OULUN AMMATTIKORKEAKOULU

Kim Satokangas

Artificial Intelligence & Large Language Models

Cooperative AI story writing

Artificial Intelligence & Large Language Models

Cooperative AI story writing

Kim Satokangas
Final projects
Spring 2024
Degree programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology, Option of Web Development

---

Author: Kim Satokangas
Title of the thesis: Artificial Intelligence & Large Language Models
Thesis examiner(s): Lasse Haverinen
Term and year of thesis completion: 2024 Pages: 37

Ever since break troughs in machine learning and creation of Large Language Machines, interest in Artificial Intelligences has been on the rise. This thesis looks into what the Artificial Intelligences and Large Language Models are.
As part of the thesis, a demonstration project was coded, this project used Large Language Models and Artificial Intelligence to generate a story and an image to accompany the story. The project was written with python and used remotely hosted Large Language Model and generative Artificial Intelligence for the image and text generation.
The finished project demonstrates cooperation between Artificial Intelligences and acts as a proof of concept. Networking Artificial Intelligences to work together in parallel or in sequence can be beneficial in refining the output into specific form or in increasing overall efficiency of the system.

Keywords: Artificial Intelligence, Large Language Model, Generative AI

CONTENTS

1 INTRODUCTION

With the advent of modern Artificial Intelligence and their rise in popularity, it will be possible to create increasingly complex artificial intelligence solutions. With multiple Artificial Intelligence's with different specializations working together and complementing each others work, it will be possible to create more complex and multi-levelled products. This thesis work will look into what Artificial intelligence is, what are Large Language Models and how multiple Artificial Intelligence's can work together.

I got the idea for this specific thesis topic while I was doing my practical training where we used artificial intelligence as a part of the project. In the project we used Large Language Models as coding assistants in figuring out some errors and problems with the code. This gave me the idea of doing my thesis work about Large Language Models. I thought of doing a project where the AIs would do something together. Combining this base idea with my own writing hobby, I arrived at the final idea for the thesis. To use LLMs in a cooperative environment to write short stories accompanied by AI generated images.

1.1 Thesis Objectives

The main objective of the thesis is to show multiple Artificial Intelligence's, from now on shortened to AI, working together. To demonstrate this a story writing program will be coded that uses different AIs to create a short story and generate an image related to the story.

Primary goals the story writer should be able to accomplish are:

- Have an AI write a story from user prompt
  - Story will be the main text body the AI generates from the prompt.
- Have an AI summarize the story into single scene
  - The scene will be a short summary of the whole story. It is supposed to explain what the story is about.
- Have an AI create image from the summarized scene.
  - The image generated by the AI will be illustration of the story.
- Show the story and image side by side.

Secondary goals the story writer should be able to accomplish are:

- Have an option to use local run AI to write the story.
- Have an option to use local run AI to summarize the scene.
- Have an option to use local run AI to create the image.
- Option to continue the story by writing a new paragraph and create another image.
- Option to translate the story from English to Finnish using AI.
- Have an AI tuned text to speech to narrate the story.

The main programming language that will be used is Python supplemented with other languages as necessary.

2 ARTIFICIAL INTELLIGENCE, WHAT IS AI

Artificial intelligence in its base form is a machine, or software, that is capable of completing tasks and making decisions to some degree without separate user input. They try to mimic problem solving capabilities of humans to enable machines to do more complex decisions. Advanced AIs are able to learn from newly gathered and processed data from tasks they perform to make their answers more accurate. Current artificial intelligences are not capable of abstract thoughts or improvising solutions. Actions and answers from the artificial intelligence come as responses to patterns it has learned from the data that was used to train the AI. Therefore, current AI is not truly intelligent or self-aware, and they are not true artificial intelligences for that reason. (1.)
AI excel at finding patterns in data and utilizing these patterns to find solutions and answers, they can go through enormous amounts of data in much shorter time than what human worker, or even multiple workers, would be capable of doing. Properly set up AIs are also highly accurate and reliable in their given tasks, and because they never tire, AIs are especially a good fit for repetitive tasks that require high accuracy over long durations. (2.)

2.1 Brief History of AI

The concept of artificial intelligence as a serious academic topic dates back to 1940s and 1950s. Alan Turing wrote a paper on machines simulating humans and being able to think for themselves. (3.)

John McCarthy, a young professor at Darthmouth College, organized a summer workshop that was to be held in the summer of 1956 to develop the ideas behind thinking machines. He chose the term artificial intelligence to be used for this new field of research. This was the first time the term Artificial Intelligence was used. In addition to being the first time the term AI was used, the workshop also produced the first example of an AI. The Logic Theorist was a program created by three participants of the summer workshop, Allan Newel, Herbert Simon and Cliff Shaw. A program that used brute force to find mathematical proof for theorems in Principa Mathematica. (4.)

While initially there was keen interest in possibilities for artificial intelligences, limited real world applications severely reduced the interest in artificial intelligence. By 1974 AI research entered

the first AI Winter, a term used to describe the times of reduced funding and slowed research progress in the field of artificial intelligences. The lack of progress drove the investors away to more pragmatic solutions.
In 1980 expert systems, an artificial intelligence using if-then rule pattern, brought in renewed interest in artificial intelligence and increases in funding. This AI boom was relatively short lived, because as early as 1987 the short comings of if-then rule based artificial intelligences and the lack of hardware technology to support more complex machines became apparent. Research on artificial intelligences entered the second AI winter. (5.)

In 1994 the progress on computer hardware and memory allowed research on artificial intelligences to pick up speed again. Interest was further boosted when Deep Blue, AI developed by IBM, beat chess champion Gary Kasparov in a chess game under standard tournament rules. (6.)

The development of artificial intelligences showed steady, albeit slightly slow progress, until 2011. The introduction of massive data sets, known as big data, and even more powerful hardware in conjunction with breakthroughs in machine learning and the advent of deep learning, launched AI development into another boom. These advances made it ultimately possible for AlphaGO, AI developed by google, to beat Lee Sedol, the world champion of Go at the time, in a game of Go. (7.)

2.2 Modern AI

Modern artificial intelligences are usually a form of Generative AI. These AIs are able to take in as input wide variety of different kinds of data and output new content based on the inputs given. For example, it is possible to input text into an AI model and it will give you an image as the output data. The three main types of generative AI are Digital Twins, Synthetic data generation and Large Language Models.
Digital Twins are doubles of real-life systems, they mirror and mimic all the parts of the physical system and utilize generative and non-generative AI to predict how the given system will behave and test its functionality.
Synthetic Data Generation creates usable synthetic data that can be used in place of real-world data. It can fill or expand data sets for machine learning as the data generated follow the same

trends, properties and probabilities as the real data its trained on. These systems are extremely useful in situation where data privacy would be an issue when using real life data.
Large Language Models are AI algorithms that can utilize extremely huge datasets for their content generation. There will be more about Large Language Models later in this thesis. (8.)

Modern AIs differ from the expert system type AIs in that they generate their own abstract rules as they go through the learning data. Expert systems have clearly defined rules and states that all have been carefully designed one by one by humans. These systems cannot handle scenarios outside of their designed environments. (9.)

3 TYPES OF AI

There are different ways to categorize artificial intelligences theoretically. Two of the most common ones are separating different types of artificial intelligence, by their capability or functionality.

3.1 AI Types Separated By Capability

Separating artificial intelligence by their capabilities into three different types according to their capabilities compared to humans.
Artificial Narrow Intelligence, which has a more narrow range of abilities compared to humans. Artificial General Intelligence, which is artificial intelligence on par with humans. And Superintelligence, which is artificial intelligence that surpasses human capabilities. (10.)

3.1.1 Artificial Narrow Intelligence

Artificial Narrow Intelligence (from now on ANI) is a lower level artificial intelligence which is commonly used for specific, and often singular, tasks. They might be more efficient than humans in their specific field, but lack intelligence in general to be comparable and adaptability to do tasks outside of their own fields. These artificial intelligences are fed data on their field of work and this data is used to teach them how to solve the tasks they are responsible for.
All current artificial intelligences belong to this type. Good examples would be most well-known AI implementations, such as OpenAI ChatGPT or Apples Siri digital assistant. (10.)

3.1.2 Artificial General Intelligence

Artificial General Intelligence (from now on AGI) is artificial intelligence on par with humans with their capabilities. This kind of AI has the ability to think as humans do, comprehend new concepts and improvise solutions. Rather than only imitating the human mind, they understand the human mind. They are capable of wide range of tasks and can learn new tasks on their own as humans would instead of needing supervised training.
There are no examples of this kind in real life yet, and they live only in pages of fiction. Companies and government entities are investing a huge amount of resources into being the first ones to

make AGI a real thing. Different kinds of robotic assistants that would fit this type of AI are common in media. For example droids from Star Wars universe are capable of doing most jobs humans could do, but are still generally only on par and not better at them. (10.)

3.1.3 Artificial Superintelligence

Artificial Superintelligence (from now on ASI) is when artificial intelligence goes beyond human capabilities. ASI is purely a theoretical concept as of now and they will not become a reality for a long time. They would be able to think faster and solve more complex problems than what humans would be able to do. ASI would be fully self-aware, with their own desires, emotions and beliefs they would develop as their ability to comprehend world around them increases.
ASI will surpass humans in most fields and would constantly self improve. In theory it would be the last invention humans would need to make, because ASI would take over any technological research and would do it faster and better than humans. ASI are only found in science fiction, and are almost always contained in some way to reduce their capabilities form their full potential. Cortana from Halo universe and HAL from 2001: A Space Odyssey are some examples of ASI in science fiction. (10.)

3.2 AI Types Separated By Functionality

Separating artificial intelligence by their functionality means classifying them by how they use their learned data, respond to input and react to the environment around them. AIs are generally separated into four different types by functionality, Reactive Machine, Limited Memory, Theory of Mind and Self-Aware. (11.)

3.2.1 Reactive Machine AI

Reactive Machine Artificial Intelligence react to input from their environment. They have no memory or very limited one. Answer only to immediate input and are incapable of learning or improving. Additionally the inputs they can react to are severely limited further limiting their capabilities. They are highly specialized, only able to do work in the specific tasks they were designed for.

This kind of AI are good for repetitive, autonomous tasks such as sorting items or recommending movies based on your watch history. (11.)

3.2.2 Limited Memory AI

Limited Memory AI have a working memory where they can save past events. As such they are able to observe things over time and react accordingly to events. The events saved in its memory are used for decision making along with instructions it has learned from its training data sets. Limited Memory AIs can be used to complete wide array of asks and can be used as part of complex implementations as driving a car, like Tesla autopilot, and AI assistants, like Apple Siri. (11.)

3.2.3 Theory of mind AI

Theory of mind AI is the next step in development of artificial intelligences and they do not exist yet. These AIs are able to form an understanding of the world surrounding them and the entities that inhabit that world. This understanding includes them knowing that the entities in this world can have their own thoughts, emotions and motives that can affect their behaviour. This enables them to blend in with humans and seamlessly work together. (11.)

3.2.4 Self-Aware AI

Self-Aware AI has a sense of self and consciousness. They can think for themselves and have wants and needs. They also know they have wants and needs and this affects their decision making. Self-Aware AIs can experience emotions such as frustration and anger for not having their needs met, or gratefulness and appreciation for receiving help. They know that their actions can affect others and their emotional states. (11.)

4 LARGE LANGUAGE MODELS

When most people today think of artificial intelligences, they are thinking of large language models. These models are able to process datasets containing vast amount of data and find the relations between different sequential data points. They can use the patterns they learn from these learned data relationships to create human like content such as images, sentences and sounds. They can even mimic specific humans or characters. These AIs are so convincing in their mimicry that some people believe they are sentient and first artificial general intelligences. This however, is not the case, these AIs only find relations in data points from their training data and give answers to the input that they have calculated to be the most probable. They do not think, have opinions or feel emotions. (12.)

4.1 How do Large Language Models work?

Large Language Models find the relations in the data they are trained on and use that data to construct their outputs that can be images, text or even sound depending on how the model in question is meant to be used. The LLMs use the relations they have learned in predicting which token comes next in the series of tokens given in the input for the LLM. As such, by design they mimic the ways humans produce text. (13.)

Large Language Models utilize unsupervised learning to find previously unknown relations and patterns in their unlabelled training datasets. Being able to use unlabelled data also eliminates one of the biggest problems when it comes to gathering data for AI training.

The data sets used for the training are very large, with the adjustable training parameters often being counted at billions. It is common practise to include the count of parameters in the name of the AI model. For example Metas llama-2-7b has been trained with a seven billion training parameters and codellama-70b has been trained with 70 billion training parameters.

These parameters include all the weights that the LLM adjusts during its training process, as well as the biases that are the baselines for the LLM before adjustments. In general, parameter count can be roughly used to estimate the complexity and processing power of an LLM. Larger parameter counts also lead to higher computational power demands. Parameter count is not all that matters though, a smaller LLM with less parameters, but with more fine-tuned training or more

advanced algorithms can out perform LLMs with higher parameter counts, but less efficient training and algorithms. (14.)

4.1.1 Tokenization and tokens

Large Language Models, and Artificial Intelligence in general, do not work well with text and non numerical data. Tokenization helps the LLM to more efficiently work with the data it receives or generates by slicing it into smaller parts, called tokens, and giving these tokens numerical IDs the LLM can use to recognize them. These IDs are shared between identical tokens and can be single characters, common sets of characters, whole words, partial words, whole sentences or phrases. While the length of tokens varies greatly, the average length of a token in general is roughly four characters. (15.)

There are many different kinds of tokenization algorithms. One such algorithm is byte-pair encoding (BPE), which is used by OpenAI chatGPT-3.5 and GPT-4. BPE as a compression algorithm was first described in 1994 and later modified and adopted as one of the common choices for word segmentation tokenizers for LLMs. BPE works by finding frequently appearing characters or character chains and converting them into a single token. Figure 1 shows an example of how OpenAI does byte-pair encoding tokenization. (16.)

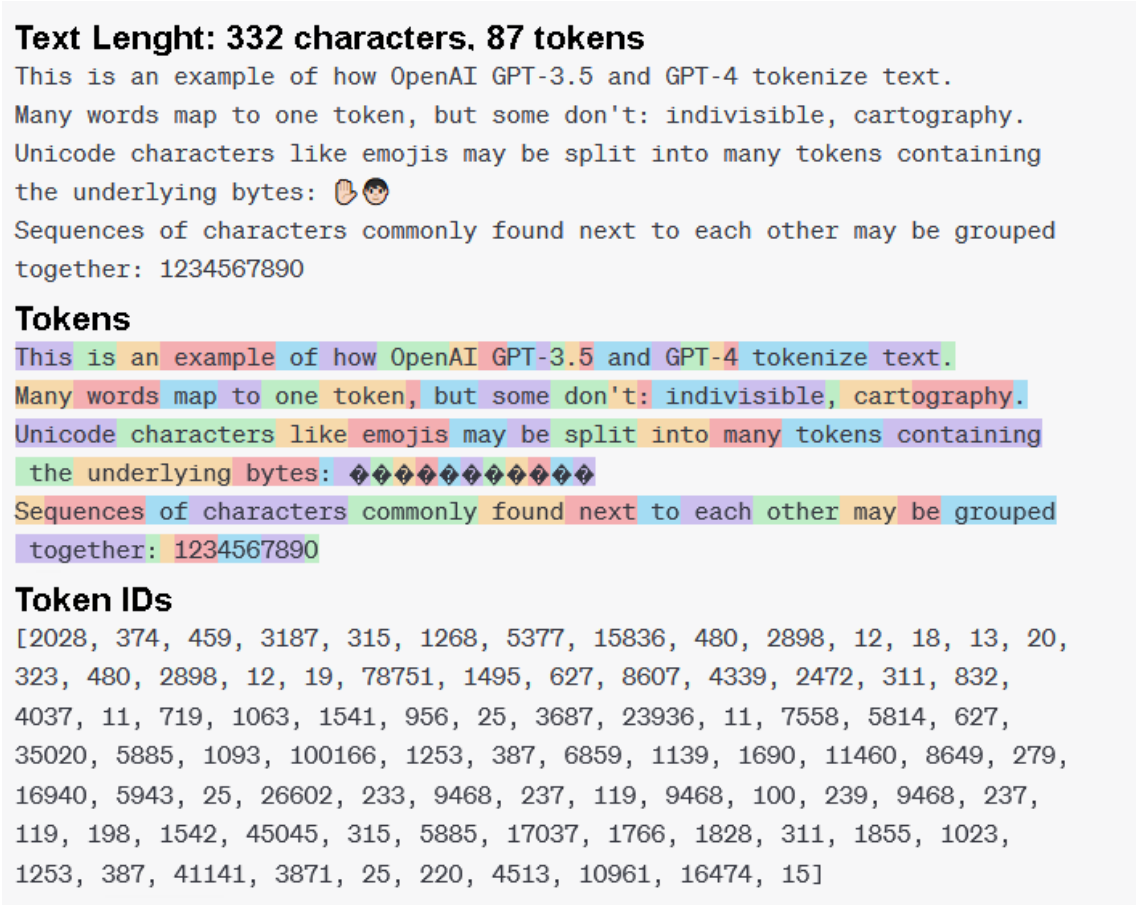

FIGURE 1. Byte-Pair Encoding tokenization example. (17.)

SentencePiece is a slightly more complex tokenizer that supports BPE as its segmentation algorithm. SentencePiece is language agnostic and treats all input text as plain Unicode characters and white spaces as just another symbol. Converting all white spaces into a "_" (U+2581) symbol. This allows it to process languages that do not use white spaces to separate words as well as ones that do. (18; 19; 20;)

4.1.2 How does Large Language Model remember things?

LLMs do not remember things as humans do, their memory consists of the prompt that is input into it and this is in general the whole context they will be working with when generating an answer. For example chatbots, like OpenAI ChatGPT, have the whole conversation placed into the prompt each time the user sends a message to them. This forms an illusion of memory for the user and makes the chatbot seem more intelligent, as it can reverence older messages. In long conversations the chatbot will start to forget things that were first discussed. This happens be-

cause the LLM removes the first tokens to enter the conversation when its token limit is reached. After the LLM starts to forget things its performance drops drastically and it could be said it has reached the end of its life cycle. (21.)

There are some different ways to work around or improve the short memory problem LLMs suffer from. One example of trying to fix the problem is the method created by a team from Massachusetts Institute of Technology by researching the core reason AIs start to forget topics, which led to applying improved practises into handling the deletion of tokens. They named this approach StreamingLLM. It works by keeping the attention sink of four first tokens and deleting the token following them while holding on to the same key values. So after fifth token is deleted, the sixth token will be fifth in the line but will still be labelled as the sixth token. This considerably increases the lifespan of the model before memory issues becomes a problem again. (22; 23;)

4.1.3 Prompts and Prompt engineering

Prompt is what the input for an LLM is called. The prompt includes all the details and parameters the LLM needs to process an answer. The input prompt body is usually in text format, but can also include images, video and audio. (24.)

Prompt engineering is the practise of improving and fine tuning the prompts itself to make the LLM being prompted to behave in desired manner. Making even small slight changes to the prompt can drastically alter the output of the LLM, as such the quality of the prompt is directly related to the quality of the responses. There are many different kinds of prompting techniques that have been developed as a result of prompt engineering evolving further.

Some examples are Zero-Shot Prompting and Few-Shot Prompting. Zero-Shot prompting offers more data inside the prompt, that has been fine tuned for specific tasks, removing the need for extensive training data for the model. Few-Shot Prompting provides the model examples of input and output pairs which can help the model to understand the logic behind the prompt. (25; 26; 27;)

4.2 Multimodal Large Language Models, image and sound generation

Large Language Models are by design good at processing languages in text format. They encode and tokenize text into usable data and generate answers from the received input prompts by decoding the tokenized data according to their predictions.

When Large Language Models ability to process data is expanded to include audio or visual data, or both, in addition to text data, it becomes multimodal. Multimodal Large Language Models have more complex encoder-decoders, which have the ability to work with text as well as with vision or audio data, and a some type of modality interface that acts as the connection between the LLM and the non-text format data. These modality interfaces can in general be separated into two groups, token-level and feature-level, depending on how they handle the merging of data into LLM usable format. Token-level interface takes the tokens output by the encoder and concatenates them with the text tokens. Feature-level interface adds special modules that allow interactions between the text and visual features. (28.)

4.3 Different ways to utilize Large Language Models

Large Language Models have a wide variety of uses in many fields. They can be utilized in some way for almost anything where it is possible to gather large enough dataset to train one. As such LLMs see use in diverse field of different industries and in personal use.

Some example use cases for Large Language Models:

- Chatbots
    - Chatbots like OpenAI ChatGPT are one of the most prolific use cases for LLMs. As shown in the figure 2, they can hold conversations and mimic human interactions like answering questions. (29.)

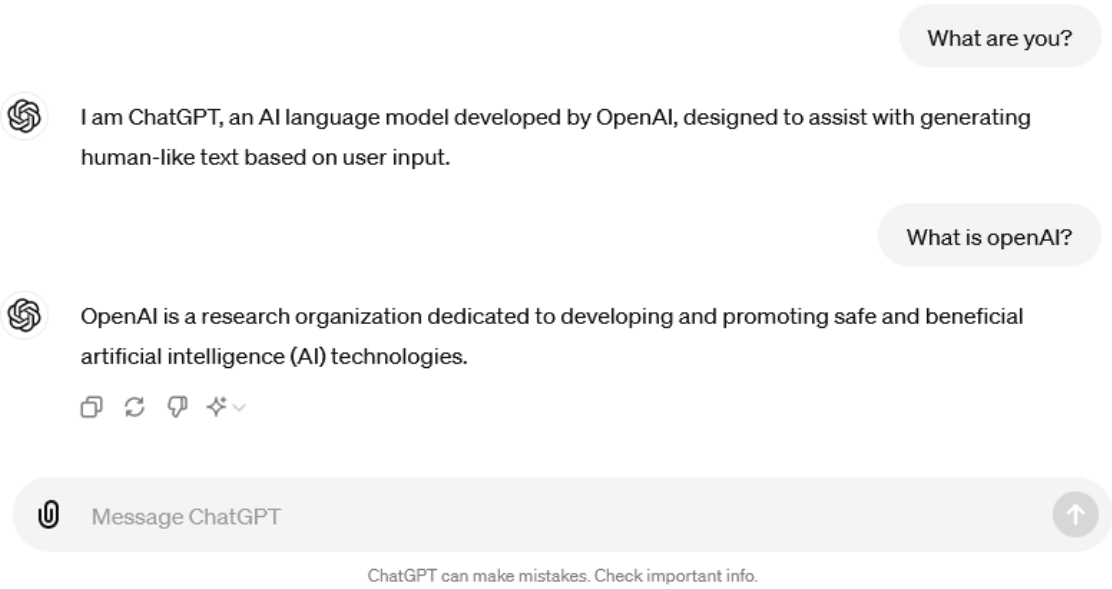

FIGURE 2. Example conversation with ChatGPT.

- Coding Assistants
  - LLMs can be fine tuned to assist developers with coding tasks. (30.)
- Translations
  - LLMs can be leveraged to improve translation tools, even in real time. (31.)
- Customer support and service
  - LLM powered chatbots can be used as a first step in customer support and services. (32.)
- Understanding and summarizing audio
  - LLMs with audio recognition can be used to summarize and edit audio. (33.)
- Database Management
  - LLMs can find use as Database Administrators to automate tedious tasks and help ease the access into the database. (34.)
- Disease outbreak tracking
  - Using LLMs to analyse and process articles in multiple languages to detect progress and spread of disease outbreaks. (35.)
- Fraud detection
  - LLMs can be used to assist in different kinds of fraud detection in the financial sector. (36.)
- Learning languages
  - Duolingo uses LLM powered AI to assist in creation of language lessons. (37.)

- Recommendation algorithms
    - Amazon has integrated LLM into their recommendation algorithm to better analyse the preferences of users to make their recommendations more personal and accurate. (38.)

4.4 What made Large Language Models possible

The three main factors that made development of Large Language Models possible were the availability of data, computing power and advancements in AI algorithms.

LLMs need huge amounts of data for their training sets. Training data was not conveniently available in the past, there was no sufficiently large and diverse data sets that could be used for training the LLMs. With the explosive growth of available data through the internet and storage options being more varied and spacious than ever, gathering sufficient training data has been simplified a lot.

LLMs require a lot of computing power for their training, operation and fine-tuning. The amount of time it takes to train a LLM model could easily take years before which now would be completed in fraction of the time. With advances in computing technology and advent of cloud computing, the resources to research, train and run LLMs has become widely available to developers.

Advancements in AI algorithms paved way for the creation of the transformer architecture model, which could be considered a breakthrough moment in LLM development. Transformer model is the basis of most modern LLMs. (39; 40;)

5 STORY WRITER DESIGN

The story writer will have three AIs working together to create a multi-page story from prompt given by the user. First AI will write a story based on the prompt, this story is given to another AI to write a summary. The summary will be then used by yet another AI to create an image. The story and the image will be shown to the user in book style format. With the story being on the left page and image showing up on the right page. There will also be a possibility to continue the story based on the already generated story. Text-to-speech can be used to narrate the story.

5.1 Story text and summary

Story generation and summary functions will be separated to their own python file, which will be imported into the main program and called from there. Prompt parameters will be restricted so that the resource usage will not be too high and the LLMs can function within the free preview allowance.

The story text generation function will accept the prompt as an input, as well as integer value for customizing the random seed used to generate the answers by the LLM. After generating the story text, the function will set it as the text inside a textbox on the left side of the current page and forward the text to the summary function.

The summary function will take the full story text and summarize it using the connection to the LLM. After the summary has been created, it will be used as input for the image generation AI. This summary will not be visible to the user and is only used to create the image related to the generated story. The summary function will share the random seed with the story generation function.

5.2 Image Generation

The image generation function will be separated into its own python file as well. The image generation function will take the summary created by summary function as input and use it to generate an image. Generated image size will be set to lower size to fit the program screen and to save on resources. After generating the image, it will be saved inside the directory of the program. The

image will be set to the right side of the current page. The same random number seed which the text generation functions use will be used for the image generation function.

5.3 User interface

The user interface will roughly mimic a book, having the pages in the middle and the controls above and below the pages itself. The controls for input and initiating the story generation will be below the pages itself, as well as the controls for browsing the pages and starting narration. The overall plan for the user interface is shown in the figure 3 below.

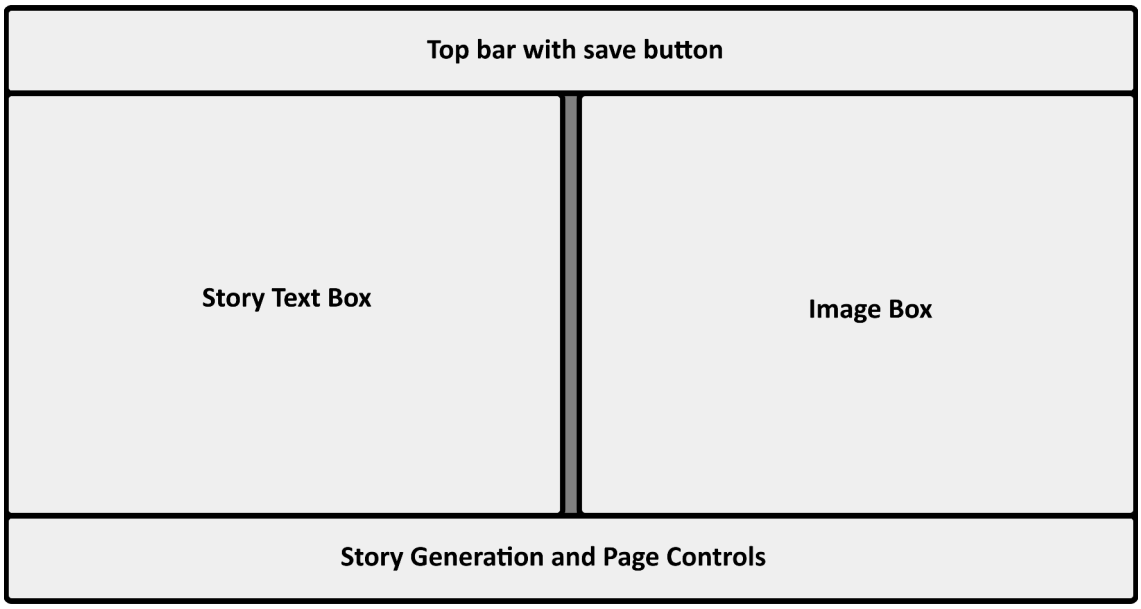

FIGURE 3. User interface design plan

6 STORY WRITER IMPLEMENTATION

Starting the practical part of the project, first step was to research and plan what kind of tools would be used in it. Once the tools were mostly planned out, the actual coding and development part was separated into three. The first part would be the development of the story text and summary generator. The second part would be the image generation and the last part would be creation of user interface and polishing, making sure all the parts worked together.

For the tools used in the project, python will be the main programming language. Pythons Tkinter library will be used to create the user interface for the program.

Llama-2-7b, created by Meta, was chosen for the story generation and summarization. This model is hosted on replicate and will be accessed remotely. The main reason for choosing this model was its availability. Story generator and the summarization could operate within the free limit of the model so there would be no risk of any sudden problems with losing access to the LLM.

StabilityAI will be used to generate the image to accompany the story. It is hosted on Stability's own site and will also be accessed remotely. They offer limited free credit for image generation which should be enough to last till the end of the project if used correctly.

At first gTTS python library was chosen to handle the text-to-speech narration, but it was later changed to Pyttsx3. The reason was that gTTS would have had to save the narration to a file before it would have been able to be played. Pyttsx3 is able to handle the conversion of the story text into speech and playing the narration with no extra libraries needed for playing sounds. The project itself will be hosted in GitHub.

6.1 Story text and summary

The connection to the LLM model that handles the story generation and summarization is handled by API call. As both the story and summary functions use the same model but in separate calls, only one API key was needed. This api key is hidden inside a ".env" file and the text generation will not run without it. A simple dummy page was created using Tkinter for purpose of testing the LLM text generation.

Write story function would use the prompt given by the user as part of the prompt sent to the StoryAI. The prompt consists of the prompt body as well as the instructions used to set up the

LLM for its role as a storyteller and the max token count. Tokens were restricted to 300 in the prompt and given a maximum new token limit of 350 to ensure that the model does not encounter memory limits and because the 300 token answers were just long enough to fit on one page of the virtual book. The prompt also contains temperature and repetition penalty parameters. Temperature is the randomness of the models output, the higher it is the more diverse the answers will be over multiple generations. Repetition penalty penalizes the model for repeating tokens and makes it consider using different ones. After some testing both of these were left at one for the story writer. Once the function receives an answer from the API, it will loop through it and turn it into a single string, store it into a variable and set it as the story shown on the current page. After all this is done, the function automatically sends the story forward to the summary function. Story generation functions work flow is shown below in figure 4.

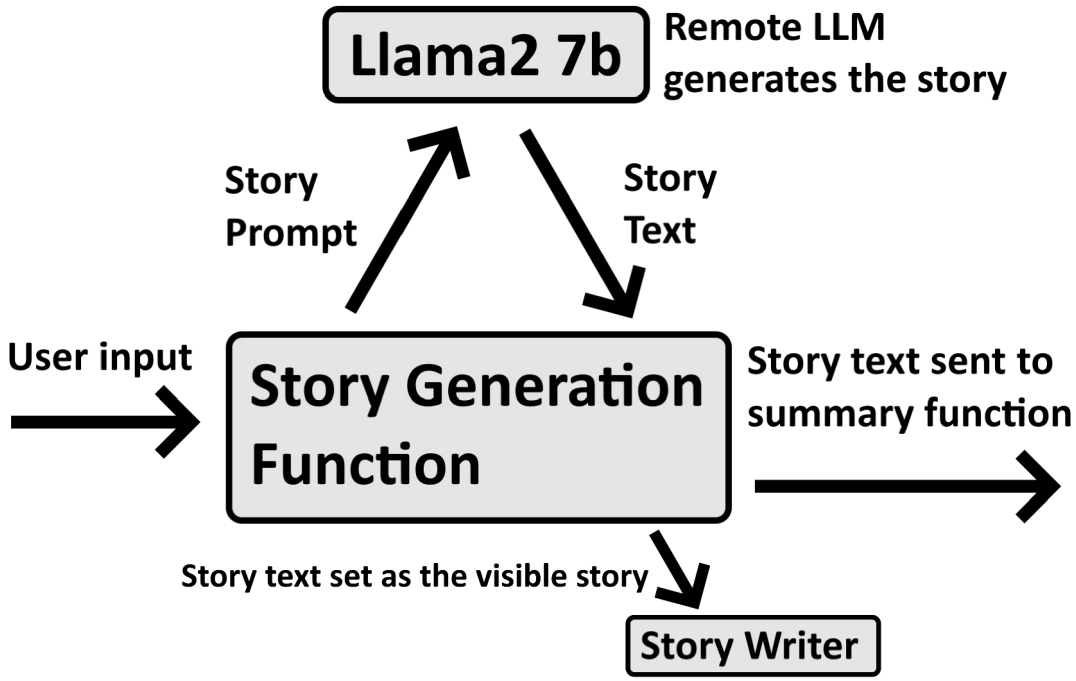

FIGURE 4. Story generation function work flow.

The summary function will use the story as body of the prompt and asks the LLM to describe a single photo based on the story. As the shortened story is only used once for the image generation and not shown anywhere, there is no need to set a token limit for it. Otherwise this summarization uses only the default settings for the prompt and the API call works the same as with the story generation. The summary function then saves the photo description into a variable and sends it to the image generation function. The workflow of the function is shown in figure 5.

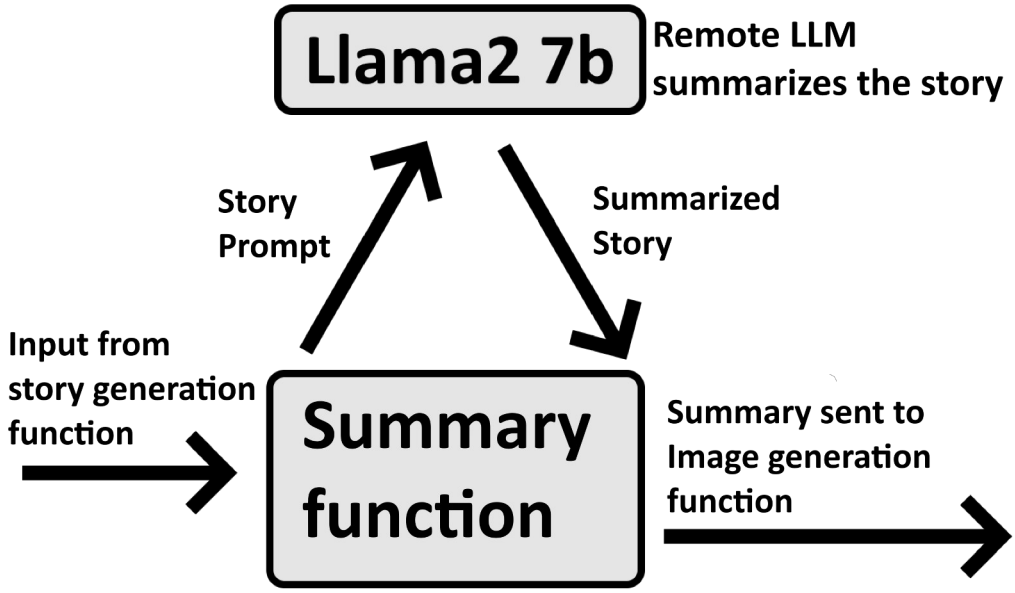

FIGURE 5. Summary function work flow.

6.2 Image Generation

Image generation function connects to StabilityAI through their gRPC API. The api key for StabilityAI is also hidden in the ".env" file and is required for the image generation to work.

Image generation function receives the summarized story from the summary function and uses it as the prompt text for the StabilityAI. The other parameters included in the prompt are seed, steps, cfg_scale, width, height, samples and sampler. All generated images use the seed 42 unless overwritten by the user through the user interface seed text box. Width and height are the size of the image in pixels, they are both set to 512 as this works well with the page layout of the story writer. Steps controls the number of inference steps taken, each inference step denoises the generated image increasing detail. Steps is set to 20 as for the purposes of this project that is more than enough. Cfg_scale controls how tightly the AI tries to stick to the prompt given, its left to default value of seven as that seemed to give most consistent results after testing. Samples is the number of images generated, the story needs only one image per page so its set to one. Sampler does the denoising of the image after generation, its left to default value for this project. After generating the image there is a chance that the image triggers StabilityAIs safety filters, if this happens, the function handles the error and defaults to old image.

After successful image generation, the image is saved into the root directory of the story writer and set as the current image shown. This image generation is the last step in the work cycle the story writer does when the story generation is ran as shown below in the figure 6.

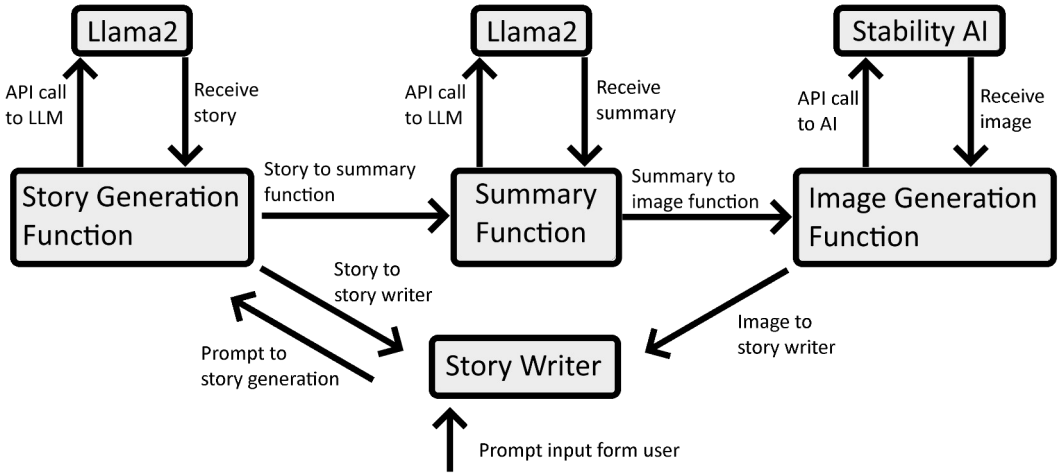

FIGURE 6. Full cycle of story generation from user input to image generation.

6.3 User Interface

The user interface was made with Tkinter python library. Using grid layout it was a relatively simple task to arrange the story text, image and the controls into roughly book like arrangement. At this stage text-to-speech was also added as an option for narrating the story out loud. Implementing pyttsx3 was straight forward as there was no need to use other libraries to play the narration. It turns the story string straight into voice and saves it into a variable.

Buttons for navigation was added and with them an option to generate more pages and the possibility to save the generated story into a folder. The navigation works by saving the whole story into a dictionary, from where the relevant text and image pairs are accessed by index and then placed as the shown pair. Saving will loop through the dictionary saving all the story text into a single txt file. Extra images are automatically saved into the folder as they are generated.

Sun-Valley-ttk theme was used to make the Tkinter theme more eye friendly. The theme gets set at the start and an option to toggle between dark and light mode was added.

6.4 Testing

Testing in the project was mainly done manually as unit testing. Each part was tested separately on their own before integrating into the main project. This made it easy to track errors and fix them. There were some problems that came up with the story AI, image AI and text-to-speech narration, but mostly the development went smoothly.

Llama2-7b which was used for the story text generation proved to be insistent in adding a short confirmation to the start of each generated story.

Stability-sdk which was the development kit for the image generation AI had not been updated to the last version of python, which caused quite a bit of headache before the correct python version was found.

GTTS was first used for the narration, it uses google translates text-to-speech to create a sound-file of the target text which was then saved into the root folder. This method needed a separate library to play the sound file. This caused file type errors and was in general slow. It was ultimately decided that it was too troublesome and the narration was switched to use pyttsx3.

7 RESULTS

The finished project works as intended. It is capable of creating a short story based on the input prompt form the user and create an image to accompany the generated story. Figure 7 shows how the finished project looks like after generating a story with the prompt “Bee”. The story is randomized, so every time the generate button is pressed there will be a new different story shown even with the same prompt words.

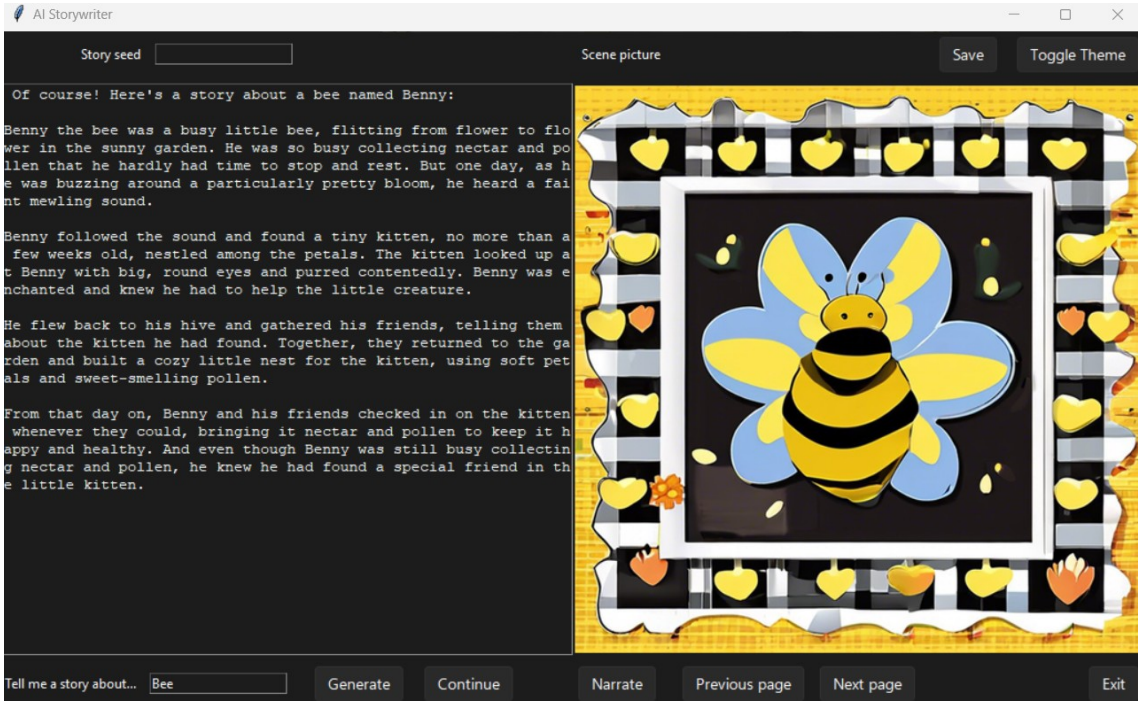

FIGURE 7. Image of the whole project with story and image generated with the word “Bee”.

User writes a prompt into the text input field and presses generate button which starts the story generation. Shown below in the figure 8 are the generation controls.

FIGURE 8. Prompt text field, generate and continue buttons.

The story writer will generate a story based on the prompt text. This takes a moment as the connection is made to the AIs and the story and story image are received. Figure 9 shows a story generated for the prompt “a round bee”.

```
 Of course, I'd be happy to help! Here's a story about a round b
ee:

Once upon a time, in a sunny meadow, there was a little round be
e named Buzz. Buzz was different from the other bees, as she was
 perfectly round in shape, with no edges or angles. She could ro
ll and spin with ease, and she loved to fly in circles around th
e flowers.

The other bees teased Buzz, calling her names like "Roundy" and
"Sphere." But Buzz didn't let their mean words bother her. She k
new she was special, and she liked being different.

One day, Buzz found a beautiful, bright yellow flower that she j
ust had to have. But when she tried to land on it, she rolled ri
ght off! The other bees laughed at her, saying she was too round
 to land properly.

But Buzz didn't give up. She kept rolling and spinning, trying t
o find a way to land on the flower. And eventually, she discover
ed that her round shape was actually a superpower! She could rol
l through the air so fast that she could fly faster than any of
the other bees.

From that day on, Buzz flew circles around the other bees, showi
ng them that being different wasn't a bad thing. And she always
remembered to never give up, no matter what challenges came her
way. The end.
```

FIGURE 9. An example story with prompt text "a round bee".

Once the story text has been generated an image will be created based on the summarized version of the story. They are then shown to the user at the same time. Figure 10 shows an example picture generated for a story about round bees.

FIGURE 10. An example image generated from the story with the prompt “a round bee”.

After the first page has been generated, user can create more pages using the continue button. User can browse the pages using the pages controls and narrate any page they want. Its possible to choose the story seed in case the user does not want to use randomized seed. Close up of these controls shown in the figure 11 below.

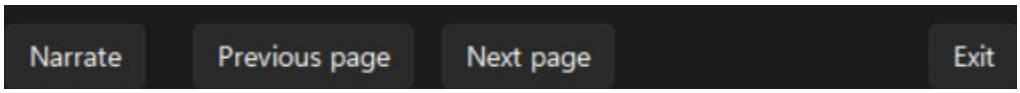

FIGURE 11. Narration and page controls.

Selecting specific seed for use for the story generation is possible and the option to save the generated story to a file. Theme can be toggled between light and dark. These controls are shown in the figure 12.

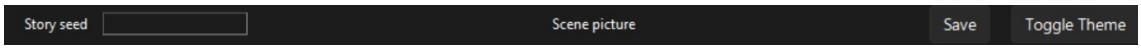

FIGURE 12. Top control bar with fields for random number generator seed, save and toggle theme buttons.

The stories that are generated vary widely in quality. Some are good or decent, but others have a hard time focusing on the initial story line and might change the subject of the story mid way or have slightly incoherent story structure. Sometimes the stories also get cut short when the LLM for some reason ignores the token limit in its instructions and hits the hard limit set in the parameters. As part of the response generated the LLM would add an introductory sentence acknowledging the user and confirming what the story will be about and sometimes it would also add an ending line hoping the user liked the generated story. There were attempts to stop the generation with prompt engineering and attempts to remove these parts from the final text string through code. But due to the randomized and unpredictable nature of these lines, it was hard to detect them correctly for deletion or stop the LLM from generating them. In the end, they were left in as it is.

The image generation manages to create images that most of the time are related to the story. However rarely it creates images that are only partially related to the story or only related to a small part of the overall story. A story about bees might have an image of a dog because in one sentence a dog was mentioned and the summarizing function elevated that sentence into more prominent role.

Continuing the story often holds the context only for a few pages. Minor details lost context after just a few pages and things like the protagonist being a fish might be lost and then in the later pages the protagonist might be a human or a dog.

From secondary goals, only the option to continue the story was completed in full and the option to narrate the story was partially completed. Other secondary goals were not completed. The option to narrate the story was completed without using AI and it is just generic text-to-speech instead of AI tuned voice. Plan to use local large language models for parts of the project was discarded after the initial research into large language models due to lack of memory and hardware required to smoothly operate them fast enough. While focusing on the other parts of the project the time passed faster than expected and the AI translation was discarded from the goal list. Overall the project was a success as all the primary goals were completed and the project functions as planned.

8 CONCLUSIONS

The main goal of this project was to demonstrate multiple AIs working together writing a short story. To showcase this, a program was created that used Large Language Models to generate a story and generative AI to create an image to accompany the story. This project has shown that even a generic LLM that has not been fine-tuned, can generate stories to some degree. A LLM that has been fine-tuned to understand and focus on story generation should be able to fare better.

Further development points for this project would be changing the story generation LLM to a larger more powerful model that has larger context window and has been fine tuned for story generation. This would enable the story writer to create more complex stories. The summarizing LLM should be fine tuned to create scenic summaries to improve the prompts sent to the image generation AI. The image generation AI could be trained on specific style of art to make it create more storybook like illustrations.

As a part of further study into cooperation between AIs, a project could be made where AI recognizes sound and speech, and turns it into compatible data for other AIs to caption and create images based on the sounds or speech. This could improve accessibility to some types of media for people with hearing difficulties. Another idea could be to make the AIs play a simulated game of logistics to form supply chains with goal of ensuring that they all have the parts they need to produce their products and move them further along the line while some of them would be competing for the same packets.

Linking up multiple AIs to work together is viable strategy depending on what is the intended use. Working this way can offload the work into multiple parts that can be worked on in tandem or parallel by multiple specialized AIs to improve the results. Improvements in memory and processing power will allow AIs to be employed in more ways. AI powered solutions can be packed into smaller setups while not compromising the speed required for comfortable use.

While outside the scope of this thesis work, some additional topics came up during the work process. One of such topics was about the data required for training different kinds of AIs. This data is readily available in the internet, however many users have not given their consent on using their created content for training AIs or are not even aware that such thing could happen. This is prevalent in AIs that specialize in image generation. Social media and image hosting sites where artists post their works are often used as sources for image data sets that are then used to train

AIs. Likewise, any chat logs, public messages or other posts in the internet can be used for AI training without the knowledge of the original authors.

Another was the ethicality of using image generating AIs as part of different kinds of projects. For free and hobby projects there would not be much issue, but for professional works where these AIs would take illustration jobs from actual artists and when these AIs are often trained for free on the artistic content posted by artists online. It becomes a much more problematic question.

While this project does not have practical use outside of its own intended use, parts of it could be modified for other projects. The code for Llama API calls could easily be modified for other uses and to use different Llama models. The UI itself could be repurposed to serve as general LLM interface with minimal effort. The whole project also serves as one example of how different kinds of artificial intelligences could work together.

LLMs and AIs are still showing a lot of potential and more ways to utilize them are constantly found as more complex models become available for people to use.

REFERENCES

1. Nvidia 2023. Artificial Intelligence. Search date 8.11.2023. https://www.nvidia.com/en-us/glossary/data-science/artificial-intelligence/
2. Duggal, Nikita 2024. Advantages and Disadvantages of Artificial Intelligence [AI]. Search date 30.3.2024. https://www.simplilearn.com/advantages-and-disadvantages-of-artificial-intelligence-article
3. Turing, Alan Mathison 1950. Computing Machinery and Intelligence. Search date 10.3.2024. https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf
4. Roberts, Jacob 2016. Thinking Machines: The Search for Artificial Intelligence. Search date 10.3.2024. https://www.sciencehistory.org/stories/magazine/thinking-machines-the-search-for-artificial-intelligence/
5. Lutkevich, Ben 2022. AI Winter. Search date 10.3.2024. https://www.techtarget.com/searchenterpriseai/definition/AI-winter
6. IBM. Deep Blue. Search date 10.3.2024. https://www.ibm.com/history/deep-blue
7. Council of Europe 2024. History of Artificial Intelligence. Search date 11.3.2024. https://www.coe.int/en/web/artificial-intelligence/history-of-ai
8. SAS 2024. Generative AI, What it is and why it matters. Search date 12.3.2024. https://www.sas.com/en_us/insights/analytics/generative-ai.html
9. Council, Greg 29.3.2018. Machine Learning vs Expert Systems AI. Search date 1.5.2024. https://www.parascript.com/blog/machine-learning-ai-vs-expert-systems-ai/
10. Kanade, Vijay 25.3.2022. Narrow AI vs. General AI vs. Super AI: Key Comparisons. Search date 9.3.2024. https://www.spiceworks.com/tech/artificial-intelligence/articles/narrow-general-super-ai-difference/
11. Hintze, Arend 14.11.2016. Understanding the Four Types of Artificial Intelligence. Search date 8.3.2024. https://www.govtech.com/computing/understanding-the-four-types-of-artificial-intelligence.html
12. Butlin, Patrick & Long, Robert & Elmoznino, Eric & Bengio, Yoshua & Birch, Jonathan & Constant, Axel & Deane, George & Fleming, Stephen & Frith, Chris & Ji, Xi & Kanai, Ryota & Klein, Colin & Lindsay, Grace & Michel, Mathias & Mudrik, Liad & Peters, Megan & Schwitzgebel, Eric & Simon, Jonathan & VanRullen Rufin 22.8.2023. Consciousness in Artificial Intelligence: Insights from the Science of Consciousness. Search date 5.5.2024. https://arxiv.org/pdf/2308.08708

13. IBM. What are large language models(LLMs)? Search date 29.4.2024. https://www.ibm.com/topics/large-language-models
14. Selvaganapathy, Chidambaram 15.7.2023. Demystifying Training Parameters in LLM - Quality Impact and Sizing Considerations. Search date 31.3.2023.. https://www.newsletter.theaidiscovery.com/p/demystifying-training-parameters
15. Humor, Michael 10.9.2023. Understanding “tokens” and tokenization in large language models. Search date 11.4.2024. https://blog.devgenius.io/understanding-tokens-and-tokenization-in-large-language-models-1058cd24b944
16. Sennrich, Rico & Haddow, Barry & Birch, Alexandra 10.6.2016. Neural Machine Translation of Rare Words with Subword Unit. Search date 16.4.2024. https://arxiv.org/pdf/1508.07909.pdf
17. OpenAI. Tokenizer. Search date 12.4.2024. https://platform.openai.com/tokenizer
18. Kerner, Jonathan 4.2.2021. SentencePiece Tokenizer Demystified. Search date 16.4.2024. https://towardsdatascience.com/sentencepiece-tokenizer-demystified-d0a3aac19b15
19. Kudo, Taku & Richardson, John 19.8.2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. Search date 16.4.2024. https://arxiv.org/pdf/1808.06226.pdf
20. Yang, Jinbiao 1.3.2024. Rethinking Tokenization: Crafting Better Tokenizers for Large Language Models. Search date 15.4.2024. https://arxiv.org/abs/2403.00417
21. Talamadupula, Kartik 21.12.2023. Guide to Context in LLMs. Search date 18.4.2024. https://symbl.ai/developers/blog/guide-to-context-in-llms/
22. Xiao, Guangxuan & Tian, Yandong & Chen, Beidi & Han, Song & Lewis, Mike 7.4.2024. Efficient Streaming Language Models with Attention Sinks. Search date 20.4.2024. https://arxiv.org/pdf/2309.17453.pdf
23. Afifi-Sabet, Keumars 23.2.2024. AI chatbots need to be much better at remembering things. Have Scientists just cracked their terrible memory problem? Search date 27.3.2024. https://www.livescience.com/technology/artificial-intelligence/ai-chatbots-chatgpt-bad-at-remembering-things-have-scientists-just-cracked-their-terrible-memory-problem
24. Huggin Face. LLM prompting guide. Search date 21.4.2024. https://huggingface.co/docs/transformers/main/en/tasks/prompting
25. IBM. What is prompt engineering? Search date 21.4.2024. https://www.ibm.com/topics/prompt-engineering
26. Amatriain, Xavier 7.5.2024. Prompt Design and Engineering: Introduction and Advanced Methods. Search date 23.4.2024. https://arxiv.org/pdf/2401.14423.pdf

27. Sahoo, Pranab & Sing, Ayush Kumar & Saha, Sriparna & Jain, Vinija & Mondal, Samrat & Chadha, Aman 5.2.2024. Search date 23.4.2024. https://arxiv.org/pdf/2402.07927.pdf
28. Yin, Shukang & Fu, Chaoyou & Zhao, Sirui & Li, Ke & Sun, Xing & Xu, Tong & Chen, Enhong 1.4.2024. Search date 28.4.2024. https://arxiv.org/pdf/2306.13549
29. OpenAI 30.11.2022. Introducing ChatGPT. Search date 3.5.2024. https://openai.com/index/chatgpt
30. Dentato, Remo 11.2.2024. Choose Your Own Coding Assistant. Search date 3.5.2024. https://dev.to/rdentato/choose-your-own-coding-assistant-11gi
31. Cushman, Jack 29.11.2023. LLMs are universal translators: on building my own translation tools for a foreign language conference. Search date 4.5.2024 https://lil.law.harvard.edu/blog/2023/11/29/llms-are-universal-translators/#how-i-built-my-own-personal-translation-tools
32. Databricks 2024. LLMs for Customer Service and Support. Search date 4.5.2024. https://www.databricks.com/solutions/accelerators/llms-customer-service-and-support
33. AssemblyAI 2024. Speech Understanding. Search date 4.5.2024. https://www.assemblyai.com/discover/products/speech-understanding?ref=assemblyai.com
34. Korol, Alexei 14.8.2024. Revolutionizing Database Management with LLMs. DB-GPT: A Comprehensive Exploration. Search date 4.5.2025. https://medium.com/@korolalexei/revolutionizing-database-management-with-llms-db-gpt-a-comprehensive-exploration-3ceb677a17c8
35. BlueDot 2024. How BlueDot is using LLMs to detect outbreaks faster. Search date 4.5.2024. https://bluedot.global/using-llms-to-detect-outbreaks-faster/
36. Levitt, Kevin 13.12.2023. How is AI Used in Fraud Detection. Search date 4.5.2024. https://blogs.nvidia.com/blog/ai-fraud-detection-rapids-triton-tensorrt-nemo/
37. Parker, Henry 22.6.2023. How Duolingo uses AI to create lessons faster. Search date 4.5.2024. https://blog.duolingo.com/large-language-model-duolingo-lessons/
38. Gokhale, Praachee & Singh Suag, Anmol & Mallela, Manyam 14.12.2023. Amazon Titan Embeddings for enhanced content recommendations to power 1:1 personalization. Search date 4.5.2024. https://aws.amazon.com/blogs/industries/amazon-titan-embeddings-for-enhanced-content-recommendations-to-power-11-personalization/
39. Chu, Zhibo & Ni, Shiwen & Wang, Zichong & Feng, Xi & Li, Chengming & Hu, Xiping & Xu, Ruifeng & Yang, Min & Zhang, Wenbin 10.2.2024. History, Development, and Principles of Large Language Models – An Introduction Survey. Search date 5.5.2024. https://arxiv.org/pdf/2402.06853

40. Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz 2.8.2023. Attention Is All You Need. Search date 5.5.2024. https://arxiv.org/pdf/1706.03762