Neubig, and V J. Hellendoorn. A systematic evaluation of large language models of code

[12] Z Qiang, K Taylor, W Wang and J Jiang. OAEI-LLM: A Benchmark Dataset for Understanding LLM Hallucinations in Ontology Matching

[13] R Lapid, R Langberg and M Sipper. Open sesame! Universal black-box jailbreaking of LLM

[14] D Lu, Y Deng, W J. Padilla, J M. Malof. Can large language models learn the physics of metamaterials? An empirical study with chatgpt

[15] O Topsakal and T C Akinci. Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast

[16] P Macedo, R N Madeira, P Miranda and P A Santos. Crafting Personalised Web Interfaces: Enhancing Accessibility for Persons with Disabilities

[17] IMPACT AI - IA et inclusion

[18] P Esquivel, K Gill, M Goldberg, S. A Sundaram, L Morris and D Ding. Voice Assistant Utilization among the Disability Community for Independent Living: A Rapid Review of Recent Evidence

[19] https://python.langchain.com/en/latest/
[20] https://platform.openai.com/docs/guides/gpt/function-calling
[21] Hugging Face Blog on Advanced Prompt Engineering: https://huggingface.co/blog/advanced-prompt-engineering
[22] https://www.promptingguide.ai/applications/function_calling
[23] https://www.promptingguide.ai/techniques/prompt_chaining
[24] https://langraph.com
[25] https://python.langchain.com/en/latest/
[26] OpenAI Community Discussions on Workflow Tools: https://community.openai.com/
[27] Introduction to StreamLit
[28] https://www.datacamp.com/fr/tutorial/streamlit
[29] Introduction to RAG
[30] Implementing RAG with OpenAI
[31] RAG and Practical Applications
[32] https://onopia.com/le-retrieval-augmented-generation-rag-une-approche-innovante-pour-la-generation-de-texte-et-la-reponse-aux-questions/
[33] https://www.apside.com/fr/blog/le-modele-rag/
[34] Introduction to HyDE
[35] Chatbot Arena LLM Leaderboard: Community-driven Evaluation for Best LLM and AI chatbots
[36] Accesslibre
[37] Wheelmap
[38] AccessNow
[39] Handicap.fr
[40] Ava
[41] Be My Eyes
[42] Seeing AI
[43] Transkriptor
[44] Tobii Dynavox