CS512 - FINAL PROJECT REPORT

“Caption generation”

A20432671 Thomas Ehling
&
A20432063 Benjamin Scialom

Table of Contents :

I. Project Statement

The goal of the project was to generate captions for a given picture. It requires to understand the content of the picture and use natural language processing to produce the caption. This topic is constantly examined by scientist in computer vision and deep learning.

Neural Image Caption (NIC) uses a **convolution neural network** that encodes an image into a compact representation, followed by a recurrent neural network (RNN : language generating neural network) that generates a corresponding sentence. The model is trained and evaluate to maximize the likelihood of the sentence given the image.

According to our publication the encoder RNN is replacing by a deep convolutional neural network (CNN). Actually, it is very common to use a CNN as an image “encoder”, by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences.

However, our project focused on the computer vision part. So we just adapted a code from internet for the natural language processing part.

There are 4 main steps to our project :

- Prepare the data-set of Flickr8k : extract features from the images and clean the captions
- Define and train the caption generator model
- Evaluate the performance of the model
- Use the model to generate captions from an image

Every sections above Section IV “Proposed Solution” has been redacted by both Benjamin and Thomas, and therefore will figure in both of the reports. Sections IV, and all sections after it are specific to each of the student, and will be different in each of the reports.

II. Organization and responsibilities

First of all, we have worked on the general aspect/idea of the code. That is to say what was needed to implement all that was required to make the generation of caption work.
Then, we have split the project in different step/code to work on different aspects at the same time.

On one hand, the preparation of the data-set and the training have been implemented on Python notebooks, using colab. That was necessary since a large computing time is necessary to prepare the data-sets and to train the models on it.

On the other hand, the evaluation and the final test have been developed in python scripts, that can be run locally.
Doing that allow the user to generate captions from an image without using colab. He just has to run the **“test.py”** python file, giving the image and the model files.

<u>What was our responsibilities ?</u>

- We **both work on the construction of the extraction-feature model** which is used for the preparation of the data-set and the training. It is also used in the final test file.

- **Thomas** took care of the preparation of the data-set and the evaluation of the model. (prepare_dataset.ipynb, eval.py)

- **Benjamin** took care of the model training and the final test file which is used to predict caption according to a picture. (train.ipynb, test.py)

III. Implementation

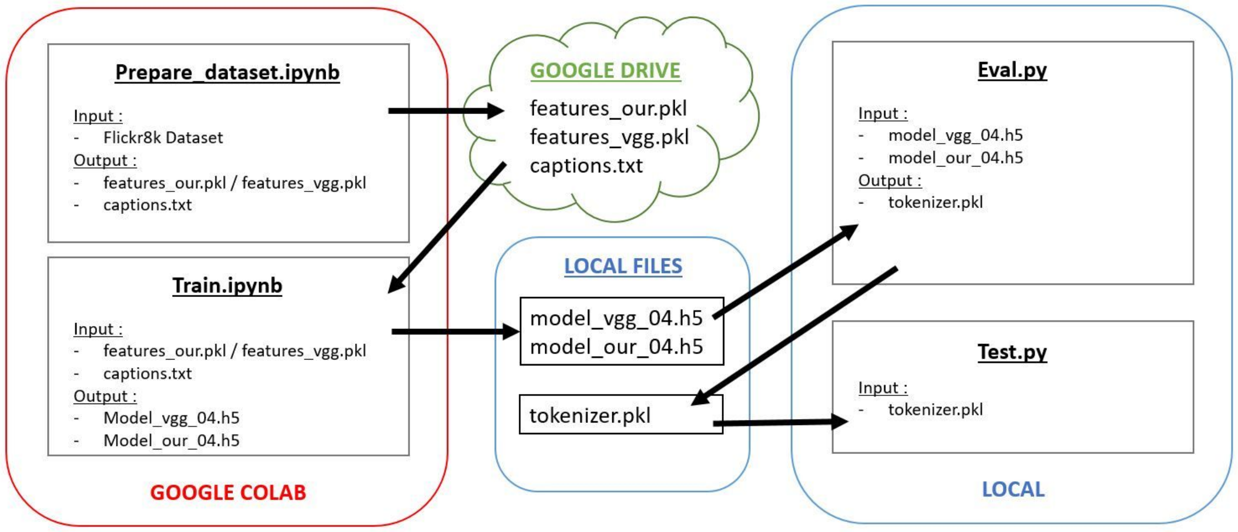
We made 4 program files, one for each part of the project :

- 2 Python Notebook files :
    - prepare_dataset.ipynb : extract features from the images and clean the captions
    - train.ipynb : define and train the caption generator model
- 2 Python files :
    - eval.py : evaluate the performance of the model
    - test.py : generate caption for the given image

The two file in the Notebook format are computing the training of the several models (cf Proposed Solutions), and have been made to be run on the Google Colab Plateform.

We used Python 3.5.

The relations between the files are represented in the following graph :



As each files generate data that are used by the other ones, there is a precise order to run them.

1. prepare_datadet.ipynb : (COMPUTATION TIME : ~3H)
    a. Create a folder “CV_Project” on your drive account
    b. Open the file with the Google Colab platform
    c. Click on “run all”
    d. Select the Google account with the drive containing the “CV_project” folder
    e. Wait for the computation to finish

2. traint.ipynb : (COMPUTATION TIME : ~6H)
    a. Open the file with the Google Colab platform
    b. Click on "run all"
    c. Select the Google account with the drive containing the "CV_project" folder
    d. Wait for the computation to finish
    e. Download the desired .h5 files generated in the "CV_project" folder, for the next part
3. eval.py : (COMPUTATION TIME : ~9 MINUTES)
    a. Execute the file from the command line, with the following arguments : (If no argument are given, default files will be loaded from the /data folder)
        i. **-m** next argument should be the **path of the file.h5** you want to use
        ii. **-f** next argument should be the path of the **feature.pkl** you want to use

-> for these two files used by the algorithm you have the choice between "our" or "VGG16". If you are using "our" make sure it is precise in the name of the file (it is by default).

4. test.py :
    b. Execute the file from the command line, with the following arguments :
        i. -p next argument should be the picture you want to predict (mandatory)
        ii. -t next argument should be the token file you want to use. This should not be used.
        iii. **-m** next argument should be the **path of the file.h5** you want to use. It depends which model is used "our" or "VGG"

Because of the long computation time, all our resulting files are given in the /data folder. Doing so, any part can be executed separately, given the right files (cf graph).

IMPORTANTE NOTE :

All along the project, we used two models to generate the feature : the predefined model VGG16 and our own model.

This choice have been made in order to test the functionalities very soon with the predefined model, and have a better idea of our model performance.

Therefore, whenever a model or a file is created, it is always specified if it has been made with our model or with the vgg model within the name of the file.

If you have any questions about the implementation part (or the rest of the report), please feel free to reach out to any of us to the following email :

- bscialom@hawk.iit.edu
- tehling@hawk.iit.edu

Or ask for an appointment, so we can show you a live demonstration.

IV. Proposed Solution

From now on, all the remaining of the report have been redacted by, and describe the work of I, Thomas Ehling.

A. Prepare_data.ipynb

Inputs :

- Link to **Flickr8k datasets**
- Link to the google drive where the files are saved.

Output :

- **features_vgg.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with the VGG16 model.
- **features_our.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with our own model.
- **captions.txt :** text files with all the cleaned captions, computed with the captions from the Flickr8k dataset.

A.I Extract features from the dataset.

The dataset folders are fetch and save with the keras function "tf.keras.utils.get_file" where the link to the datasets is specified.

In order to extract the features, we need to use a CNN model.

The Architecture and parameters of the model we used has been chosen by **Benjamin and I together.**

As the default model we selected to test the performance of our project was the VGG16, our own model is based on the same architecture that the VGG16 :

```
#create the model
def create_model():
    model = Sequential()
    model.add(ZeroPadding2D((1,1),input_shape=(224,224,3)))
    model.add(Conv2D(receptive_fields[0], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[0], (3, 3), activation=my_activation))
    model.add(MaxPooling2D(pool_size=my_pool_size, strides=my_stride))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[1], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[1], (3, 3), activation=my_activation))
    model.add(MaxPooling2D(pool_size=my_pool_size, strides=my_stride))

    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[2], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[2], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[2], (3, 3), activation=my_activation))
    model.add(MaxPooling2D(pool_size=my_pool_size, strides=my_stride))

    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[3], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[3], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[3], (3, 3), activation=my_activation))
    model.add(MaxPooling2D(pool_size=my_pool_size, strides=my_stride))

    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[4], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[4], (3, 3), activation=my_activation))
    model.add(ZeroPadding2D((1,1)))
    model.add(Conv2D(receptive_fields[4], (3, 3), activation=my_activation))
    model.add(MaxPooling2D(pool_size=my_pool_size, strides=my_stride))

    model.add(Flatten())
    model.add(Dense(dense_values[0], activation=my_activation))
    model.add(Dropout(dropout))
    model.add(Dense(dense_values[1], activation=my_activation))

    return model
```

In order to make the code granular and cleaner, all the important parameters for the model are declared as global variables and can be tune very easily. (strides, activation function, receptive fields, …)

Once the model has been defined, it is loaded in memory and we can proceed to the features extractions.

As the feature extraction is a time-consuming process (~1h30 by model) a security has been add to check if the file already exists in the drive (from a previous run) and launch the process only if there isn't any match.

Then, for each model, the following extraction feature function is called :

```
# extract features from the pictures of the Flickr Dataset
def extract_features(directory, model):

  # we save the features in a dictionnary
  # the image ids are the key, the features the associated values
  features = dict()
  files = listdir(directory)
  print("number of images : ",len(files))

  # We use tqdm library to show the progress of the process
  for index in tqdm(files):
    #load the image
    image = load_img(directory + '/' + index, target_size=(224, 224))
    image_ar = img_to_array(image)

    # reshape for the model
    image_re = image_ar.reshape((1, image_ar.shape[0], image_ar.shape[1], image_ar.shape[2]))
    image_re = preprocess_input(image_re)

    # get features
    feature = model.predict(image_re, verbose=0)
    image_id = index.split('.')[0]
    features[image_id] = feature
  return features
```

It extract every image from the Flickr8 images dataset. The image are reshape and process prior to be given to the model. Then the selected model is used to generate one feature from each picture.

The "tqdm" library have been used to show the progress of the computation, alongside the time spent on the computation and an estimation of the remaining time.

Finally the resulting "features" variable is save to a pickle file on the drive with the "dump" function, with the name "features_vgg.pkl" for the featres process by the vgg model, and "features_our.pkl" for the ones process with our model.

A.II Clean the dataset captions.

The Flickr8 captions dataset is loaded the same way as the images for the previous part.

The objective of the cleaning process, is to check if there is any repetition or words that are one character in length, remove punctuation, remove words with numbers in it, and convert everything to lowercase.

The captions are in the file “/Flickr8k.token.txt”, where all the image id are written with the corresponding caption right next to them, separated by a “.”

When the captions are successfully loaded in memory, a dictionary variable is used to store the captions with the image id as the key.

Then, the captions are cleaned through basic string manipulation, and save to a file “captions.txt” on the drive. As it is only composed of strings, it can be save to a txt format and does not have to be save to a pickle format, like the previous features.

B. eval.py

Inputs :

- **model_vgg_04.h5 :** file with all the weights of the caption generation model, trained on the features generated by the vgg model.
- **model_our_04.h5 :** file with all the weights of the caption generation model, trained on the features generated by our model.
- **features_vgg.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with the VGG16 model.
- **features_our.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with our own model.

Output :

- **tokenizer.pkl :** pickle file, used to save the value of the tokenizer variable

The goal of this program is to evaluate the performance of the caption generation model

According to our publication, the most commonly used metric so far in the image description literature has been the BLEU score, which is a form of precision of word n-grams between generated and reference sentences. So, the model will be evaluated thanks to this tool which is commonly used and even has its library.

We will focus on the BLEU-1 score, as it is the score used in the article.

The BLEU score is calculated from two list, one with the original captions from the Flickr dataset, and one for our predictions.

Here are the steps and the core code in order to generate these two elements :

B.I Get the datasets

The user inputs are computed thanks to the parser library.

From the path to the h5 file, the model is loaded.
From the path to the features file, the features are loaded to a list.

To proceed to the evaluation, we have to split the training set from the testing set.
In order to do so, there are two files already available in the Flickr8k dataset : Flickr8k_testImages.txt, and Flickr8k_trainImages.txt. These two files contains all the ids of the images for the training and testing set.

With the ids provided by these files, we are able to split the features list into a training and testing datasets. In the code the function used for this operation is the function *“get_flickr8k_datasets()”*

We load the file “captions.txt”, and one more time, use the files from Flickr8k Dataset to split the cleaned captions into a training and testing dataset. In the code the function used for this operation is the function *“get_captions_set()”*

At this end of this part, four datasets have been generated, two training datasets of 6000 elements : one with features and one with cleaned captions, and two testing datasets of 1000 elements : one with features and one with cleaned captions.

B.II Predict captions

First of all, we create a Tokenizer file from the training dataset with all the cleaned captions. This tokenizer is saved to the file “tokenizer.pkl” so it can be used later on in the testing part.

Then, for every element of our training set of captions :

- The original caption is saved in a list
- The selected model (vgg or our) is used to predict a set of words
- The tokenizer is used to generate a sentence from these words
- The final predictions is saved in a list

At the end of this part, we have two lists, one with the original captions for the testing set of images, and one with the predicted captions.

By passing these two lists as parameters of the function *“corpus_bleu()”* from the nltk library, the BLEU-Score is compute.

V. Results

A. Prepare_data.ipynb

Inputs :

- Link to **Flickr8k datasets**
- Link to the google drive where the files are saved.

Output :

- **features_vgg.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with the VGG16 model.
- **features_our.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with our own model.
- **captions.txt :** text files with all the cleaned captions, computed with the captions from the Flickr8k dataset.

It is impossible to check the integrity or efficiently of the features files alone. However, as the other results are coherent all along the project up until the final captions, we assume that they are fine.

For the captions.txt file, the first few lines are :

```
1000268201_693b08cb0e child in pink dress is climbing up set of stairs in
an entry way
1000268201_693b08cb0e girl going into wooden building
1000268201_693b08cb0e little girl climbing into wooden playhouse
1000268201_693b08cb0e little girl climbing the stairs to her playhouse
1000268201_693b08cb0e little girl in pink dress going into wooden cabin
1001773457_577c3a7d70 black dog and spotted dog are fighting
1001773457_577c3a7d70 black dog and tricolored dog playing with each other
on the road
1001773457_577c3a7d70 black dog and white dog with brown spots are staring
at each other in the street
1001773457_577c3a7d70 two dogs of different breeds looking at each other on
the road
1001773457_577c3a7d70 two dogs on pavement moving toward each other
1002674143_1b742ab4b8 little girl covered in paint sits in front of painted
rainbow with her hands in bowl
```

```
1002674143_1b742ab4b8 little girl is sitting in front of large painted
rainbow
1002674143_1b742ab4b8 small girl in the grass plays with fingerpaints in
front of white canvas with rainbow on it
1002674143_1b742ab4b8 there is girl with pigtails sitting in front of
rainbow painting
1002674143_1b742ab4b8 young girl with pigtails painting outside in the
grass
1003163366_44323f5815 man lays on bench while his dog sits by him
1003163366_44323f5815 man lays on the bench to which white dog is also tied
1003163366_44323f5815 man sleeping on bench outside with white and black
dog sitting next to him
1003163366_44323f5815 shirtless man lies on park bench with his dog
1003163366_44323f5815 man laying on bench holding leash of dog sitting on
ground
1007129816_e794419615 man in an orange hat starring at something
1007129816_e794419615 man wears an orange hat and glasses
```

And this is exactly what we expected. There are too many captions to check all of them, but these ones are cleaned and well format, so we can assume that the entire file have been processed.

B. eval.py

Inputs :

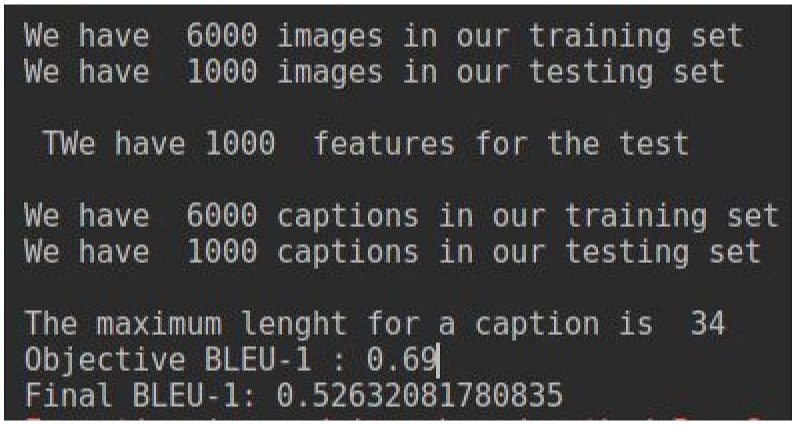
- **model_vgg_04.h5 :** file with all the weights of the caption generation model, trained on the features generated by the vgg model.
- **model_our_04.h5 :** file with all the weights of the caption generation model, trained on the features generated by our model.
- **features_vgg.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with the VGG16 model.
- **features_our.pkl :** pickle file containing the features generated from the images of the Flickr8k dataset, with our own model.

Output :

- **tokenizer.pkl :** pickle file, used to save the value of the tokenizer variable

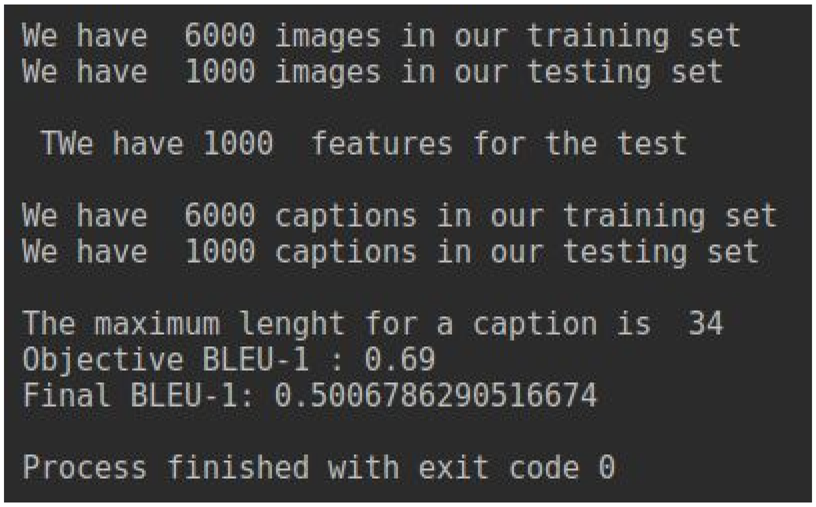
According to the article, the BLUE-1 score of an human is 0.7 in average (the closer to 1, the better). This was our reference while running the evaluation.

At the end, the results printed by the program with the vgg model are :



```
We have  6000 images in our training set
We have  1000 images in our testing set

 TWe have 1000  features for the test

We have  6000 captions in our training set
We have  1000 captions in our testing set

The maximum lenght for a caption is  34
Objective BLEU-1 : 0.69
Final BLEU-1: 0.52632081780835
```

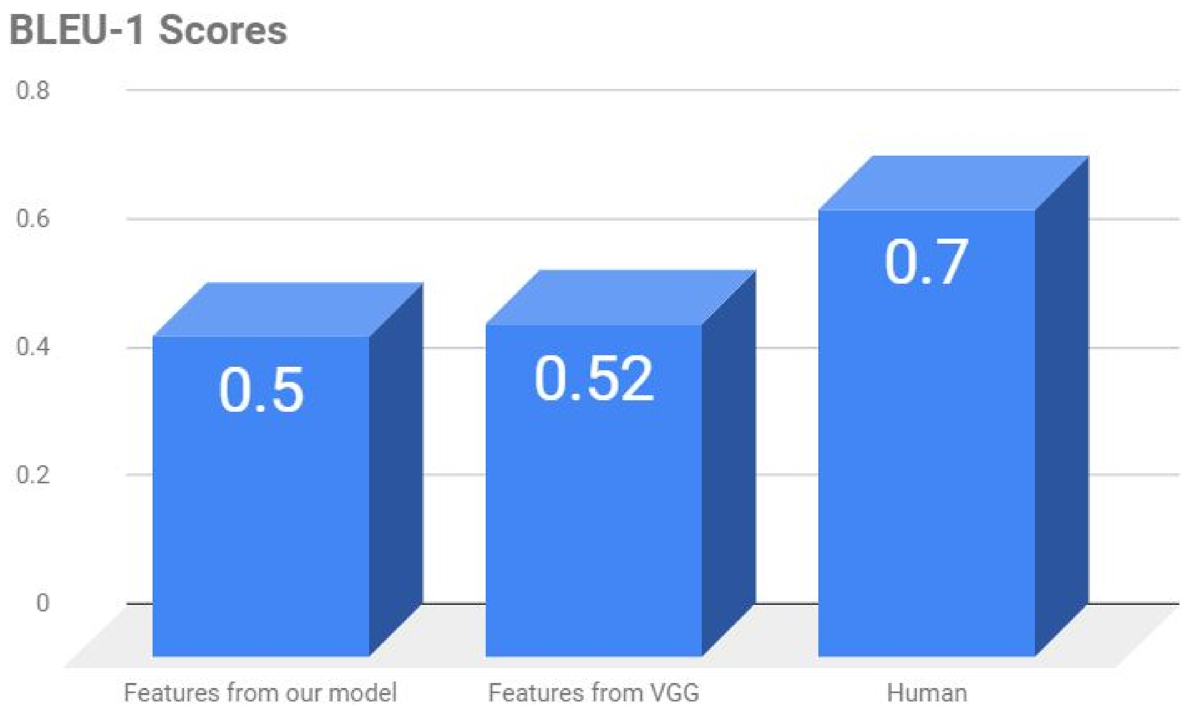
And the results printed by the program with the our model are :



```
We have  6000 images in our training set
We have  1000 images in our testing set

 TWe have 1000  features for the test

We have  6000 captions in our training set
We have  1000 captions in our testing set

The maximum lenght for a caption is  34
Objective BLEU-1 : 0.69
Final BLEU-1: 0.5006786290516674

Process finished with exit code 0
```

Also, the BLUE-1 score are :

- Human : 0.7
- Our system with features from the Vgg model : 0.52
- Our system with features from our own models : 0.50



This is great because our two BLUE-1 scores are high, close to the human, and the score with our own model is actually pretty close to the one with vgg.

VI References

Flickr8k Dataset :

The request to use the dataset has been made through the url :
https://forms.illinois.edu/sec/1713398
Our Webtools ID is 1713398.

We are hereby required to cite M. Hodosh, P. Young and J. Hockenmaier (2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artificial Intelligence Research, Volume 47, pages 853-899 http://www.jair.org/papers/paper3994.html

Publication :

- https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vinyals_Show_and_Tell_2015_CVPR_paper.pdf (our publication)

- M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. JAIR, 47, 2013.

- W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. In arXiv:1409.2329, 2014.

- K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: A method for automatic evaluation of machine translation. In ACL, 2002.

Codes :

- https://www.pythonforbeginners.com/basics/string-manipulation-in-python
- https://docs.python.org/2/library/tokenize.html
- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/generative_examples/image_captioning_with_attention.ipynb
- https://github.com/vsmolyakov/cv/tree/master/captions
- https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/
- https://colab.research.google.com/notebooks/io.ipynb
- https://www.quora.com/What-is-the-VGG-neural-network
- https://stackoverflow.com/
- Tensorflow documentation : https://www.tensorflow.org/api_docs/
- Keras documentation : https://keras.io/
- OpenCV documentation : https://docs.opencv.org/2.4/index.html