DALI-DOODLE ARCHITECTURE DOCUMENT

26/06/23

OVERVIEW

1. Project Background and Description

Dali-Doodle, or DD for short, is meant to be a feature integrated into the ElliQ product by Intuition Robotics.
The purpose of the project is to generate a prompt to use in an image generator, and to do this via a fluent conversation with the user.
Dali-Doodle uses AI products through api calls or locally but the conversation and NLP analysis is rule based.

The innovation of this project is mainly it's use of simple NLP libraries (SpaCy, GPT2) to create an interactive and fun process to create something completely new in a joint effort with a user.

2. Project Scope

Dali-Doodle aims to develop a core idea in the form of a sentence or short paragraph, and to do so either vocally or via text.

This is done with a step-by-step process composed of input from the user, then feedback by DD, then addition of ideas by DD, until a final idea is achieved and approved by the user, which is then sent to an external image-generation engine either locally or through api calls.

DD is not meant as a computer vision product- it is not meant to generate pictures manually, nor is it meant to do any visual analysis of the picture it creates for users.

DD is meant to be integrated into a larger ux experience, but for the purposes of this project in the context of a final assignment, a web-based ux experience was created to operate the system and receive feedback.

3. High-Level Requirements (Design)

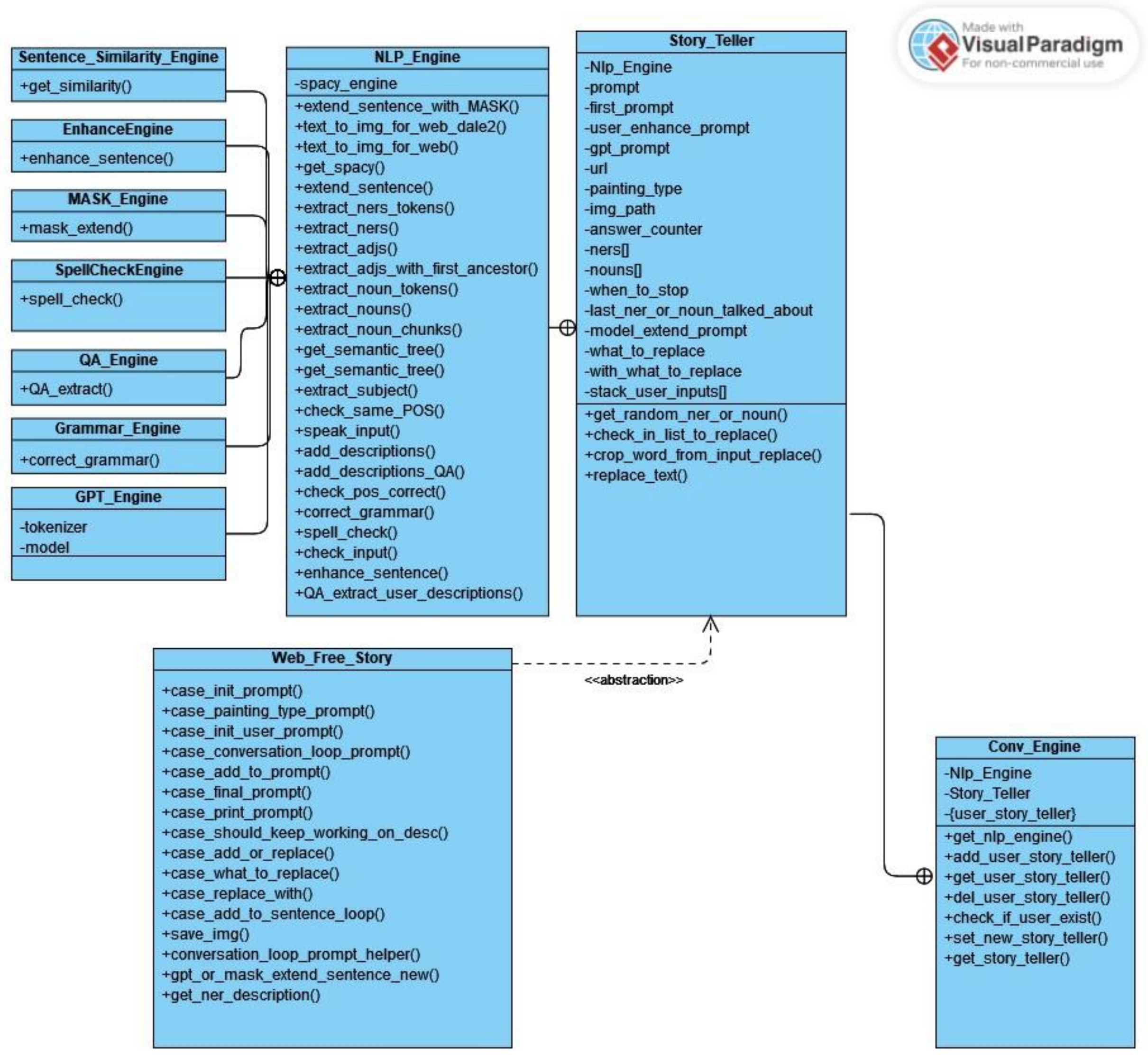
- Ability to access various api's as part of DD's core functions

4. Who are the users?

The intended users of this product are elderly / lonely individuals who wish to draw in an interactive way with the help of an AI based assistant.
With the previous sentence in mind, the possible audience is any person with a solid grasp of english.

5. Project entities (Architecture)

The Architecture of the main classes of the project are shown as follows in the UML graph:



Explanation of the class hierarchy:

Conv_Engine contains all active user conversations with the system in the form of a dictionary mapping a user id to a Story_Teller.

Currently, the only form of Story_Teller Deployed is Web_Free_Story (which inherits from Story_Teller), which contains the conversation loop currently implemented.

The abstraction allows for more types of conversations to be easily integrated (for different types of users, different conversations, different end results, etc.)

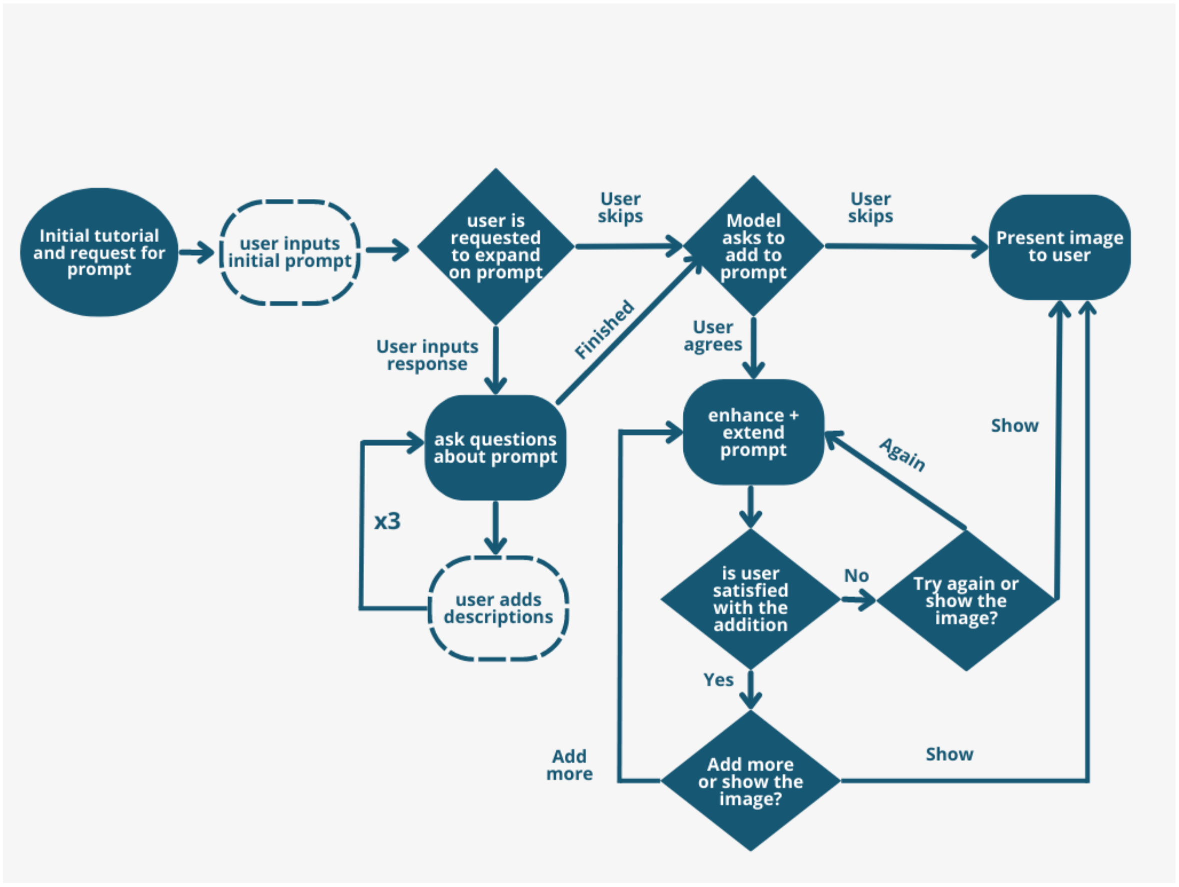
The Web_Free_Story class contains all history with a particular user, the final prompt, parts of the sentence already identified, and the current stage of the conversation the user is currently in.

It also contains an instance of the NLP_Engine class, which provides every language processing the conversation may require (like sentence analysis, input checking, sentence enhancement and sentence extension).

Note that the NLP Engine is shared among the classes, only one instance is created within the system.

The NLP_Engine class is a container of many types of out of the box NLP tools to assist in certain tasks (SpaCy, GPT2, Bert, etc), some of which are contained in classes.
This helps with switching out different engines (like the enhancement engine which was changed regularly during development) without changing the out-facing NLP_Engine interface or the inside implementations of the NLP_Engine core functions.

The following graph details the core conversation flow:


Initial tutorial and request for prompt
user inputs initial prompt
user is requested to expand on prompt
User skips
Model asks to add to prompt
User skips
Present image to user
User inputs response
Finished
User agrees
ask questions about prompt
enhance + extend prompt
Again
Show
x3
user adds descriptions
is user satisfied with the addition
No
Try again or show the image?
Yes
Add more
Add more or show the image?
Show

As seen in the above diagram, the user inputs an initial prompt (e.g. “the dog walked in the park”).
Then, the system recognizes ‘dog’ and ‘park’ as valid nouns in the sentence, and will ask the user for more details on them (“Can you add some more details? How does the dog look like?”).
Once the user has entered some form of description (“the dog is blue”), the system will integrate this into the

sentence (“The blue dog walked in the park”), and ask at most 2 more times for different entities in the sentence.
After this step is completed, the user will be asked if they want the system to improve the prompt.
If yes, the system will add adjectives to the entities in the sentence (“the blue wonderful dog walked in the big park”), and some extra words to extend the sentence (“the blue wonderful dog walked in the big park in the evening”).
This is followed by a three option page - it’s good and I want to keep going/it’s good and I want to paint it/it’s bad, do the enhancing again.
This repeats until the user is satisfied with the end results, and the prompt is sent to the image generation engine.
Finally, the user is requested to fill out a form detailing their experience.

6. Data Collection from user

1. A CSV file containing the entire log conversation for each user.

You can find in the program folder, inside the folder “../users_history”.

2. MongoDB - all rank page information, including:

The user's gender, age, first time, satisfaction, promptness, overall experience, relevant text suggestions, and any issues or anything relevant they would like to add.

As well as collecting each step of the prompt enhancement process
(first -> user enhancement -> GPT enhancement -> final prompt)

URL to see this page: “../stat”

3.Images created by the app.

URL to see this page: “../images_generated”

7. Experiment Flow

The Project is hosted on a crowd-sourcing site like Prolific or Mechanical Turk, where the user receives a link to the project site.
Then, on the project landing page the user is shown an interactive tutorial on how to use the system.
After the finishing the tutorial, the user is redirected to the main page where a text box is shown with the initial prompt – “let’s draw together, what would you like to draw today?”

Then, the user goes through the conversation loop – the user will input an initial sentence, e.g. “the dog rode the tricycle”, after which DD will ask for descriptions of various objects in the sentence, e.g. “can you describe the dog in more detail?”

After 3-5 loops of this, DD will offer to enhance the sentence on it’s own, e.g. “I’ve enhanced the prompt, it is now “the brown muscular dog rode a tricycle”, do you like it?

The user will go through this process with DD until he is satisfied with the enhancement, then the user is shown the final image and redirected to the feedback page where the user gives feedback on the process and the output.

Finally, the user is shown a return code for the crowdsourcing site and needs to input this unique code to receive their compensation.

NOTE: this is deployed practically by the developers receiving a code from prolific, e.g. “EA3FCV”, then inserting this code into the “thank.html” page at line 126. Users will then see the code at the very end of the experiment, after the feedback stage.

8. Required Installations

This project requires several python libraries and AI models (mostly from huggingface). Below is a comprehensive list of all the required libraries and models to install:

brotli==1.0.9

ConfigParser==5.3.0

cryptography==40.0.2

Cython==0.29.28

dl==0.1.0

docutils==0.20.1

en_core_web_sm==3.4.1

Flask==2.3.2

Flask_SocketIO==5.3.3

happytransformer==2.4.1

HTMLParser==0.0.2

importlib_metadata==6.4.1

jnius==1.1.0

keyring==24.2.0

lockfile==0.12.2

lxml==4.9.2

numpy==1.23.5

openai==0.27.6

ordereddict==1.1

Pillow==9.5.0

protobuf==4.23.3

pymongo==4.3.3

pyOpenSSL==23.2.0

pywin32==305

```
sentence_transformers==2.2.2

spacy==3.4.4

toml==0.10.2

torch==2.0.1+cu118

tornado==6.3.2

transformers==4.26.1

typing_extensions==4.6.3

xmlrpclib==1.0.1

zipp==3.15.0
```

you can insert this into a ‘requirments.txt’ file in the project directory to install all dependencies at once.

9. Links

github for the project - https://github.com/sagivantebi/DaliDoodle-Development-AWS

our emails:

Ben - ganonben@gmail.com

Omri - omribh0@gmail.com

Sagiv - sagiva13@gmail.com