Project Specification

This project is part of my research on the relationship between visual and narrative features and how we can use deep learning to better understand them. The goal of this project is to collect data, make human and machine based experiments on the classification of characters according to their narrative prominence based on their character design, and analyse and visualise the results. The goal being to identify the impact different features have on a character's perceived prominence and improve deep learning performance on the task.

The project can be used by AI or Data Science researchers to evaluate how different models perform on a task that relies on implicit patterns present in the data, and how this knowledge can be expanded to other tasks. It can also be used in the task of automatic character generation, helping create characters based on narrative parameters.

Or by researchers in the areas of character design or narratology to better understand how audiences react to different visual features and how those correlate with narrative features. Understanding these correlations and patterns may help character designers create characters that more closely follow (or deliberately subvert) audiences' expectations and writers know the implicit implications of how they visually describe their characters.

It's important to note that knowledge from this project relies on general patterns, so following them too closely may lead to creations, be they automatic or human, feeling formulaic or stale.

The workflow of the project is divided in three steps:

Collecting the Data: In this step the user gatters labelled images from myanimelist, with the images being automatically organised according to their label. It's a slow process, limited by the site's transfer limits.

Running the Experiments: Here the user runs human and machine based experiments, using a ResNet50 as a classifier. After that the user aggregates the results of both experiments.

Analysing the Results: The user compares the human and model results, visualising how they're impacted by different visual features.

Project Requirements

- A user should be able to use this project from start to finish, starting with collecting the data and finishing with analysing the results of the experiments. This is useful because new characters are constantly added to the source of our data, so updating the dataset may be useful in the future.
- A user should not need to collect their own data. Collecting new data is a slow process, so we provide the data originally used to make sure that users can start their experiments with that, if they so desire.
- A user should be able to make their own analysis of our results. Along with the previous requirements, this ensures that users can start the process (collect data -> experiment -> analyse) at whichever point they desire.

Requirements per Step

Collecting the Data:

- The data collected should be labelled.
- The data collected should be organised according to the label.

- The data already collected should not be lost in case of issues in the myanimelist servers.
- The script should avoid continuous requests at a time where there might be an issue with the server.

Running the Experiments:

- Should receive as input the a labelled dataset.
- Should output the results of the experiments as JSON files, indexed per character, to make it easier to compare the results.
- The model should be able to be trained with the data collected in the previous step.
- The training history of the model should be saved.
- The model should be evaluated on a test subset. And the results of this test should be saved as a JSON file.
- The data used for the human survey should be the same as the test subset for the model.
- Each survey taker should classify a character only once.
- Each survey taker should classify the same characters, but at a random order.
- Each survey taker should only classify characters they don't know.
- The results of the human survey should be saved individually per survey taker.
- The individual responses to the survey should be aggregated in JSON files.

Analysing the Results:

- Should receive as input the JSON files with the test results.
- Should work with the JSON files generated in the previous step.
- Should compare the performance of humans and the model.
- Should analyse and show graphs representing how each visual feature impacts the results for humans and the model.

Project Architecture

Collecting the Data:

- Image Gattering.py: This script is responsible for collecting labelled data from the internet and organising it according to the label. It receives no input and its ouput is an "Images" folder with the data.

Running the Experiments:

- Resnet/Train and Eval.ipynb: The purpose of this Notebook is to train CNN models on our dataset and evaluate their performance on the classification task. It uses a ResNet50 from Tensorflow as an example, but the model can be easily changed. It receives the labelled data as an input and outputs the model's weights and training history, and a JSON file with the test results indexed by character.
- Human Classification Test/Human Classification Test.ipynb: This Notebook implements a survey meant to be run on Google Colab. The code implements an interface survey takers can use. It receives as input the test subset, which must be uploaded to Google Drive and outputs the individual answers and character order to Google Drive.
- Human Classification Test/Generate JSON.ipynb: The purpose of this Notebook is to aggregate the answers from the survey into JSON files indexed by character, in order to facilitate the future analysis of the results. It receives as an input the individual answers and character order from the survey and outputs three JSON files with the aggregate results.

Analysing the Results:

- Human Classification Test/Generate JSON.ipynb: On top of its tasks for the previous step, this file also takes care of calculating human accuracy on the classification test.
- Results Analysis.ipynb: This Notebook is meant to analyse and visualise the results of the experiments. It compares both the model and human survey results to the ground truth, while evaluating how they are impacted by different visual features. It receives the results from the model and human tests as JSON files indexed by character, as a series of visual features also as JSON files indexed by characters*. It outputs graphs showing how different features impact humans and the model differently, and how they alter the chances of a character receiving a certain label.

* We extracted our desired visual features with CLIP and Illustration2Vec, so we cannot provide the code for that in this project.. You can find more about how to use them on their respective Github pages linked above.

User Documentation

There are two main groups of users for this project:

- AI or Data Science researchers who want to try and improve performance on this task or wanna see how its performance has evolved with newer data.The project allows this group to collect their own data, or run new experiments on our original dataset, while having code ready to analyse their new results with the questions we asked during our analysis.
- Researchers or professionals in the fields of character design or narratology who want to see the results of our experiments and possibly ask their own questions based on their knowledge of the topic. For them, it provides an environment where they can easily do so, requiring some, but not much, coding knowledge.

The level of technical knowledge required to use this project depends on which step the user desires to start. Just running the analysis only requires basic knowledge of Jupyter Notebook, as it’s meant to be usable by the second group, and adding new questions to the analysis is doable with basic coding knowledge. Running the project from the start also doesn’t require much technical knowledge, but if the user wants to collect their own data or run their own experiments, it’s likely they will wanna make changes to our experiments to compare their results with ours, which would require technical knowledge.

Installing the Project

A user can find the code in our Github and download the repository from there. After that, they can follow the instructions from the README file about in which order they should run the project depending on their desired use case.