A SURVEY OF METHODS OF TEXT-TO-IMAGE TRANSLATION

D.O. Pydorenko
Supervisor - Ph.D., Associate Professor O. Turuta
Kharkiv National University of Radio Electronics
61166, Kharkiv, Nauky ave, 14, Software Engineering Department,
e-mail: daria.pydorenko@nure.ua

The given work is devoted to the text-to-image translation. This translation can be realized in two steps. First, a large text should be transformed into a small set of captions. Second, these captions should be used to generate images. There are different methods that can help to implement both steps.

Self-publishing becomes more and more popular. And many people create cover and/or illustrations to their own texts by themselves. So, it would be very convenient to automate the process of creating images by generating them to text using certain software. The text may be too large to convert it to a specific image. First, it should be shortened, for example, by finding a description (without dialogs), generating an annotation on the whole work or finding keywords in the text. So, the task can be divided into two main steps: the compression of the text in the best way for the next processing and the generation of the image from the first step output. And both steps should get a quality assessment of conversions. The image can be converted to the caption and compared with the input of the first step.

There are several algorithms to find keywords in the text such as TF-IDF, RAKE, and TextRank. The RAKE algorithm is based on the construction of a co-occurrence graph from the words that occur most often in the text.

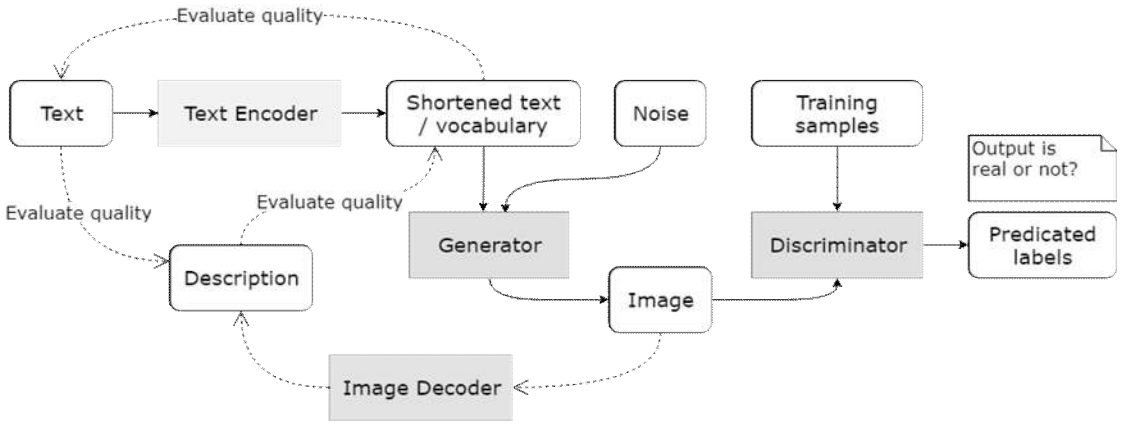
All these algorithms are represented by Python libraries. There is NLTK – a suite of libraries in Python for natural language processing (NLP) for English written text. One of the Python libraries which implements RAKE algorithm is multi-rake that helps to extract keywords from a text written in any language. They all work enough quickly but represents result in different ways. Multi-rake library should be configured to return word combinations instead of one word, because for the next step captions are needed as input. Also, the output doesn't always correspond to the most important part of the text, but the most repeatable motive. There is a problem with the language of the text on this step because texts written in English have more possibilities for their own processing.

Generation of images using neural networks is actively developing now. Different variations of Generational Competitive Networks (GAN) are used for that. GAN is represented by two networks - Generator and Discriminator, the first generates images based on the input; the second attempts to distinguish real images from generated and outputs a certain result (they are real or not).

A group of developers from the Lehigh University, Rutgers University, and Duke University and Microsoft have proposed their own solution for generating images – the Attentional Generative Adversarial Network (AttnGAN) allows multi-step change of the generated image according to the change of some

words in the text description. There is an implementation of AttnGAN on Python with PyTorch framework. But there are problems with the choice of a dataset with labels (the language of the labels, the number of images, the classes of images), and with the speed of training neural networks.

The whole process is represented in picture 1.



Picture 1 – Text to image conversion

The results below (pic. 2) were received after combining multi-rake algorithm and AttnGAN. The Last Leaf by O. Henry was chosen as a test input.

Picture 2 – Illustrations for The Last Leaf by O. Henry (1 – leaves, 2-3 – doctor)

The quality of generated images depends on datasets on which the neural network was trained. So, it is better to concentrate on a certain type of texts to improve the generation. For example, generate images of nature only and train on a dataset with such images. This may not be nature; any topic can be chosen.

The first step should be expanded with the extraction of only necessary parts of the text in this case. The text can be divided into paragraphs (or the sentences) and the number of words indicating the nature can be calculated (by the dictionary, by the classes of the dataset or its captions, etc.). Sequential paragraphs or sentences can be combined into one. All dialogs can be deleted before processing the text. If the number of words (or the percentage) exceeds a given threshold, the paragraph is sent to further processing. Thus, the program knows which input it expects and the output becomes more predictable and text-related.