University of Pisa
School of Engineering

Business and Project Management

Social Media Content Production

Students
Martina Burgisi
Matteo Razzai

`https://github.com/matteorazzai1/SocialMediaContentProduction/`

September 10, 2024

Contents

1 Introduction **2**

2 Design and implementation **3**
2.1 Text and image generation models 3
2.1.1 Text generation model 3
2.1.2 Image generation model 5
2.2 Graphical User Interface (GUI) 7
2.2.1 Advanced settings 8

3 Prompt Engineering **9**
3.1 Image Generation 9
3.1.1 Examples 10
3.2 Text Generation 10
3.2.1 Few-Shot Prompting 10
3.2.2 Chain-of-Thought Prompting 11
3.2.3 Example 11

4 Running the Flask Application **13**
4.1 Guide 13
4.1.1 Requirements 13
4.1.2 Project structure 13
4.1.3 How to run the project 14

5 Results **15**
5.1 Generated caption 15
5.2 Generated images 18
5.3 Examples of some generated posts 20
5.3.1 Example 1 20
5.3.2 Example 2 22
5.4 Midjourney test 25
5.4.1 Conclusions 26

Chapter 1

Introduction

This project explores the integration of advanced artificial intelligence techniques for the automated creation of social media contents. It combines the usage of Groq for the text generation and StableDiffusion for the image.

The main goal was to develop a tool able to generate a social media post composed of a relevant caption and a coherent associated image. Part of the work was about using Retrieval-Augmented Generation (RAG) to improve the quality of the text generated; we also focused on developing a similar technique for image generation, that is an example-based optimization to improve the images generated by StableDiffusion.

Another important aspect of the project was the "prompt engineering" part: the creation and optimization of prompts to achieve better results in terms of the user intention. To obtain this, we studied some papers and applied the techniques learned. We implemented for this purpose an "Advanced Settings" section within our web interface. This section allows users to customize various details related to text and image generation using specific input fields.

We wrote this documentation with the aim to provide an overview of our work, deepening the technologies used, the optimization adopted and the solutions implemented to ensure high-quality, automated social media content generation.

Chapter 2

Design and implementation

In this section, we will focus to the architecture of our project, covering the models we used and the main processes involved in developing the final version of our application.

2.1 Text and image generation models

For our application we decided to use Python, as a programming language due to its robust capabilities, that are resulted very useful for handling API calls and to process data. In order to produce a post, we needed a caption and a related image, for this reason we used two models, one capable to produce text and the other one well-suited to the generation of images.

2.1.1 Text generation model

For the first one, we used LLaMa 70B, developed by Meta and hosted by Groq, an American firm born in 2016. Groq allows us to use its models via API calls.

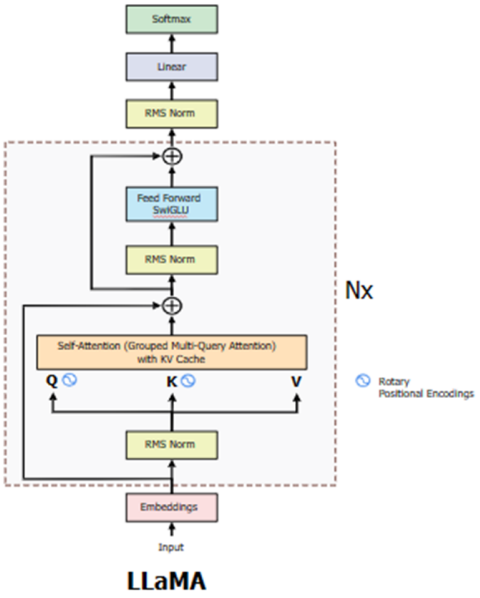


Figure 2.1: LLaMa architecture

We use LLaMa for the generation of the caption of the post, we used a prompt specifically designed for the user needs, like explained in the Prompt Engineering chapter.

Retrieval-Augmented Generation

Another key component of the text generation is the RAG mechanism. RAG is an AI framework for improving the quality of LLM-generated responses by giving to the model external sources of knowledge. In our application we used this technique to give as input to the model, some examples of post, written by celebrities and influencers. We thought that to create a more appealing post, trying to use the knowledge and the capabilities of people who work with social. In order to do that we use the following dataset:

`https://github.com/minimaxir/interactive-facebook-reactions/tree/master/data`

The next operation was to produce the embedding for all the posts and save them on a database. To produce embedding we used **all-MiniLM-L6-v2**, an embedding model of **sentence-transformers**, a Python module specialized in the construction and manipulation of embedding. The embeddings are a numerical representation of information, and we needed that in order to find the most similar post to our caption, looking into the database before mentioned.

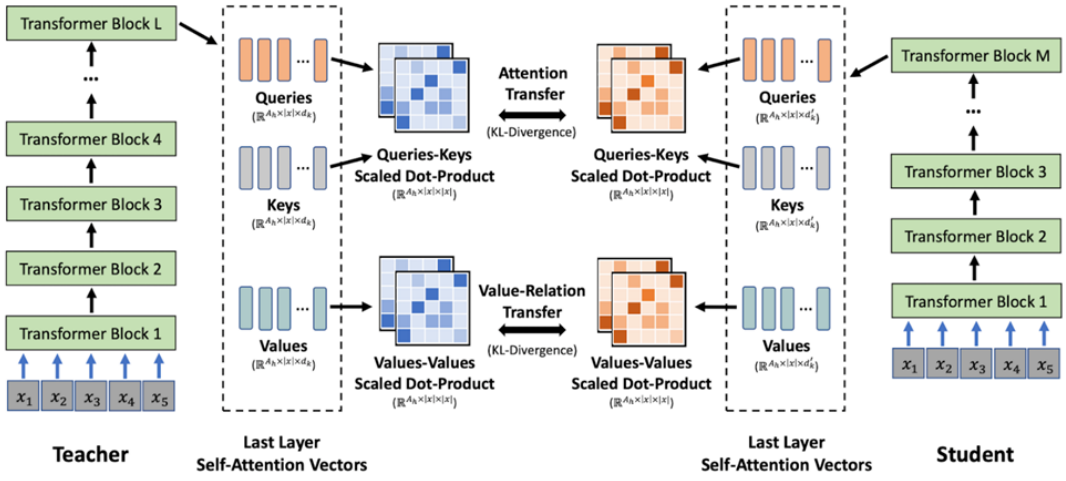


Figure 2.2: MiniLM architecture

Those embeddings have to be saved in a database, for this goal we used **Supabase**, which allows us to save large vectors, like the embeddings quoted above. For each record of the dataset mentioned before, we saved: the index, the embeddings and the post. In this way, during the post creation phase, we could find the 3 most similar posts using the cosine similarity between the embeddings, and then add them to the prompt.

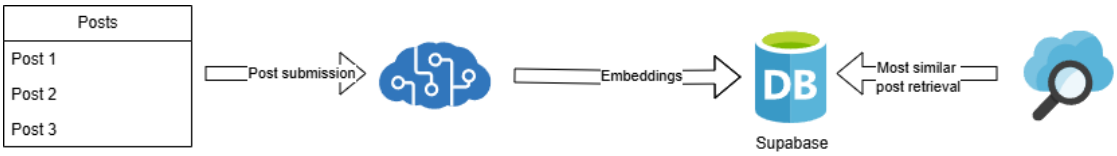


Figure 2.3: Interaction with the vector database

2.1.2 Image generation model

For the image generation, we used StableDiffusion, a text-to-image latent diffusion model capable of generating photo-realistic images given any text input, in particular we used the model *"CompVis/stable-diffusion-v1-4"* [1].

Simil-RAG for image generation

One of the primary goals of this project is to improve the performance of the image generator. Unlike text generation, where techniques like RAG (Retrieval-Augmented Generation) are available, there isn't an equivalent technique for image generation. Therefore, we have developed a different pipeline that similarly provides the model with examples

to facilitate its task.

The quoted pipeline is visible in the following image:

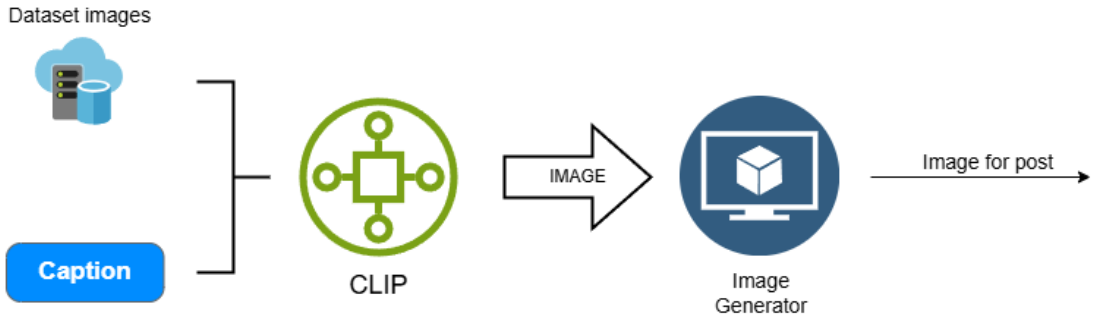


Figure 2.4: Pipeline of the simil-RAG mechanism used in this project

The new component in the previous pipeline is CLIP (Contrastive Language-Image Pretraining). CLIP is a deep learning model developed by OpenAI, it combines vision and language by learning to associate images and text in a way that allows it to perform a wide range of tasks, including image classification and natural language understanding, without task-specific training.

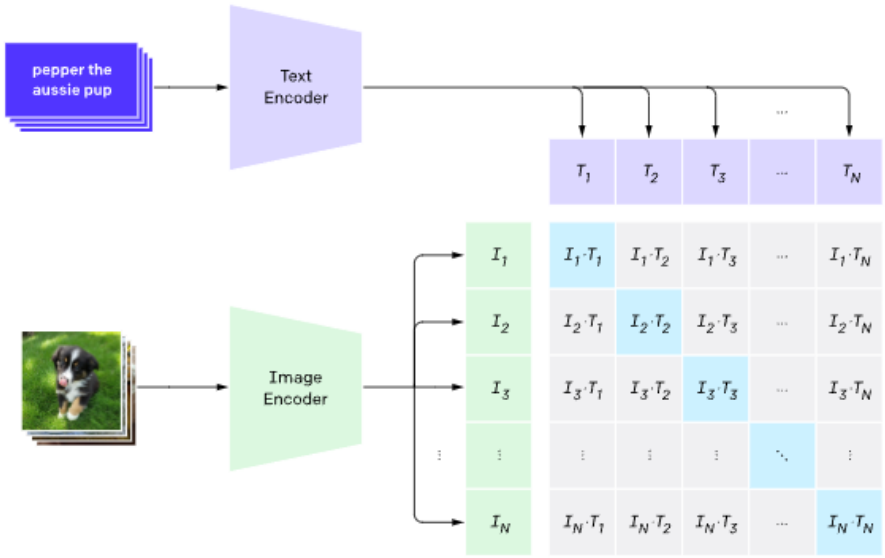


Figure 2.5: CLIP general structure

CLIP is composed of two neural networks: an image encoder and a text encoder. These two encoders work together to enable the model's cross-modal understanding of images and text.

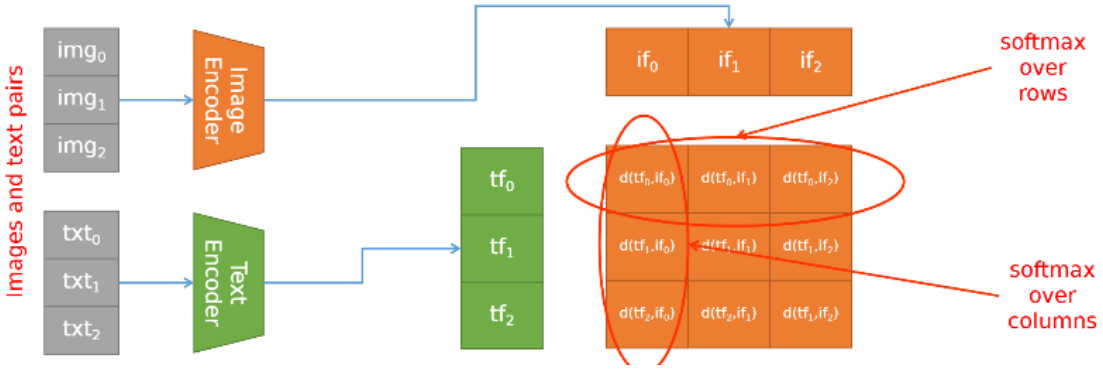


Figure 2.6: CLIP operation to compare images and text

We used the above CLIP schema, but throwing a single text to the text encoder, so we will have only the first row in the orange matrix on the right of the image, in which we will have the similarity between that text and all the images in the dataset. The *Flickr 8k* dataset is accessible at the following link:

`https://kaggle.com/datasets/adityajn105/flickr8k`

In this way we are then capable to retrieve the most similar image in the dataset, and use that as initial image from which the generation of a new image starts, visible in the Figure 2.4.

2.2 Graphical User Interface (GUI)

The interface of our project is designed as a web page built using HTML, CSS and JavaScript. When users open the page via browser, they immediately see a simple panel containing two essential input fields:

- **Company Name**, where users can enter the name of their company;
- **Main Field**, that allows users to specify the main topic or content they wish to focus on or the field of interest of the company.

They are both mandatory fields. The idea is to make the user write only the essential information to obtain a result.

At the bottom of this basic form is a "**Submit**" button. Clicking this button triggers the processing of the user's input. Initially, a "*Processing*" message appears and remains the time the request is handled.

2.2.1 Advanced settings

In addition to the core fields, users have the option to click on the "Advanced Settings" button. This makes the interface show additional optional fields. These advanced settings are based on detailed prompt engineering articles in literature, which are discussed later in Chapter 3. This possibility allows users to improve their inputs with more specific details, which can have an important impact on the quality of the generated content.

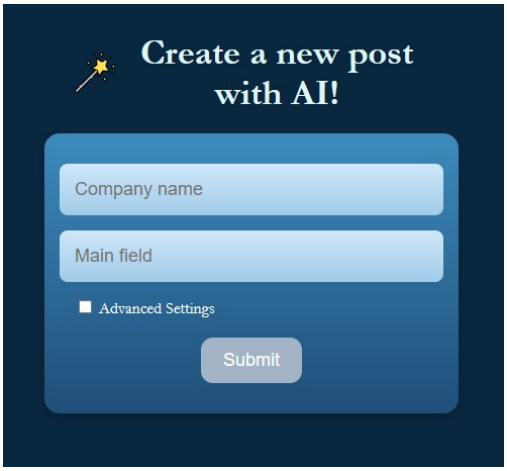


Figure 2.7: Input panel with only mandatory fields showing.

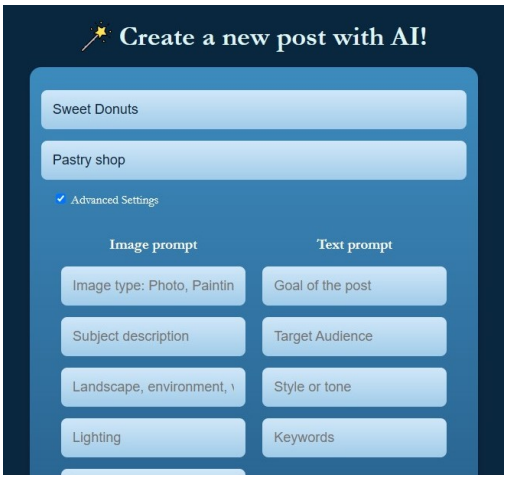


Figure 2.8: Input panel with some of the new input fields designed using prompt engineering, showing after clicking "Advanced Settings" checkbox.

Chapter 3

Prompt Engineering

3.1 Image Generation

As mentioned, the Python library used to implement the image generation logic is StableDiffusion. To get good results with StableDiffusion, as for every AI image generation model, it's important to write the prompt with detail and making explicit the generating goal.

As specified in the *Stable Diffusion Prompt Book* [2], a useful approach is to start with a series of questions that help define the desired results. Following a basic list of possible input fields, obtained from the study of the questions previously mentioned:

Image type

- **Photo**: Use terms like "photograph," "realistic image," or "high-resolution photo."
- **Painting**: Consider terms like "oil painting," "digital art," or "watercolor illustration."

Subject

- **Person**: Mention details like age, gender, ethnicity, and clothing.
- **Animal**: Specify the type of animal, its pose, and any distinguishing features.
- **Landscape**: Describe the environment, whether it's a cityscape, forest, ocean view, etc.

Specific Details

- **Lighting**: The type of lighting (e.g., "soft light," "neon glow," "dramatic shadows").
- **Color Scheme**: Color palette, such as "vibrant," "pastel," or "monochrome."
- **Point of View**: Angle or perspective (e.g., "overhead shot," "side view").

Art Style The image can mimic a particular art style or genre:

- **Art Style**: Options could be "3D render," "Studio Ghibli," "comic book," "vintage poster," etc.

Photo Type If it's a photo, specify the type:

- **Photo Type**: Mention if it's a "macro shot," "wide-angle," "telephoto," etc.

This list is not exhaustive but it is a useful starting point for prompt engineering. The more specific the prompt, the more the model can generate an accurate image.

3.1.1 Examples

Figure 3.1: Image generated with a short and simple prompt.

Figure 3.2: Image generated using prompt engineering.

Figure 3.1 was generated with the simple prompt: "Cat in the mountain", while figure 3.2 was created using a more detailed prompt: "Painting of a siamese cat running in the mountain during the sunset. I see him in front. Wide-angle." This shows the advantages of an adequate prompt engineering, with specific details. The precise description scene is helpful for the model to focus on key elements (such as lighting, perspective or color scheme) to obtain a more contextually accurate visual output.

3.2 Text Generation

To improve the quality of the generated post caption, two prompt engineering techniques were used: **Few-Shot Prompting** and **Chain-of-Thought Prompting** [4].

3.2.1 Few-Shot Prompting

Few-Shot Prompting is a technique that exploits input-output examples to help the AI better understand the required task and desired goal. The idea is to make the model learn from examples with the aim to produce high-quality output in similar situations.

Implementation in the Web App: This technique's application is perfectly integrated in the usage of the RAG mechanism, as deeply explained in the Chapter 2. The intention is to enrich the prompt with explicative example to positively bias the output result.

3.2.2 Chain-of-Thought Prompting

Another prompt engineering technique we studied is The Chain-of-Thought Prompting, which guides the language model through a sequence of logical thoughts. This technique is another method to generate coherent and detailed outputs, using step-by-step reasoning.

Implementation in the Web App: When the user need to generate a complex post or a very specific one, the application can make use of the Chain-of-Thought Prompting to improve the creation of the text using logical steps.

We design the application with a sequence of input fields that the user can use to specify the objectives, keywords, tone, and detail about the needed text. To apply the above-mentioned prompt engineering technique we designed 4 more fields:

- Goal of the Post
- Target Audience
- Style and Tone
- Keywords

These inputs can guide the model to generate an appropiate caption, such as a chain-of-thought entire prompt.

3.2.3 Example

Here is an example of a prompt constructed using the Few-Shot Prompting and the Chain-of-Thought Prompting techniques:

```
Create a social media post for a company.
The goal of the post is: To promote a new product.
The target audience is: Young professionals interested in technology.
The tone of the post should be: Enthusiastic and modern.
Include the following messages or keywords: Innovation, efficiency,
sustainability.

Post given in input as an example:
Discover the future of innovation with our new product! Designed for young
professionals, it combines efficiency and sustainability to improve your
daily work. Don't wait, join the technology revolution today!
```

```
Resulted caption:
Feeling overwhelmed by your daily work? Our new productivity tool is designed
to help you better manage time, improve efficiency and reduce stress.
Try it today and see how it can transform your day!
```

Chapter 4

Running the Flask Application

4.1 Guide

Flask is a Python framework for web application development. We chose Flask because it allows us to develop the project using our code editor but making the page accessible directly from a browser.

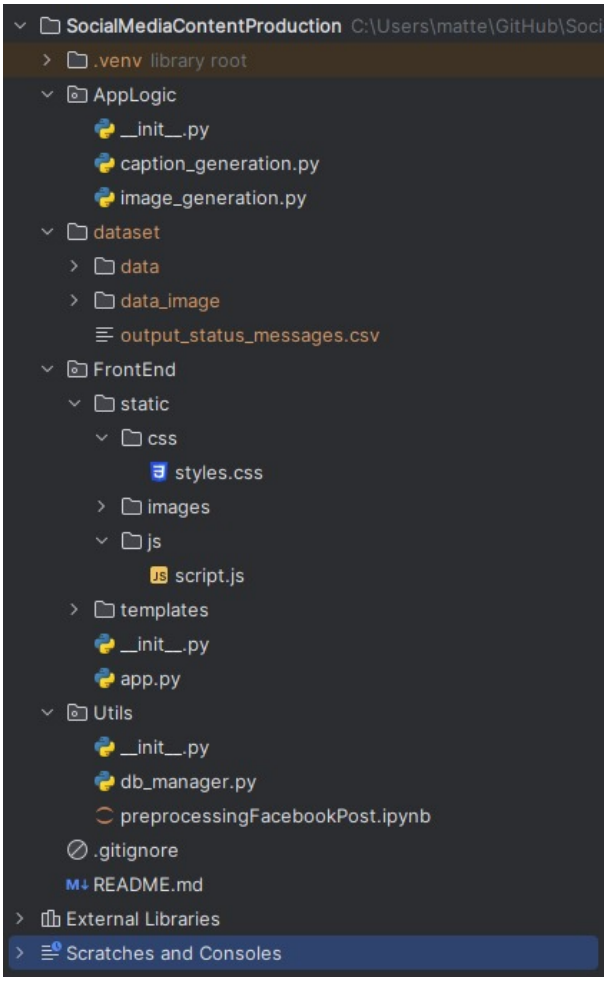
4.1.1 Requirements

Before running the project, make sure you have the following installed on your system:

- Python (version 3.6 or higher)
- pip (Python package installer)
- flask (as python library)
- other libraries included in the Python files.

4.1.2 Project structure

The project structure is the following:



4.1.3 How to run the project

To run the Flask application, you can use two methods:

Method 1

```
python app.py
```

Method 2

```
export FLASK_APP=app.py  # Unix/Linux/macOS
set FLASK_APP=app.py     # Windows
flask run
```

Both methods will run the application and make it accessible from the address http://127.0.0.1:5000.

Chapter 5

Results

In this chapter we approach the description of results obtained during the project, in particular comparing the results with and without RAG technique, both for what concern the caption production and the image generation.

5.1 Generated caption

We proceed here to analyze the results obtained after prompt engineering and after the application of a RAG technique in order to give, with the prompt, some useful example. In order to choose the prompt that would most efficiently return an eye-catching caption that well represented the company's goal, we performed multiple tests; below are some of them (always without advanced settings):

1. First attempt: resulted in a bulleted point list
   `"Can you suggest me a caption for a post related to" field_name "for my firm called" firm_name "?"`
2. Second attempt: resulted in a response announcing the caption, and not directly the caption
   `"Can you suggest me a caption for a post related to" field_name "for my firm called" firm_name "? But not a list, a single caption that presents the field and the firm in the best way possible"`
3. Last attempt before adding RAG: resulted in a well-formulated post
   `"Can you suggest me a caption for a post related to" field_name "for my firm called" firm_name "? But not a list, a single caption that presents the field and the firm in the best way possible and I don't want 'Here is your caption' and then the caption, I want directly the caption in response"`

After the application of the RAG technique and the possibility of using a more advanced setting, the prompt is generated in the function that generate the caption:

```python
def generate_caption(firm_name, field_name, text_prompt):
    goal = text_prompt.get('goal', '')
    target = text_prompt.get('target', '')
    style = text_prompt.get('style', '')
    keywords = text_prompt.get('keywords', '')

    prompt_parts = [
        f"Can you suggest me a caption for a post related to" + field_name + " for my firm called" + firm_name + "?"]

    if goal:
        prompt_parts.append(f"The goal of the post is: {goal}.")
    if target:
        prompt_parts.append(f"The target audience is: {target}.")
    if style:
        prompt_parts.append(f"The tone of the post should be: {style}.")
    if keywords:
        prompt_parts.append(f"Include the following messages or keywords: {keywords}.")

    text_prompt_str = " ".join(prompt_parts)+("But I don't want a list, I don't want 'Here is your caption' and then "
                                              "the caption, I want directly the caption in response from you."
                                              "And these are some post example:") + retrieve_K_most_similar_post(firm_name, field_name)

    caption = perform_request(text_prompt_str)

    return caption
```

Figure 5.1: Function used to generate the caption given the prompt input.

In the following example of some tests, we can see the improvement of using a RAG technique:

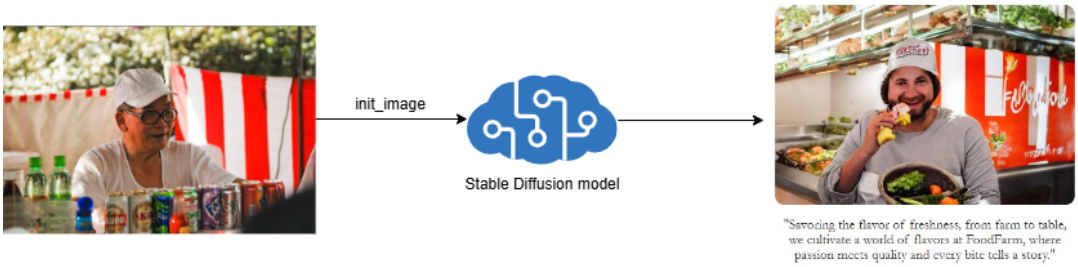
Input	Result Without RAG	Result With RAG
`firm`: TrekkingDream `field`: Trekking in the mountain	"Step into the unknown and let the mountains leave you breathless! Join TrekkingDream on an unforgettable journey to conquer the peaks and discover the thrill of adventure #TrekkingDream #MountainLove #AdventureAwaits"	When the mountain calls, we answer, despite the rugged trails and unpredictable weather, for in its majesty, we find our own strength. Join TrekkingDream to experience the thrill of conquering the peaks and discovering the uncharted territories within yourself.
`firm`: Ferrero Roche `field`: Fusion with Nestlè	"Sweet innovation alert! We're thrilled to announce our exciting fusion with Nestlè! As Ferrero Rocher, we're passionate about crafting unparalleled chocolate experiences, and this partnership is the perfect blend of our rich heritage and Nestlè's expertise. Stay tuned for the most indulgent creations yet to come! #FerreroRocher #Nestle #ChocolateInnovation"	Sweet harmony takes on a new meaning! Ferrero Rocher and Nestle are joining forces in a fusion that will change the game - stay tuned for the sweetest collaboration of the year!
`firm`: SkisLife `field`: Influencer Mario Rossi try our ski suit	"Getting ready to shred the gnar with @MarioRossi in our newest SkisLife ski suit! The king of the slopes has given us his royal approval, and we couldn't be more stoked! #SkisLife #SkiSuit #MarioRossiApproved"	"Mario Rossi takes on the slopes in style with SkisLife, the ultimate ski suit for those who live for the thrill of the ride"

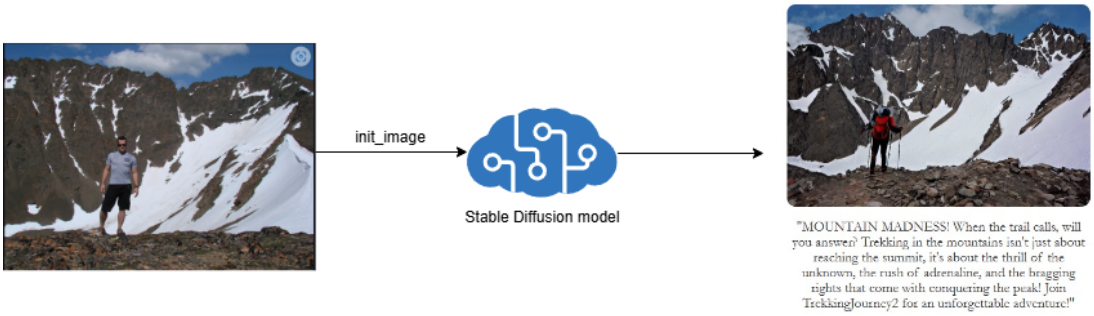
Table 5.1: Comparison of Results Without and With RAG

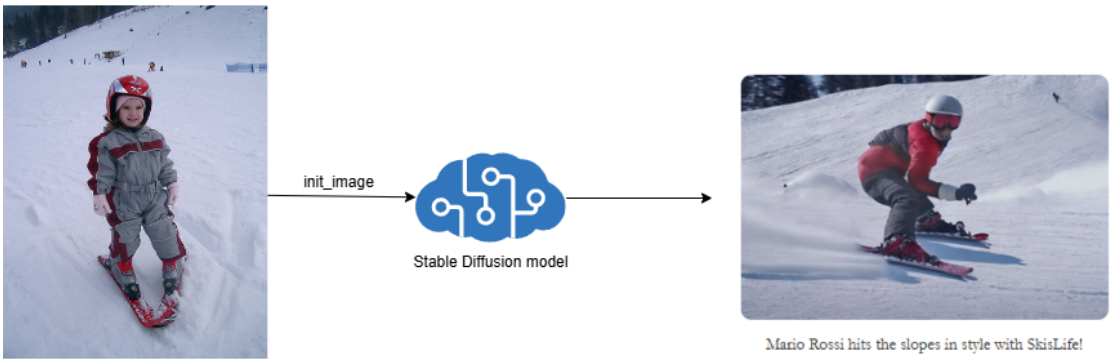
In our project, the use of RAG has significantly improved caption generation. This technique enables us to create shorter captions but more impactful and captivating. The improvement is about focus on essential information using a compact format without compromising content quality or the ability to capture the user's attention.

5.2 Generated images

In this section we want to show the advantages of using a simil-RAG technique for the generation of images. In the following images we can see some examples:







We can easily see the strong connection between the starting image and the generated one. The starting image has been chosen from the dataset using the technique mentioned in the section 2.1.2.
Before adding these technique the generation of the images, starting only from the caption, could take us to an image with more error in the production, like in the following example:

Figure 5.2: Image generated with the only usage of the caption, less realistic.

5.3 Examples of some generated posts

5.3.1 Example 1

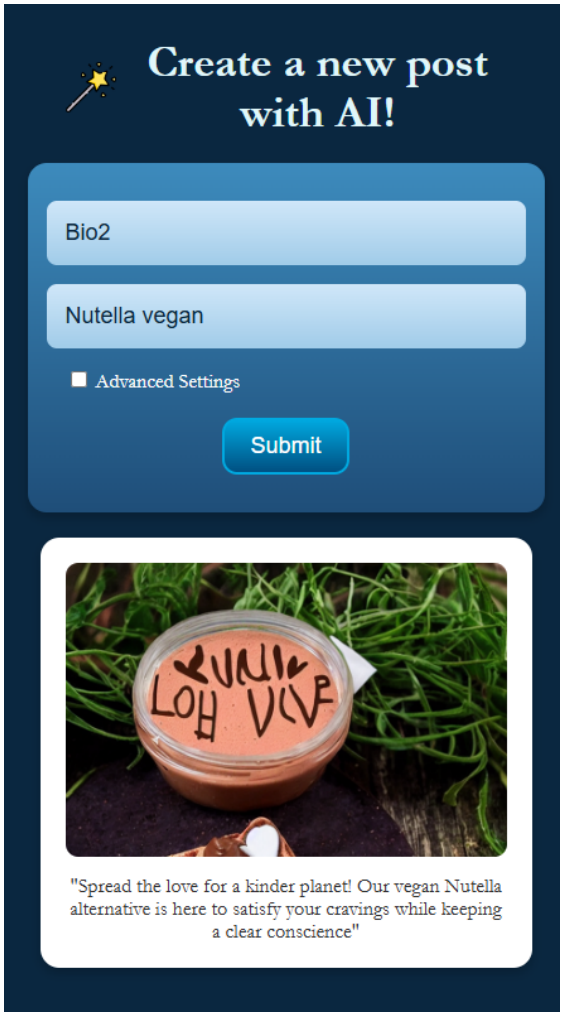


Figure 5.3: Image generated with a short and simple prompt about vegan Nutella.

Figure 5.4: Whole prompt with advanced settings partially filled to obtain post about vegan Nutella.

Figure 5.5: Image generated about vegan Nutella with a more detailed prompt using prompt engineering. The caption and image results now as more realistic and detailed.

5.3.2 Example 2

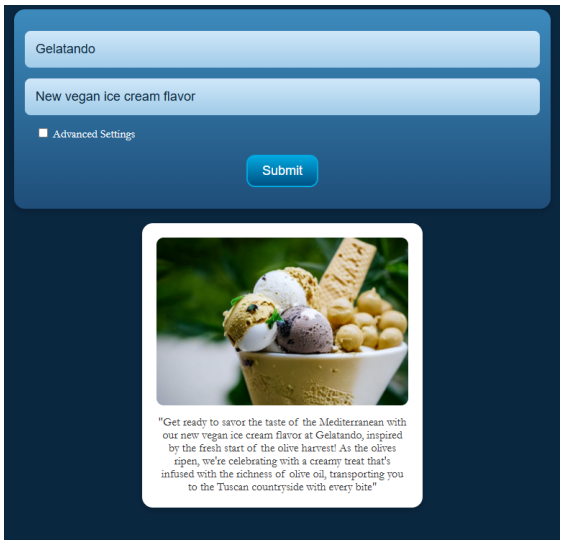


Figure 5.6: Image generated about vegan ice cream with a simple prompt.

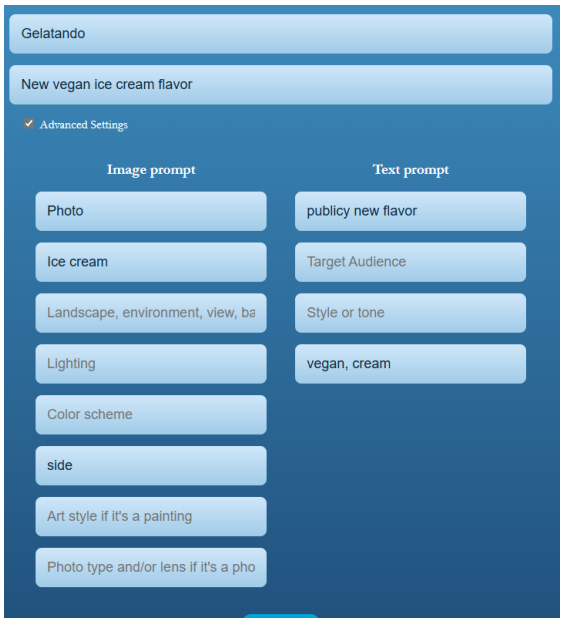


Figure 5.7: Whole prompt with advanced settings partially filled to obtain a post about a new flavor of ice cream.

Figure 5.8: Whole prompt with advanced settings partially filled to obtain post about ice cream.

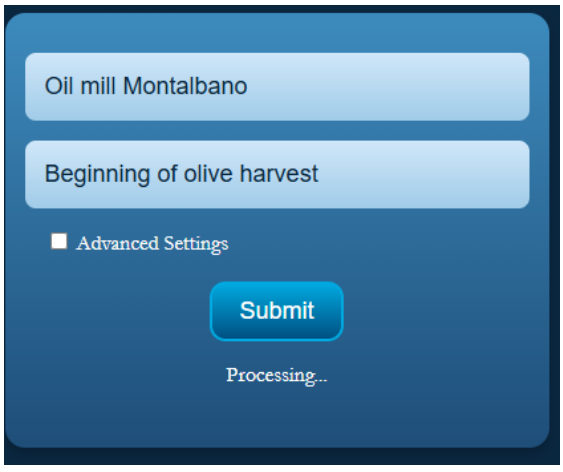


Figure 5.9: Simple and short prompt.

"Harvest vibes are in the air! Our olive harvest has officially begun at Oil Mill Montalbano! Get ready to taste the freshness and reap the health benefits of our new, premium olive oil - perfect for fueling your fast-paced lifestyle!"

Figure 5.10: Image and caption generated from the previous prompt.

5.4 Midjourney test

We conducted an in-depth analysis to evaluate new image generator models, considering also Vincos Blog as a reference to search feedback about newest versions possible. On this site we came across the following article: "*La prova di Flux 1.0 alternativa open per la generazione di immagini*" [5] in which the author talks about the newest image generator model Flux 1.0 in comparison with the best one on the market, Midjourney. The first model is still in Beta version and the APIs are not available, while the Midjourney model allow us to test the image generation but with limited token and consequently a limited number of possible tests.

```python
from PIL import Image
import requests
from io import BytesIO

endpoint = "https://api.ttapi.io/midjourney/v1/imagine"

headers = {
"TT-API-KEY": "7b0f5d41-31be-c35a-5330-86ee23430070"
}

data = {
    "prompt": "New vegan ice cream flavor",
    "model": "fast",
    "hookUrl": "",
    "getUImages":"true"
}

response = requests.post(endpoint, headers=headers, json=data)

jobId=response.json()['data']['jobId']

endpoint = "https://api.ttapi.io/midjourney/v1/fetch"

data = {
    "jobId": jobId,
}

response = requests.post(endpoint, headers=headers, json=data)
url=response.json()['data']['images'][0]['url']

response = requests.get(url)
img = Image.open(BytesIO(response.content))

img.show()
Executed at 2024.09.10 12:02:24 in 322ms
```

Figure 5.11: Code needed to make an API request to Midjourney to generate an image.

Figure 5.12: The image generated using Midjourney API with the previous code (the prompt is specified in the code screen itself).

In future works it may be interesting to expand the project building the social media post generator tool with the Midjourney model for the images part, provided there is adequate funding to cover the associated costs.

5.4.1 Conclusions

The results we obtain with our project allow us to vision the potential of text and image generating model, combined with the aim to create a social media post. Our goal was to demonstrate how these types of tools can give a huge help especially to companies that sell products or services. In fact, the companies can exploit the automated generation application to create posts that are engaging and captivating with respect to the text and to the image. As mentioned before, future works could focus on finding new ways to improve text and image quality: not only regarding the precision of the drawing or how much the photo is realistic, but also in terms of what the client imagines as output. At the end of the development of the project, however, what we obtained is an effective application, that can lead to visually impacting images and captivating text.

Bibliography

[1] https://huggingface.co/CompVis/stable-diffusion-v1-4

[2] Mohamad Diab (PublicPrompts), Julian Herrera, Musical Sleep, Bob, Chernow, Coco Mao (2022), *Stable Diffusion Prompt Book*, OpenArt.

[3] Jonas Oppenlaender (2023), *A taxonomy of prompt modifiers for text-to-image generation*, Behaviour & Information Technology.

[4] Louie Giray (2023), *Prompt Engineering with ChatGPT: A Guide for Academic Writers*, BIOMEDICAL ENGINEERING SOCIETY.

[5] VINCOSBLOG, *La prova di Flux 1.0 alternativa open per la generazione di immagini*, https://vincos.it/2024/09/04/la-prova-di-flux-1-0-alternativa-open-per-la-generazione-di-immagini/