UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «AI-Based Model for Knowledge Specific Assistance»

ΠΜΣ «Μοντέλο Τεχνητής Νοημοσύνης για Βοήθεια σε Συγκεκριμένη Γνώση»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**AI-Based Model for Knowledge Specific Assistance** Μοντέλο Τεχνητής Νοημοσύνης για Βοήθεια σε Συγκεκριμένη Γνώση
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Thanos Apostolou** Θάνος Αποστόλου
Father's name: Πατρώνυμο:	**Christos** Χρήστος
Student's ID No: Αριθμός Μητρώου:	**MPSP2203** ΜΠΣΠ2203
Supervisor: Επιβλέπων:	**Dionisios Sotiropoulos, Assistant Professor** Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

September 2024/ Σεπτέμβριος 2024

3-Member Examination Committee Τριμελής Εξεταστική Επιτροπή		
Dionisios Sotiropoulos **Assistant Professor** Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής		

Contents

Abstract

This MSc thesis is about utilizing artificial intelligence models in order to find specific knowledge. As part of this goal we will develop a complete web application, where users will be able to ask questions to artificial intelligence models, which will answer them based on a specific context. We will follow two different methodologies. For the first methodology we will create our own text generation AI model [1] which will be trained to understand specific knowledge. For the second methodology, we will use existing artificial intelligence models, trying to limit them so that they respond only to the specific knowledge context that we have chosen. In the end we will be able to come to conclusions about the usefulness of these methodologies.

Περίληψη

Η παρούσα μεταπτυχιακή εργασία ασχολείται με την αξιοποίηση μοντέλων τεχνητής νοημοσύνης για την υποβοήθηση ανεύρεσης συγκεκριμένης γνώσης. Στα πλαίσια αυτού του στόχου θα αναπτύξουμε μια πλήρη διαδικτυακή εφαρμογή, στην οποία οι χρήστες θα μπορούν να κάνουν ερωτήσεις σε μοντέλα τεχνητής νοημοσύνης, τα οποία θα τους απαντάνε με βάση συγκεκριμένο πλαίσιο. Θα ακολουθήσουμε δύο διαφορετικές μεθοδολογίες. Για την πρώτη μεθοδολογία θα δημιουργήσουμε ένα δικό μας μοντέλο τεχνητής νοημοσύνης παραγωγής κειμένου [1] το οποίο θα εκπαιδευτεί για να κατανοεί συγκεκριμένη γνώση. Για την δεύτερη μεθοδολογία θα χρησιμοποιήσουμε υπάρχοντα μοντέλα τεχνητής νοημοσύνης προσπαθώντας να τα περιορίσουμε ώστε να απαντάνε μόνο στο συγκεκριμένο πλαίσιο γνώσης που έχουμε επιλέξει. Στο τέλος θα μπορέσουμε να καταλήξουμε σε συμπεράσματα. Στο τέλος θα μπορέσουμε να καταλήξουμε σε συμπεράσματα για την χρησιμότητα αυτών των μεθοδολογιών. [2]

1. Introduction

In our era, the knowledge we have acquired is bigger than ever. The number of books, notes, web pages and other forms of content keeps increasing year by year. It is impossible for any human being, to be able to read an process all this available knowledge. Fortunately, technology has been greatly improved and is being used daily for tasks involving knowledge search and analysis. While traditional tools like search engines made it easier for us to find existing knowledge, in the past years we have observed the increasing development of tools using artificial intelligence. We will study the usage of text generation machine learning models in specific knowledge search and analysis assistance. We will use two different methodologies for these tasks and we will develop a full web application with which users will be able to ask questions

In chapter 2 we will describe and analyze the fundamental theoretical concepts needed for better understanding of this thesis. We will also describe the various technologies and their advantage, which we will use for our application development and deployment.

In chapter 3 we will dive in the details of the two methodologies that we will use. We will compare them and we will describe their advantages and disadvantages.

In chapter 4 we will describe the architecture and the implementation of our application. We will show the components which construct our application, the tasks each component can perform and how they are connected together.

In chapter 5 we will show the design and execution results of our deployed application. We will investigate the various ways in which our application can be used by the users in order to find specific knowledge based on raw data like documents or web pages.

In chapter 6 we will write our conclusions we reached. We will describe the problems and limitations we faced. Finally, we will specify future improvements that can be made as well as future goals about scaling and expand the core idea.

2. Theory and Literature Review

In this chapter we will talk about the theoretic terms that this thesis is based upon. We will also describe the main technologies which we will use.

2.1. Theoretic Terms

The fundamental theoretic concepts of this thesis stem from the study field of artificial intelligence. We will describe the connection between artificial intelligence, machine learning and deep learning. We will focus on a specific category of machine learning models involving text generation and the state of the art practices of utilizing their capabilities to the maximum by using Retrieval Augmented Generation (RAG) technique.

2.1.1. Artificial intelligence

In the general sense, Artificial intelligence (AI) is intelligence exhibited by machines, particularly computer systems. It is a field of research in computer science that develops and studies methods and software that enable machines to perceive their environment and use learning and intelligence to take actions that maximize their chances of achieving defined goals. Such machines may be called AIs. [3]

Intelligence can be considered to be a property of internal thought processes and reasoning, or a property of intelligent behavior, an external characterization. From these two dimensions (human vs. rational and thought vs. behavior) there are four possible combinations. The methods used are necessarily different: the pursuit of human-like intelligence must be in part an empirical science related to psychology, involving observations and hypotheses about actual human behavior and thought processes; a rationalist approach, on the other hand, involves a combination of mathematics and engineering, and connects to statistics, control theory, and economics. These 4 approaches are the following:[2]

- Acting humanly: The Turing test approach The Turing test, proposed by Alan Turing (1950) and it consists of 4 core principles that a computer would need to follow in order to pass it.
  - natural language processing to communicate successfully in a human language
  - knowledge representation to store what it knows or hears
  - automated reasoning to answer questions and to draw new conclusions
  - machine learning to adapt to new circumstances and to detect and extrapolate patterns

  The full turing test is completed with 2 additional characteristics which have been added by later researchers:
  - computer vision and speech recognition to perceive the world
  - robotics to manipulate objects and move about
- Thinking humanly: The cognitive modeling approach We can determine if a computer or a program thinks like a human by analyzing the human thought in 3 main concepts:
  - introspection - trying to catch our own thoughts as they go by
  - psychological experiments - observing a person in action
  - brain imaging - observing the brain in action
- Thinking rationally: The “laws of thought” approach Rationally thinking can be achieved by following the rules defined by the “logic” study field. When conventional logic requires knowledge

that cannot be obtained realistically, then the theory of probability helps us define logical thinking.

- Acting rationally: The rational agent approach Rational thinking can achieve a construction of a comprehensive model of rational thought, but cannot generate intelligent behavior by itself. A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.

2.1.2. Machine Learning

We described the fundamental concepts with which artificial intelligence is defined. Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data and thus perform tasks without explicit instructions. [4]

Machine learning is a subset of artificial intelligence (AI) focused on developing algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, these systems learn and improve from experience by identifying patterns in data. Machine Learning uses algorithms and statistical models to enable computers to perform specific tasks without being explicitly programmed to do so. Machine learning systems learn from and make decisions based on data. The process involves the following steps:

- Data Collection: Gathering relevant data that the model will learn from.
- Data Preparation: Cleaning and organizing data to make it suitable for training.
- Model Selection: Choosing an appropriate algorithm that fits the problem.
- Training: Using data to train the model, allowing it to learn and identify patterns.
- Evaluation: Assessing the model's performance using different metrics.
- Optimization: Fine-tuning the model to improve its accuracy and efficiency.
- Deployment: Implementing the model in a real-world scenario for practical use.

There are 4 basic types of Machine Learning: [4]–[6]

- Supervised Learning: The model is trained on labeled data, meaning the input comes with the correct output. The goal is to learn a mapping from inputs to outputs. Examples: Regression, classification.
- Unsupervised Learning: The model is trained on unlabeled data, and it must find hidden patterns or intrinsic structures in the input data. Examples: Clustering, association.
- Semi-Supervised Learning: Combines a small amount of labeled data with a large amount of unlabeled data during training. It falls between supervised and unsupervised learning.
- Reinforcement Learning: The model learns by interacting with an environment, receiving rewards or penalties based on its actions, and aims to maximize the cumulative reward. Examples: Game playing, robotic control.

Deep learning is a subset of machine learning that uses multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain [7]. Deep learning is being used in order to teach computers how to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions. Deep learning methods can be used in order to automate tasks that typically require human intelligence, such as describing images or transcribing a sound file into text [8]. We can visualize the subsets of Deep Learning, Machine Learning and Artificial Intelligence with the diagram below:

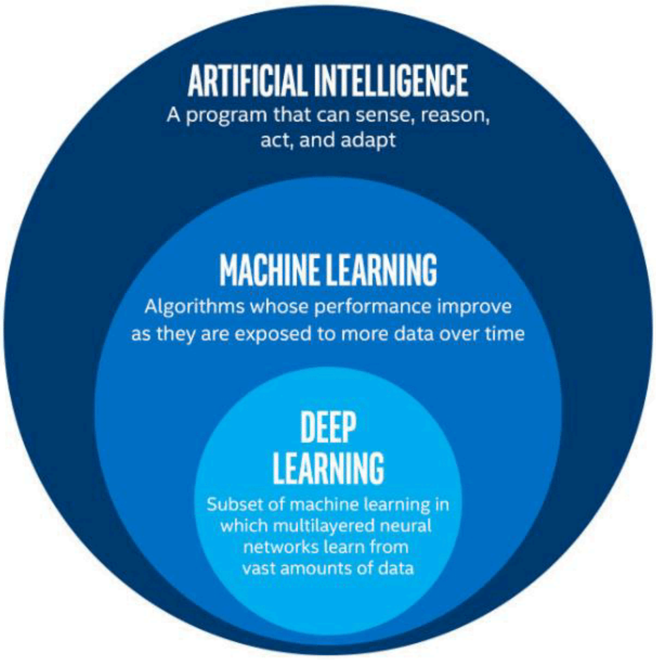
ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

IMAGE 1: Venn Diagram for AI, ML, Deep Learning [9]

Artificial Intelligence, Machine Learning and Deep Learning are involved in many applications like Image Recognition, Speech Recognition, Traffic prediction, Recommender Systems, Self-driving cars, Email Spam and Malware Filtering, Virtual Personal Assistant, Fraud Detection, Stock Market trading, Medical Diagnosis, Automatic Language Translation, Chatbots, Generation of text images and videos. [10]–[12]. All these applications required different artificial intelligence disciplines that can be combined in order to create a complete artificial intelligence system which produces the required output.

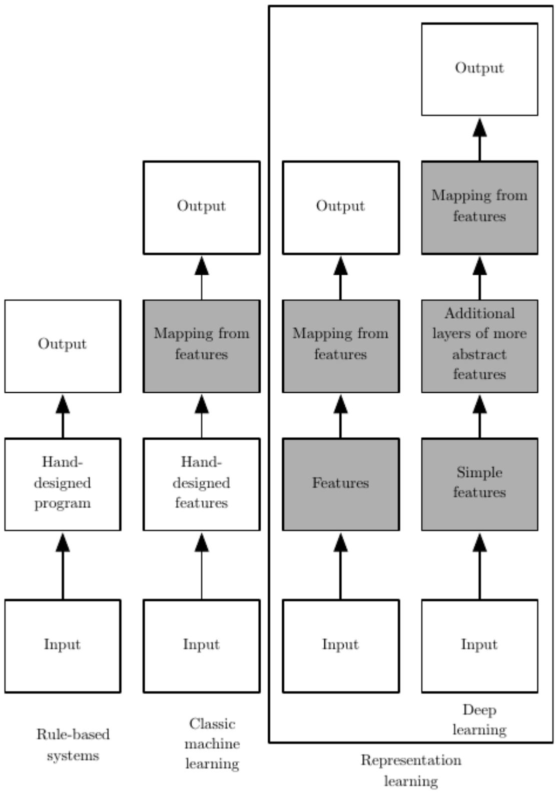

IMAGE 2: Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data. [13]

2.1.3. Text Generation Models, LLM

We described the fundamental concepts of artificial intelligence and machine learning. Now we will take a closer look into a specific category o machine learning models which are used in text generation tasks.

Generative AI refers to deep-learning models that can generate high-quality text, images, and other content based on the data they were trained on. [14]. A text generation model is a type of generative AI models which is designed to produce coherent and contextually relevant textual content. These models are typically based on natural language processing (NLP) techniques and are trained in text data to learn the patterns, grammar, and context required to generate human-like text. When these these models are trained in huge sets of data and have been fed enough examples to be able to recognize and interpret human language or other types of complex data, then they are called large language models (LLM) [15].

These are the key components and concepts of text generation models:

- Training Data:
  - Corpora: Large collections of text used to train the model. These can include books, articles, websites, dialogues, and other text sources.
  - Preprocessing: Cleaning and organizing the text data, including tokenization (breaking text into words or subwords), removing special characters, and normalizing text.
- Model Architecture:

  - Recurrent Neural Networks (RNNs): Earlier models for text generation, including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), which handle sequential data by maintaining context over time.
  - Transformers: Modern architecture that has become the standard for NLP tasks. Transformers use self-attention mechanisms to process entire sequences of text at once, allowing for better handling of context and dependencies over long distances in the text. Examples include the GPT (Generative Pre-trained Transformer) series, BERT (Bidirectional Encoder Representations from Transformers), and others.

- Training Process:
  - Unsupervised Learning: Most text generation models are trained using unsupervised learning, where the model learns to predict the next word or sequence of words based on the context provided by preceding text.
  - Fine-Tuning: After pre-training on a large corpus, models are often fine-tuned on specific datasets to adapt them to particular tasks or domains.

- Generation Techniques:
  - Sampling: Randomly selecting the next word from the probability distribution generated by the model.
  - Beam Search: An algorithm that searches for the best sequence of words by considering multiple candidate sequences at each step and selecting the most likely ones.
  - Temperature Adjustment: Modifying the probability distribution to control the randomness of the generated text. Lower temperatures result in more deterministic outputs, while higher temperatures produce more diverse and creative text.

  Usually the most popular LLMs have these parameters in order to control sampling. Parameter “top_k” limits the model’s output to the top-k most probable tokens at each step. This can help reduce incoherent or nonsensical output by restricting the model’s vocabulary. Parameter “top_p” filters out tokens whose cumulative probability is less than a specified threshold (p). It allows for more diversity in the output while still avoiding low-probability tokens. Temperature adjusts the randomness or confidence level of the model’s predictions by scaling the log probabilities. Higher temperatures lead to more diverse but potentially nonsensical outputs, while lower temperatures yield more focused and predictable responses [16], [17].

- Evaluation:
  - Perplexity: A measure of how well a probability model predicts a sample. Lower perplexity indicates better performance.
  - Human Evaluation: Assessing the coherence, relevance, and fluency of the generated text through human judges.
  - Automated Metrics: BLEU (Bilingual Evaluation Understudy), ROUGE (Recall-Oriented Understudy for Gisting Evaluation), and other metrics comparing the generated text to reference texts.

2.1.4. Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) is a technique for augmenting LLM knowledge with additional data. LLMs can reason about wide-ranging topics, but their knowledge is limited to the public data up to a specific point in time that they were trained on. If you want to build AI applications that can reason about private data or data introduced after a model’s cutoff date, you need to augment the knowledge of the model with the specific information it needs. The process of bringing the appropriate information and inserting it into the model prompt is known as Retrieval Augmented Generation (RAG) [18]. RAG extends the already powerful capabilities of LLMs to specific domains or an organization's internal knowledge base, all without the need to retrain the model. It is a cost-effective approach to improving LLM output so it remains relevant, accurate, and useful in various contexts [19].

RAG is important because of these reasons [19]:

- LLMs have known drawbacks:
  - Presenting false information when it does not have the answer.
  - Presenting out-of-date or generic information when the user expects a specific, current response.
  - Creating a response from non-authoritative sources.
  - Creating inaccurate responses due to terminology confusion, wherein different training sources use the same terminology to talk about different things
- RAG comes with additional benefits:
  - Cost-effective implementation: The computational and financial costs of retraining text generation model for organization or domain-specific information are high. RAG is a more cost-effective approach to introducing new data to the LLM.
  - Current information: RAG allows developers to provide the latest research, statistics, or news to the generative models. They can use RAG to connect the LLM directly to live social media feeds, news sites, or other frequently-updated information sources. The LLM can then provide the latest information to the users.
  - Enhanced user trust: RAG allows the LLM to present accurate information with source attribution. The output can include citations or references to sources. Users can also look up source documents themselves if they require further clarification or more detail. This can increase trust and confidence in your generative AI solution.
  - More developer control: With RAG, developers can test and improve their chat applications more efficiently. They can control and change the LLM's information sources to adapt to changing requirements or cross-functional usage. Developers can also restrict sensitive information retrieval to different authorization levels and ensure the LLM generates appropriate responses. In addition, they can also troubleshoot and make fixes if the LLM references incorrect information sources for specific questions. Organizations can implement generative AI technology more confidently for a broader range of applications.

A typical RAG application has two main components [18]:

- Indexing: a pipeline for ingesting data from a source and indexing it. This usually happens offline.
  - Load: First we need to load our data. This is done with Document Loaders.
  - Split: Text splitters break large Documents into smaller chunks. This is useful both for indexing data and for passing it in to a model, since large chunks are harder to search over and won't fit in a model's finite context window.
  - Store: We need somewhere to store and index our splits, so that they can later be searched over. This is often done using a VectorStore and Embeddings model.
- Retrieval and generation: the actual RAG chain, which takes the user query at run time and retrieves the relevant data from the index, then passes that to the model.
  - Retrieve: Given a user input, relevant splits are retrieved from storage using a Retriever.
  - Generate: A ChatModel / LLM produces an answer using a prompt that includes the question and the retrieved data

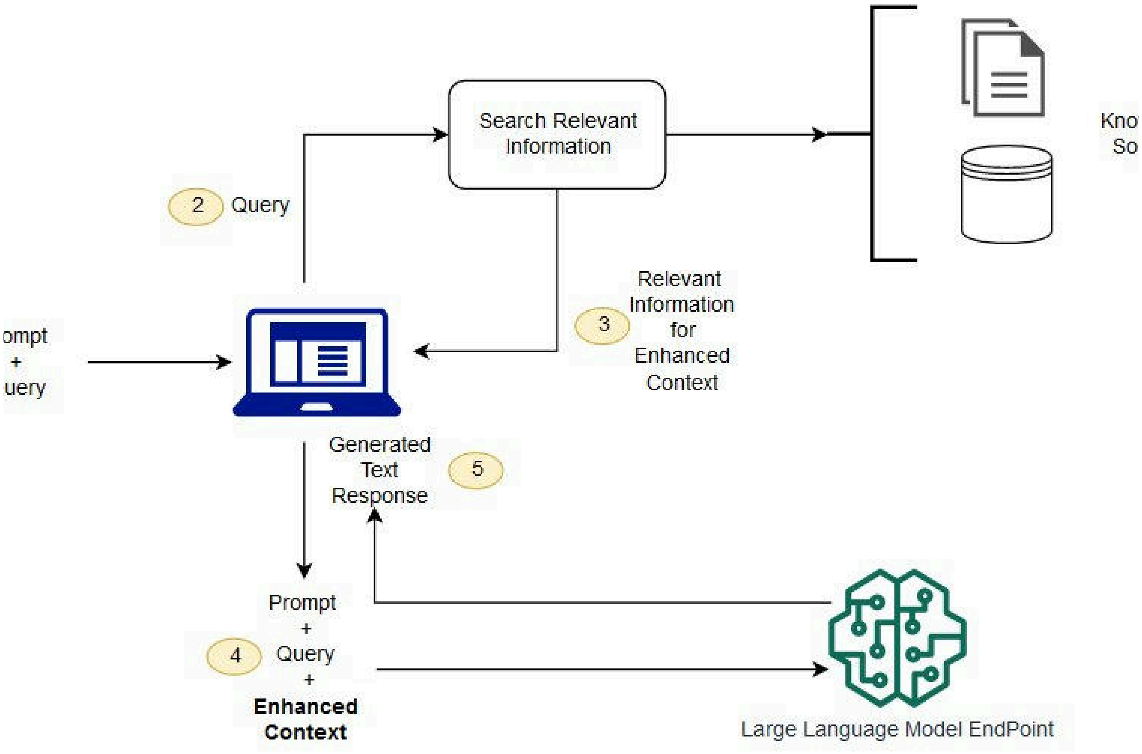
Search Relevant Information
Kno
So
2 Query
Relevant Information for Enhanced Context
3
ompt + uery
Generated Text Response
5
Prompt + Query + Enhanced Context
4
Large Language Model EndPoint

IMAGE 3: Conceptual flow of using RAG with LLMs. [19]

2.2. Technologies

In the context of this thesis we will use many technologies in order to produce a complete application. We will describe the main programming languages we used for this application, Rust and Python. We will see more details about the core programming libraries our application is using and their basic features we utilize. Finally we will talk about the state of the art deployment procedure of deployments based on containers utilization.

2.2.1. Python Programming Language

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a “batteries included” language due to its comprehensive standard library. [20]

Choosing python as a programming language has several benefits [21]

- Developers can easily read and understand a Python program because it has basic, English-like syntax.
- Python makes developers more productive because they can write a Python program using fewer lines of code compared to many other languages.
- Python has a large standard library that contains reusable codes for almost any task. As a result, developers do not have to write code from scratch.

- Developers can easily use Python with other popular programming languages such as Java, C, and C++.
- The active Python community includes millions of supportive developers around the globe. If you face an issue, you can get quick support from the community.
- Plenty of helpful resources are available on the internet if you want to learn Python. For example, you can easily find videos, tutorials, documentation, and developer guides.
- Python is portable across different computer operating systems such as Windows, macOS, Linux, and Unix.

The python programming language is very popular in various applications: [21]

- Server-side web development: Server-side web development includes the complex backend functions that websites perform to display information to the user. For example, websites must interact with databases, talk to other websites, and protect data when sending it over the network.
- Graphical User Interfaces development (GUI): Develop user friendly GUI applications while also providing a good user experience.
- Gaming development: Python can be used to develop games. Popular python libraries exist for both 2D and 3D graphics development.
- Automation with Python scripts: A scripting language is a programming language that automates tasks that humans normally perform. Such tasks usually include:
  - filesystem operations like files renaming, reading, writing, converting.
  - Mathematical operations
  - download content
  - text operations and transformations in files
  - basic log analysis
- Data science and machine learning: Data science is extracting valuable knowledge from data, and machine learning (ML) teaches computers to automatically learn from the data and make accurate predictions.
- Software deployments and operations: Python can be used for different development tasks and software applications such as:
  - Automatically building the software
  - Utilize continuous integration and continuous deployments (CI/CD)
  - Handling software project management
- Software test automation: Check whether the actual results from the software match the expected results to ensure that the software is error-free.

Using python programming language has various advantages [22]:

- Presence of third-party modules: Python has a rich ecosystem of third-party modules and libraries that extend its functionality for various tasks.
- Extensive support libraries: Python boasts extensive support libraries like NumPy for numerical calculations and Pandas for data analytics, making it suitable for scientific and data-related applications.
- Open source and large active community base: Python is open source, and it has a large and active community that contributes to its development and provides support.
- Versatile, easy to read, learn, and write: Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.
- User-friendly data structures: Python offers intuitive and easy-to-use data structures, simplifying data manipulation and management.

- High-level language: Python is a high-level language that abstracts low-level details, making it more user-friendly.
- Dynamically typed language: Python is dynamically typed, meaning you don't need to declare data types explicitly, making it flexible but still reliable.
- Object-Oriented and Procedural programming language: Python supports both object-oriented and procedural programming, providing versatility in coding styles.
- Portable and interactive: Python is portable across operating systems and interactive, allowing real-time code execution and testing.
- Ideal for prototypes: Python's concise syntax allows developers to prototype applications quickly with less code.
- Highly efficient: Python's clean design provides enhanced process control, and it has excellent text processing capabilities, making it efficient for various applications.
- Internet of Things (IoT) opportunities: Python is used in IoT applications due to its simplicity and versatility.
- Interpreted language: Python is interpreted, which allows for easier debugging and code development.

However python programming language has also some drawbacks [22]:

- Performance: Python is an interpreted language, which means that it can be slower than compiled languages like C or Java. This can be an issue for performance-intensive tasks.
- Global Interpreter Lock: The Global Interpreter Lock (GIL) is a mechanism in Python that prevents multiple threads from executing Python code at once. This can limit the parallelism and concurrency of some applications.
- Memory consumption: Python can consume a lot of memory, especially when working with large datasets or running complex algorithms.
- Dynamically typed: Python is a dynamically typed language, which means that the types of variables can change at runtime. This can make it more difficult to catch errors and can lead to bugs.
- Packaging and versioning: Python has a large number of packages and libraries, which can sometimes lead to versioning issues and package conflicts.
- Lack of strictness: Python's flexibility can sometimes be a double-edged sword. While it can be great for rapid development and prototyping, it can also lead to code that is difficult to read and maintain.

2.2.2. Rust Programming Language

Rust is a general-purpose programming language emphasizing performance, type safety, and concurrency. It enforces memory safety, meaning that all references point to valid memory, without a garbage collector. To simultaneously enforce memory safety and prevent data races, its "borrow checker" tracks the object lifetime of all references in a program during compiling. Rust was influenced by ideas from functional programming, including immutability, higher-order functions, and algebraic data types. It is popular for systems programming.Rust does not enforce a programming paradigm, but supports object-oriented programming via structs, enums, traits, and methods, and supports functional programming via immutability, pure functions, higher order functions, and pattern matching. [23]

Rust programming languages can be used in many applications [24], [25]:

- Server-side web development: Rust can be used to write simple REST apis or even full stack backend applications which connect to databases and serve complex html pages using various template engines.

- Client-side web development: Rust can be compile to WebAssembly and be used to create client side web applications supported in all modern web browsers.
- Graphical User Interfaces development (GUI): Develop user friendly GUI applications while also providing a good user experience. Rust gui frameworks can use different architecture including Elm, immediate mode, reactive and others.
- Gaming development: Rust can be used to develop games. New modern rust libraries have emerged for both 2D and 3D graphics development.
- Embedded development: Due to low resource usage and high performance, rust can be used to develop applications for low resource devices.
- Command line development: Rust can be used to create both command line interfaces (CLI) as well as terminal user interfaces (TUI).
- Internet of Things development (IoT): IoT devices typically have limited resources, and Rust's memory safety and low-level control make it an excellent choice for developing embedded systems.
- Robotics: Robotics requires real-time processing, and Rust's low-level control and memory safety make it ideal for developing real-time applications. Rust's concurrency features make it possible to handle multiple threads efficiently, which is essential in robotics applications.
- Machine Learning: Rust libraries and ecosystem for machine learning development is newer and currently more limited than Python's. However rust can be preferred for its higher performance, the memory safety and its immutability features.

Using rust programming language has various advantages [26]:

- Memory Safety: Rust's borrow checker ensures memory safety without the overhead of garbage collection. This means fewer memory leaks and crashes.
- Performance: Comparable to C and C++, Rust provides fine-grained control of memory and other resources.
- Concurrency: Rust's ownership model makes concurrent programming more manageable and less prone to bugs.
- Modern Tooling: Cargo, Rust's package manager and build system, is highly praised for its ease of use.
- Vibrant Community: Rust has a growing and enthusiastic community, which results in good documentation, community support, and an expanding ecosystem.

However rust programming language has also some drawbacks [26]:

- Steep Learning Curve: Rust's unique features, like ownership and lifetimes, can be challenging to grasp for newcomers.
- Compilation Time: Rust programs can have longer compile times compared to some other languages.
- Lesser Library Support: While growing, the Rust ecosystem is still smaller than those of older languages like C++, Java or Python.

2.2.3. Python Libraries

We will talk about the most important Python libraries that we use for our application and their important parts that we utilize.

One of the most significant libraries we use is NLTK. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project [27]. NLTK comes with many corpora, toy grammars, trained models, etc which are called NTLK Data [28]. One of the most important modules of NLTK is the “tokenize” module which can help us split a long text into sentences or into words.

pytorch: TODO
llama-cpp: TODO
matplotlib: TODO
huggingface-hub: TODO
langchain: TODO
unstructured: TODO
fais: TODO

2.2.4. Rust Libraries

We will talk about the most important Rust libraries that we use for our application and their important parts that we utilize.

clap: TODO
tokio: TODO
axum: TODO
sea-orm: TODO

2.2.5. Typescript, HTML, CSS, Vue.js

TODO

2.2.6. Containers, Docker and Kubernetes

TODO

3. Specific Knowledge Assistance Approaches

3.1. Custom Text Generation Model Method

3.2. Retrieval Augmented Generation Method

4. System Architecture

TODO

5. Usage and Execution of the Application

TODO

6. Conclusions and Future Work

TODO

Bibliography

[1] H. Face, “Text Generation.” [Online]. Available: https://huggingfagce.co/tasks/text-generation

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th Global Edition. Pearson, 2021.

[3] Wikipedia, “Artificial intelligence.” [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence

[4] Wikipedia, “Machine Learning.” [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[5] GeeksforGeeks, “Types of Machine Learning.” [Online]. Available: https://www.geeksforgeeks.org/types-of-machine-learning

[6] lakeFS, “Machine Learning Components: Elements & Classifications.” [Online]. Available: https://lakefs.io/blog/machine-learning-components/

[7] J. Holdsworth and M. Scapicchio, “Deep learning.” [Online]. Available: https://www.ibm.com/topics/deep-learning

[8] Amazon, “Deep learning.” [Online]. Available: https://aws.amazon.com/what-is/deep-learning/

[9] R. Khalkar, A. S. Dikhit, and A. Goel, “Handwritten Text Recognition using Deep Learning (CNN & RNN),” *International Advanced Research Journal in Science, Engineering and Technology*, 2021, [Online]. Available: https://www.researchgate.net/publication/353939315_Handwritten_Text_Recognition_using_Deep_Learning_CNN_RNN

[10] javatpoint, “Applications of Machine learning.” [Online]. Available: https://www.javatpoint.com/applications-of-machine-learning

[11] GeeksforGeeks, “Applications of Machine learning.” [Online]. Available: https://www.geeksforgeeks.org/machine-learning-introduction/

[12] C. Staff, “10 Machine Learning Applications.” [Online]. Available: https://www.coursera.org/articles/machine-learning-applications

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Artificial Intelligence: Deep learning - adaptive computation and machine learning*. 2016.

[14] K. Martineau, “What is generative AI.” [Online]. Available: https://research.ibm.com/blog/what-is-generative-AI

[15] Cloudflare, “What is a large language model (LLM).” [Online]. Available: https://www.cloudflare.com/learning/ai/what-is-large-language-model/

[16] Ruman, “Setting Top-K, Top-P and Temperature in LLMs.” [Online]. Available: https://rumn.medium.com/setting-top-k-top-p-and-temperature-in-llms-3da3a8f74832

[17] A. Verma, “Understanding temperature, top_p, top_k, logit_bias in LLM parameters.” [Online]. Available: https://aviralrma.medium.com/understanding-llm-parameters-c2db4b07f0ee

[18] LangChaing, “Build a Retrieval Augmented Generation (RAG) App.” [Online]. Available: https://python.langchain.com/v0.2/docs/tutorials/rag/

[19] Amazon, “What is Retrieval-Augmented Generation.” [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/

[20] Wikipedia, “Python (programming language).” [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language)

[21] Amazon, “What is Python.” [Online]. Available: https://aws.amazon.com/what-is/python/

[22] GeeksforGeeks, “Python Language advantages and applications.” [Online]. Available: https://www.geeksforgeeks.org/python-language-advantages-applications/

[23] Wikipedia, “Rust (programming language).” [Online]. Available: https://en.wikipedia.org/wiki/Rust_(programming_language)

[24] Rust, “Rust.” [Online]. Available: https://www.rust-lang.org/

[25] C. Mittal, “10 Best Use Cases of Rust Programming Language in 2023.” [Online]. Available: https://medium.com/@chetanmittaldev/10-best-use-cases-of-rust-programming-language-in-2023-def4e2081e44

[26] M. Tawfik, “Rust Language: Pros, Cons, and Learning Guide.” [Online]. Available: https://medium.com/@apicraft/rust-language-pros-cons-and-learning-guide-594e8c9e2b7c

[27] N. Project, “Natural Language Toolkit.” [Online]. Available: https://www.nltk.org/

[28] N. Project, “Installing NLTK Data.” [Online]. Available: https://www.nltk.org/data.html