**Deep Fiction : Every Image has a story to tell.**

Input has all the trained models. This folder is available on google drive (signin through Lion Mail)

[**https://drive.google.com/open?id=1jqopTG1DRwWIo2ffJVh7XbKDuc7S5\_zk**](https://drive.google.com/open?id=1jqopTG1DRwWIo2ffJVh7XbKDuc7S5_zk)

**Baseline Model**

First put paths to all the models romance dictionary, romance style, caption style in the right place in config.py

Now use generate\_stories.py to run a folder of images and k and bw can be adjusted, where k is the number of captions to condition on and bw is the beam width.

Recommended setting is k = 100, bw = 50 (for base line model)

In this file generate is being imported this file has only the baseline model. If we want to run for a single file we can also use the following steps -

We have made improvements on this model by improving captioning using the following two models. They can be trained as follows –

import generate

z = generate.load\_all()

generate.story(z, './images/ex1.jpg', k=100, bw=50)

**Steps to train your own model can be followed on** [**https://github.com/ryankiros/skip-thoughts/tree/master/decoding**](https://github.com/ryankiros/skip-thoughts/tree/master/decoding)**.**

Explained in a lot of detail.

**Model 2 - Good Caption**

**More details -** [**https://github.com/karpathy/neuraltalk2**](https://github.com/karpathy/neuraltalk2)

**The model was used to improve input before stories were generated. It has been further described in detail in the reports.**

**Step 1** : MS COCO is downloaded using coco python API

**Step 2** : The dataset is preprocessed so that it can be trained on

python prepro.py --input\_json coco/coco\_raw.json --num\_val 5000 --num\_test 5000 --images\_root coco/images --word\_count\_threshold 5 --output\_json coco/cocotalk.json --output\_h5 coco/cocotalk.h5

**Step 3** : The command is used to start the training the file

th train.lua -input\_h5 coco/cocotalk.h5 -input\_json coco/cocotalk.json

**Step 4** : Just to test on new images

th eval.lua -model /path/to/model -image\_folder /path/to/image/directory -

**Model 3 - Captioning phrases for regions**

**More details -** [**https://github.com/jcjohnson/densecap**](https://github.com/jcjohnson/densecap)

**The model was used to improve input before stories were generated. It has been further described in detail in the reports.**

**Step 1** - Download Visual Genome

**Step 2** - Preprocessing the downloaded data

python preprocess.py --img\_dir /image\_folder\_path

**Step 3** - Train the model and it loads from the saved checkout if needed

th train.lua -checkpoint\_start\_from /path\_to\_checkpoint\_file

**Step 4** - Testing the model images

th run\_model.lua -input\_dir /path/to/my/image/folder -output\_dir /path/to/output/folder/

**Pipeline**

**When running generate2 and generate3 use k=20 and bw=5. It can also be used as generate given above.**

These two image captioning models can be run on together using the following bash file

sh compile.bash

This makes two files - output.txt and results.json

Python pipeline.py is run to make a file that matches the image names and combines captions in their order and makes a file called input\_story.py

This file is then fed to generate3.py which can either be called individually on single images or can be changed in generate\_stories.py to be run on a folder of images

If only one good caption is to be used then either generate2.py is used for a single image or generate\_stories.py is used with generate2 to run on folders.

**Evaluation Codes**

1. BLEU Score

bleu.py is used to run on predict.txt and text.txt where predict.txt has the generated captions from baseline model or model 1 and text.txt contains the actual captions. Using nltk package we find their score.

Python bleu.py

2. Language\_Check

Run the following command inside the ‘langauge\_check\_files’ folder to get the average number of mistakes produced by each of the 3 models for story generation.

python language\_check\_trial.py

**Training on a new genre**

We also tried training our entire model on another genre. It was the Children’s Book dataset, provided by Facebook research.

It was a Q&A dataset, so it did not exactly pertain to our needs and we had to do a bit of preprocessing. Each passage had a question at the end of it and we removed that, as well as other declarative statements, which marked the start of a chapter or the start of a book.

The preprocessing script is inside the folder called ‘children’s book’ which also has the book data. The script is written in python3 and can be run as follows :

python tokenize.py

Please note that the path for the data would have to be changed in order to run the script.

This script generates a new file called ‘new\_file.txt’. This is read during the training for the new genre. The reading of the file to give as input to the training algorithm is being handled in the function called ‘read\_sentences’ in train.py.

Execute this command for generating the children’s book style file.

python new\_pickle.py which generates ‘childrens\_book\_style.npy’.

However, we hit a roadblock and results were not generated for this genre.

We attempted at predicting the preposition between bounding boxes but that also did nothelpthe score as well. Still attaching the file. This is run on results.json, the output of captioning model that generates phrases associated with regions.