FROM AN IMAGE TO A TEXT DESCRIPTION OF THE IMAGE

Interim Report

By

Lee Kuan Ing Aavan
U1821828L

Supervisor: Assoc Prof Chng Eng Siong

School of Computer Science & Engineering
2021

Submitted in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Engineering (Computer Engineering)
in Nanyang Technological University

Objective of the Study

The objective of this project is to develop a program that generates accurate descriptive captions of the extracted keyframes in a video.

Scope of Work

a) Review existing image captioning programs.

b) Extract keyframes from a video.

c) Apply image captioning on the extracted frames.

d) Integrate into MAGOR pipeline.

e) Improve image captioning models.

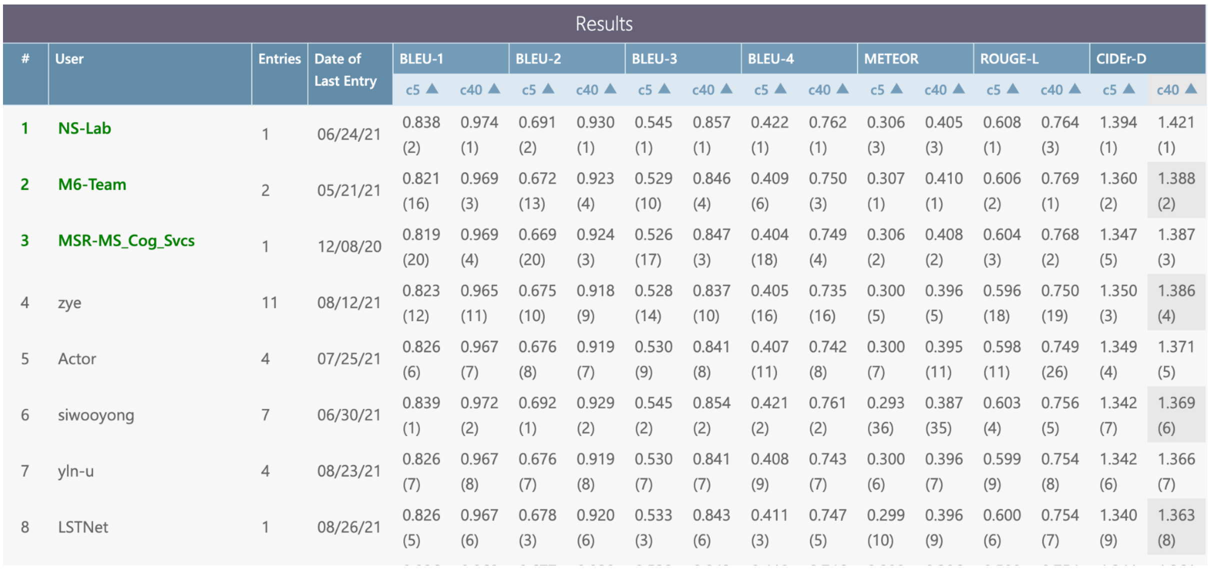
Literature Review

Image captioning is a relatively new, growing focus in Artificial Intelligence research. Many methods are in existence currently which achieve satisfactory results, but are still lacking compared with human results. Some notable programs are Facebook's AI, Amazon's Rekognition and Imagga.

A typical image captioning model will consist of an Encoder(CNN), Decoder(RNN, usually LSTM), Attention and Beam Search. The CNN understands the contents of an image and detects objects present, RNN generates words and assigns weights, Attention stage computes the weights and Beam Search selects the best caption sequence.

The most commonly used datasets for image captioning are MSCOCO and Flickr. BLEU-1 to BLEU-4, CIDEr and SPICE are among commonly used evaluation matrices.

Table 1 shows the current leaderboard and their respective scores for Image Captioning models using the MSCOCO Dataset.



Results

#	User	Entries	Date of Last Entry	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr-D	
				c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲
1	NS-Lab	1	06/24/21	0.838 (2)	0.974 (1)	0.691 (2)	0.930 (1)	0.545 (1)	0.857 (1)	0.422 (1)	0.762 (1)	0.306 (3)	0.405 (3)	0.608 (1)	0.764 (3)	1.394 (1)	1.421 (1)
2	M6-Team	2	05/21/21	0.821 (16)	0.969 (3)	0.672 (13)	0.923 (4)	0.529 (10)	0.846 (4)	0.409 (6)	0.750 (3)	0.307 (1)	0.410 (1)	0.606 (2)	0.769 (1)	1.360 (2)	1.388 (2)
3	MSR-MS_Cog_Svcs	1	12/08/20	0.819 (20)	0.969 (4)	0.669 (20)	0.924 (3)	0.526 (17)	0.847 (3)	0.404 (18)	0.749 (4)	0.306 (2)	0.408 (2)	0.604 (3)	0.768 (2)	1.347 (5)	1.387 (3)
4	zye	11	08/12/21	0.823 (12)	0.965 (11)	0.675 (10)	0.918 (9)	0.528 (14)	0.837 (10)	0.405 (16)	0.735 (16)	0.300 (5)	0.396 (5)	0.596 (18)	0.750 (19)	1.350 (3)	1.386 (4)
5	Actor	4	07/25/21	0.826 (6)	0.967 (7)	0.676 (8)	0.919 (7)	0.530 (9)	0.841 (8)	0.407 (11)	0.742 (8)	0.300 (7)	0.395 (11)	0.598 (11)	0.749 (26)	1.349 (4)	1.371 (5)
6	siwooyong	7	06/30/21	0.839 (1)	0.972 (2)	0.692 (1)	0.929 (2)	0.545 (2)	0.854 (2)	0.421 (2)	0.761 (2)	0.293 (36)	0.387 (35)	0.603 (4)	0.756 (5)	1.342 (7)	1.369 (6)
7	yln-u	4	08/23/21	0.826 (7)	0.967 (8)	0.676 (7)	0.919 (8)	0.530 (7)	0.841 (7)	0.408 (9)	0.743 (7)	0.300 (6)	0.396 (7)	0.599 (9)	0.754 (8)	1.342 (6)	1.366 (7)
8	LSTNet	1	08/26/21	0.826 (5)	0.967 (6)	0.678 (3)	0.920 (6)	0.533 (3)	0.843 (6)	0.411 (3)	0.747 (5)	0.299 (10)	0.396 (9)	0.600 (6)	0.754 (7)	1.340 (9)	1.363 (8)

Table 1
Source: https://competitions.codalab.org

Implementation

1. **Reviewed open source projects for video keyframe extraction.**

   Tested each program and modified and added parameters to achieve the best results. Accuracy is measured by the number of keyframes extracted out of total keyframes in reference videos. Reference video for timing measurement is 3m 38s long, with 36 keyframes.

   a) https://github.com/amanwalia123/KeyFramesExtraction

      Extracts many repeated frames.

      Accuracy: **86.1%**. Reference video time taken: **1m 10s**.

   b) https://github.com/keplerlab/katna

      Some keyframes not detected by this program.

      Accuracy: **75%**. Reference video time taken: **3m 25s**.

   c) https://github.com/Zhujunnan/shotdetect

      Observe better frame detection with this program.

      Accuracy: **88.8%**. Reference video time taken: **49s**.

   d) https://ffmpeg.org/ffmpeg.html

      Good frame detection. Runtime is much faster.

      Accuracy: **83.3%**. Reference video time taken: **9s**.

   e) https://github.com/Breakthrough/PySceneDetect

      Most accurate frame detection. Fast runtime.

      Accuracy: **91.7%**. Reference video time taken: **9s**.

      Currently using PySceneDetect in content-aware detection mode, with a threshold setting of 29.

2. **Reviewed open source projects for image captioning with pretrained models.**

2.1) *Below (a) to e)) is a list of image captioning projects that were tested unsuccessfully or with no pretrained models. Initially attempted to train the model personally but results obtained were very inaccurate.*

a) https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_10_4_captioning.ipynb

   Jupiter notebook, no pretrained model available. Inaccurate captions.

b) https://github.com/tensorflow/docs/blob/master/site/en/tutorials/text/image_captioning.ipynb

   Jupyter notebook, no pretrained model available. Inaccurate captions with 40,000 images for training.

c) https://github.com/microsoft/Oscar

   No pretrained model available.

d) https://github.com/peteanderson80/Up-Down-Captioner

   Unable to test this program as it requires an NVIDIA graphics card.

e) https://github.com/HughKu/Im2txt

   This program implements Tensorflow v1, as such, many deprecated modules needed to be replaced. Unsuccessful in debugging fully.

2.2) *Below (f) to h)) is a list of image captioning projects that were tested successfully with pretrained models. Tested each program and assessed the accuracy of the image captions generated. Reference video for timing measurement is 46min 13s long, with 270 keyframes.*

f) https://github.com/yuyay/chainer_nic

Implemented with pretrained model MSCOCO iteration 50000 (best model available). Captions are more accurate than captions from a) and b), but are still quite limited. Captions were not evaluated against MSCOCO dataset at this stage.

Reference video time taken: **25min 11s**

g) https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning#implementation

Implemented using pretrained model checkpoint and wordmap. Captions are much more accurate. Scores are evaluated against MSCOCO val2014 dataset.

BLEU-1: 0.773	CIDEr: 1.185
BLEU-2: 0.620	SPICE: 0.212
BLEU-3: 0.489	METEOR: 0.285
BLEU-4: 0.386	ROUGE-L: 0.581

Reference video time taken: **6min 1s**. Currently using this model.

h) https://github.com/ruotianluo/ImageCaptioning.pytorch

Tested the 3 available pretrained models in the ModelZoo. Captions are generally better than f) but not as accurate as g). Scores are evaluated against MSCOCO val2014 dataset.

Model: FC

BLEU-1: 0.732	CIDEr: 0.953
BLEU-2: 0.561	SPICE: 0.182
BLEU-3: 0.413	METEOR: 0.250
BLEU-4: 0.302	ROUGE-L: 0.535

Model: FC+SC

BLEU-1: 0.761	CIDEr: 1.099
BLEU-2: 0.598	SPICE: 0.193
BLEU-3: 0.451	METEOR: 0.262
BLEU-4: 0.337	ROUGE-L: 0.555

Model: FC+NSC

BLEU-1: 0.765	CIDEr: 1.115
BLEU-2: 0.601	SPICE: 0.194
BLEU-3: 0.453	METEOR: 0.263
BLEU-4: 0.338	ROUGE-L: 0.556

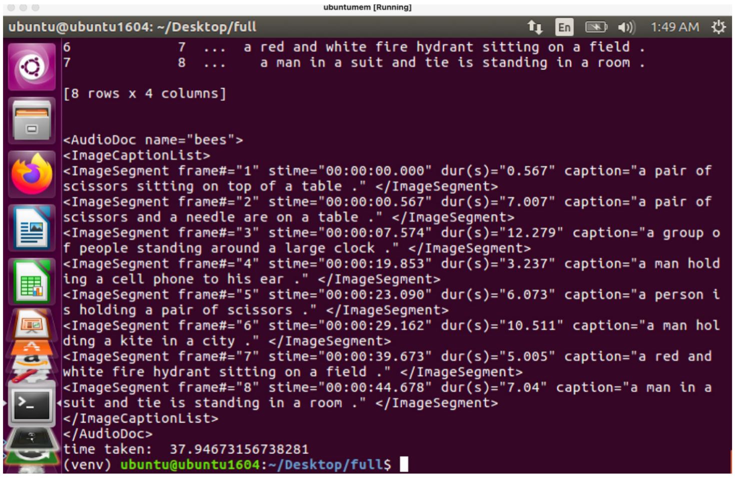
Reference video time taken: **13min 40s**.

3. **Created basic API for program.**

   Allows user to browse disk for video file to upload.

4. **Transferred packaged working program into MAGOR server for integration.**

   Documented full requirements for use in Ubuntu 16.04.



```
6              7  ...  a red and white fire hydrant sitting on a field .
7              8  ...    a man in a suit and tie is standing in a room .

[8 rows x 4 columns]


<AudioDoc name="bees">
<ImageCaptionList>
<ImageSegment frame#="1" stime="00:00:00.000" dur(s)="0.567" caption="a pair of
scissors sitting on top of a table ." </ImageSegment>
<ImageSegment frame#="2" stime="00:00:00.567" dur(s)="7.007" caption="a pair of
scissors and a needle are on a table ." </ImageSegment>
<ImageSegment frame#="3" stime="00:00:07.574" dur(s)="12.279" caption="a group o
f people standing around a large clock ." </ImageSegment>
<ImageSegment frame#="4" stime="00:00:19.853" dur(s)="3.237" caption="a man hold
ing a cell phone to his ear ." </ImageSegment>
<ImageSegment frame#="5" stime="00:00:23.090" dur(s)="6.073" caption="a person i
s holding a pair of scissors ." </ImageSegment>
<ImageSegment frame#="6" stime="00:00:29.162" dur(s)="10.511" caption="a man hol
ding a kite in a city ." </ImageSegment>
<ImageSegment frame#="7" stime="00:00:39.673" dur(s)="5.005" caption="a red and
white fire hydrant sitting on a field ." </ImageSegment>
<ImageSegment frame#="8" stime="00:00:44.678" dur(s)="7.04" caption="a man in a
suit and tie is standing in a room ." </ImageSegment>
</ImageCaptionList>
</AudioDoc>
time taken:  37.94673156738281
(venv) ubuntu@ubuntu1604:~/Desktop/full$ 
```

5. **Developed scripts to evaluate against datasets in MSCOCO format.**

   Allows evaluation against original MSCOCO dataset and any custom dataset in similar format, displaying BLEU-1 to 4, METEOR, ROUGE-L, CIDEr and SPICE scores. Evaluation script references from https://github.com/salaniz/pycocoevalcap. Currently writing script to create custom MSCOCO dataset with junk values for info and licenses sections from json of image file names and 4 captions for each image.

Tentative Plan

1. **Complete scripts for evaluation and formatting.**

   a) Create script to evaluate the output file of the video captioning program, given captions for the frames generated.

   b) Finish writing script to create custom datasets in MSCOCO format, given json file of image filenames and captions.

2. **Train model with custom dataset.**

   Plan to create a custom dataset that can be used to train models that will be more suitable for the goals of MAGOR, and possibly train a better model for usage.