Leuphana Universität Lüneburg
Module: 42028000 Technological Basics II
Seminar: Technological Basics II (Stream B)
Instructor: Sarah Haq
Winter Semester 2024/2025

Sahadsawat Khamsuk
Matrikelnummer: 4000595
Sahadsawat.Khamsuk@stud.leuphana.de

Date: February 15, 2025

DoodleFindFilm: Searching Films from Scribbles

Introduction

DoodleFindFilm (formerly VisIt) helps users find a film given a graphic input. The project's idea begins from my observation when I tried recalling films I have watched. I remembered some visual elements such as distinct characters or props in those films. However, sometimes it was also difficult for me to describe these clues in words that can be used for searching the movies. The issue as well made regular search engines and film finding apps or websites that function by inputting textual queries hard to use for me. From this problem, I developed this idea of a program that can look for a film by simply drawing scribbles of what we remember.

Development

The app was developed in PyCharm using Python language. The interface and components of the app are created using Streamlit. I started by cloning Tharindu Madhusanka's repository as the main body of this program. His code provides these prominent functionalities: initializing a collection of film data; filtering results by production year; searching from the database and displaying results; including functions for connecting API and getting results from websites.

Then, I added the drawing pad into the main app. The drawable canvas for users to input their graphic data is a Streamlit component originally created by Fanilo Andrianasolo. The code also includes a sidebar where users can choose a drawing tool, e.g. freedraw, point, and shapes, and setting the stroke's width and color. A picture can be freely drawn by dragging a mouse or using a stylus or finger (if the screen is touchable).

Lastly, function `generate_caption` was then built to transform a graphic input into a textual query for retrieving results from the database. In the definition I employed Hugging Face's image-to-text model to generate a caption describing the graphic input. Also, given that the doodles drawn were stored as an array of numbers, a few lines of code were added to turn the array into image data before passing it to the function.

Design

DoodleFindFilm is a single-page web app and can be divided into three main sections. The first part consists of the canvas. Above it is the instruction of how to use the app. There are also two slidebars for opting a number of films to be displayed and filtering results by release year. Below the canvas are buttons for undoing, redoing and clearing the drawing.

On the left-side, there is a sidebar. Users can select other drawing tools such as point, line, rectangle and circle. Also, users can set the stroke's weight and color. In this same area, there is a small section allowing users to upload a photo instead of drawing by drag-and-drop or browsing file.

The last part is the result section. It will appear after clicking the 'Search' button. This area contains a caption of the picture drawn as well as movies matching the query. For each movie, there display the movie's name linking to its page on TMDB, genres, release year, run time, synopsis, and links to its page on IMDB and YTS.

Problems and Challenges

During the development process, there were some technical issues, but they can be solved by consulting ChatGPT. After finishing developing the app, it can run without problems on my local computer. However, during the process of deploying the app on Streamlit, I encountered two main technical problems.

The first issue involved importing library `chromadb` since there was no supported version of `sqlite3` , which is required for `chromadb`. The issue could be solved by adding several lines of code from this forum to my code. The library could then be imported.

The second issue was that result films were not displayed after inputting and clicking 'Search'. This was because the collection of film data was not uploaded onto my GitHub repository, which is the code source for Streamlit to deploy. The collection file is larger than 25 MB and thus could not be uploaded via GitHub website, and the absence of the file in my repo went unnoticed. After identifying the cause, I solved this problem by git-push-ing the collection file into my repository. Doing so, the app can now display results. While this issue seems difficult to solve, by calmly indicating its cause, the solution can be found.

App Testing

To evaluate the app's functionality and limitations, I performed app testing, focusing on the input's type (drawn or photo) and number. The test result can be summarized as below:

Case	Case Description	Objects	Caption	Expected	Shown	Satisfaction
1	1 drawn object	cat	a black and white drawing of a cat	yes	10/10	very satisfied
2	Case 1 with 1 drawn object added (2 drawn objects)	cat + fish	a drawing of a cat with a face drawn on it	no	10/10	dissatisfied
3	Case 2 with 1 drawn object added (3 drawn objects)	cat + fish + car	a drawing of a face with the word ' i love you '	no	10/10	dissatisfied
4	photo of 3 real objects	cat + fish + car	a black background with a white and red flower	no	10/10	very dissatisfied
5	(new) 2 drawn objects	dog + person	a drawing of a dog with a person walking behind it	yes	9/10	satisfied

From the table, there are cases where not all objects are recognized and the captions expected do not match with the graphic input. From case 2, not all objects drawn are recognized by the image-to-text model since a cat and a fish were drawn but only the cat was recognized. In case 3 and 4, regardless of the input being drawings or a photo of real objects, the captions generated do not at all describe the inputs. The graphic inputs are a cat, a fish and a car. Yet, the captions are completely different things irrelevant to the prompts. These limitations may be due to my drawing skill and photo and due to the AI model itself as well.

It seems to be a technical issue with the database as well. In case 5, only nine films were displayed. The one missing film was instead displayed with a message saying “Error fetching data for movie”. This issue could be considered one of the app’s limitations.

Testers’ Feedback

In total, there are three testers. All of them tested the app by drawing scribbles directly on the canvas. Each of them used different equipment for drawing, i.e. bare hand, dragging mouse, and stylus. Only one tester found a film they were looking for. Most of the testers found the app not easy to use but still felt it intuitive. The app was rated accessible but fairly attractive.

One of the users gave a comment that they received the same wrong results regardless of what or how detailed they drew it. This suggests the AI model should be more developed.

As I was informed that the stroke and background features did not function properly, I made two changes to the app. First, the stroke did not change its color after picking a color. I investigated the issue and found out that after choosing the stroke’s color, the color picker widget has to be closed by clicking anywhere on the screen. I added a text element notifying how to change the stroke’s color. Second, when the background’s color was changed, the app kept returning the same results regardless of what was drawn and suggested that the background’s color be removed. Therefore, I removed this feature to eliminate the issue.