Machine Learning Approach for Data Annotation

Venkatesh Bhattad and Jemsoft Co.

Carnegie Mellon University

Abstract

The task of annotating an arbitrary data set is tedious. The objective of this project is to streamline and make the process of annotating data as seamless as possible. With the use of Machine Learning and the use of Google’s TensorFlow library, the outcome of this project is achieved. This project is developed as a web application using the Django framework of Python.

Keywords: Data annotation, TensorFlow, Django

### Problem Statement

Given an arbitrary unlabeled dataset, a user should be able to annotate data as easily and seamlessly as possible.

### Objective

The objective of this project is to develop a prototype tool to stream the process of Data Annotation. The scope of this project can be divided into two parts.

1. **Identify / Detect the appropriate data buckets**

The idea is to come with an application to input as many images as possible to classify them into the appropriate data buckets. Once the images are classified/labelled, they are then imported into the data bucket to enhance the data set. Therefore, increasing the efficiency of the application.

1. **Train / Re-train the model**

The idea is to develop a train / re-train the model given a situation that a set of images are currently not trained by the application. The user will have an option to label the data bucket and import as many images as possible to train the model.

### Implementation

As mentioned earlier, there are two parts in this application. The complete application is developed using the Model-View-Controller (MVC) framework of Django – a python based web framework. Additionally, to perform the data annotation and retraining the model, Google’s TensorFlow library is used.

1. **Identify / Detect & Label the appropriate data buckets**

The user inputs “.JPG” files as input. Upon submitting the form, all the images are taken as a list and stored locally in a folder called “uploads” in the project directory. Each of this image is then, given as input to the model to detect the appropriate data bucket. If an image is detected with a score of greater the “0.8” out of 1, then the image is moved from the uploads folder to the appropriate data bucket.

If an image is not detected by the model, then the list of undetected images is given by the user to retrain the model.

1. **Re-training the model**

The user is asked to input and label the data set. Then, the user is asked to import the .JPG images. Each of these images is then trained using the TensorFlow retrain API.

### Steps to run the project.

1. The requirements of this application are
   1. Python 3.5
   2. Django 1.8 *(pip install Django==1.8)*
   3. TensorFlow *(pip install tensorflow)*
2. Copy the project folder and *‘cd’* into the project directory.
3. Run the application by command “*python manage.py runserver 0.0.0.0:8000*”. This will run the application on the port 8000.

**Note**: Make sure to change the “*retrain.bat*” to “*retrain.sh*” in the ‘retrain’ function of the ‘classify/views.py’ file, if the host machine is Linux.

### Results

* Each time a user searches an image, the application becomes smarter. Every input increases the training set of the model.
* The user is given the flexibility to train the model.

### Future Scope or Enhancement

1. To implement a cronjob to run daily on the server to retrain the application.
2. To implement based on user. To have the model based on user specific data folders.

**References**

*"Tensorflow". TensorFlow. N.p., 2016. Web. 19 Dec. 2016.*

*"The Web Framework For Perfectionists With Deadlines | Django". Djangoproject.com. N.p., 2016. Web. 19 Dec. 2016..*