Web App for Generating Image Descriptions Using NLP/CV Techniques

Adam Zucker[1], Dragomir Radev[1], Lawrence Staib[1]

[1]Yale College, New Haven, CT.

Introduction

This project involved the creation of a functional, deployable web application that automatically generates descriptions or “captions” for uploaded images. It consists of two parts, the web application that hosts and interfaces with the captioning mechanism and the computational engine that actually handles the image description generation.

Web Application

As is standard with all web development projects, the needs of this app dictated both its front-end and back-end design choices. The focus of this project was to be the caption generation engine, so the app was designed to be minimalistic, with the image uploader and captioned image components front and center. Besides these two components, the only other major component of the app was a document viewer, in which a user could view the project’s report, proposal, and abstract. The web app uses the Flask web framework (Python) for the back-end, ReactJS (JavaScript) for the front-end interactions, and ReactBootstrap (CSS) for the user interface styling.

Caption Generation Engine

The caption generation engine combines some existing CV and NLP engines, namely TensorFlow’s im2txt algorithm and Clarifai’s Image Recognition API, in a novel way to generate captions for images. The core algorithm (im2txt) is built on the *Show and Tell* Model, an encoder-decoder network trained on a large training set of pre-segmented and pre-captioned images called the COCO (Common Objects in Context) dataset. The encoder half of the system is a convolutional neural network that extracts the features from the provided image and encodes them into vectors. These vectors are then run through the decoder half of the system, which is a series of recurrent neural networks called long-short term memory (LSTM) neural networks. The LSTM networks decode the feature vectors into the words of the caption (the s_i terms in Figure 2) probabilistically, one word at a time.

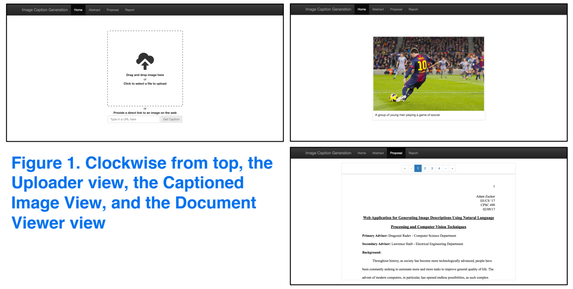


Figure 1. Clockwise from top, the Uploader view, the Captioned Image View, and the Document Viewer view

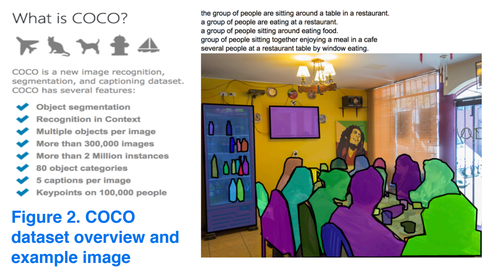


Figure 2. COCO dataset overview and example image

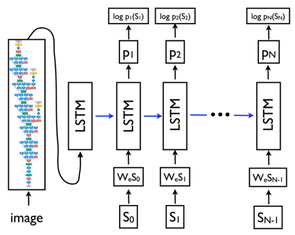
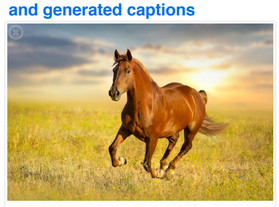
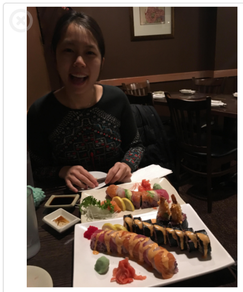


Figure 3. Show and Tell Model and generated captions


A brown horse standing on top of a lush green field


A woman sitting at a table with a plate of food


A boat floating on a lake next to a forest

Figure 4. Caption with misidentified feature


A white bowl filled with fruit on top of a table


A white bowl filled with berry on top of a table

Figure 5. Demonstration of hypernym-hyponym problem and solution

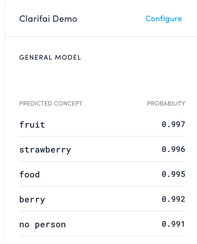
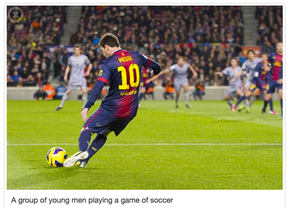


Figure 6. Sample Clarifai output


A group of young men playing a game of soccer

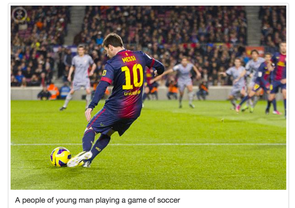

A people of young man playing a game of soccer

Figure 7. Demonstration of issues with hypernym-hyponym fix

Misidentification

The most common problem im2txt runs into is the misidentification of features. In Figure 4, for example, the algorithm seems to believe the person and/or rocks at the bottom constitute a boat, which is clearly not the case. This issue is probably the most difficult to correct, as it lies within the encoding part of the model.

Hypernym-Hyponym Problem

Another common but perhaps more readily fixable issue is that of the identification of “specialized objects.” This is apparent in the leftmost image in Figure 5, where the algorithm identifies “fruit” in the bowl, but not “strawberries” (the general term: “fruit” is a hypernym of “strawberry”, which is in turn a hyponym of “fruit”). The algorithm has no way of identifying a strawberry as a “strawberry,” because it cannot possibly identify something it has never encountered before.

Possible Fix

Clarifai’s API identifies specific objects, which it calls “concepts”, in an image. A post-processing algorithm can use this list of identified objects to attempt to replace non-specialized hypernyms with specialized hyponyms. However, this fix can create new issues besides grammatical agreement ones in the form of false positives. As a result of a proliferation of captions like these, this post processing was ultimately disabled in the final product.

Conclusion

This project has presented a user-friendly web application that can automatically generate captions for images. If nothing more, this project has been a fun attempt at tackling a challenging problem.

Acknowledgement

Thank you to Dragomir Radev and Lawrence Staib for their advising, as well as to the researchers behind Tensorflow, im2txt, and COCO