Cornell Tech CS5785 Applied Machine Learning Fall 2018 - Final Exam

Abstract

In this paper, we develop the algorithm for an image search engine. The search engine can retrieve 20 most relevant images from a large image dataset given a single five-sentence natural language query. We extract features from image's tag, description, two feature vectors from *pool5* and *fc1000* layers of the ResNet. Using the extracted features, we train a machine learning model to predict the description from image's tag and two feature vectors and compare the query with predicted image description to find 20 most relevant images.

1 Introduction

Our final solution is training a model which can generate a description vector using each image's tag and two feature vectors. Using this trained model, we can generate a description vector for every image. Then, we can find the closest image by calculating the distance between the query's word vector and the generated image description vector.

1.1 Feature Extraction

1.1.1 Image Tag

We create a bag of word for the training image tag and a bag of word for the testing image tag. Then we merge the two bags into one bag using intersection. The reason for merging into one bag is that we want a consistent bag for both training and test. The reason for doing an intersection instead of union is that intersection can greatly reduce the size of the bag which allows for a simpler model. We get a bag with size 102. The process for creating one bag of words and build a word vector for each image tag is as follows:

1. Create two dictionaries to record the word frequency in each tag (called term frequency and denoted as *tf*) and the total number of tags containing the word (called document frequency and denoted as *df*). The *tf* and *df* will be used later to calculate a weighted score for the bag vector.

2. Each image has several tags and each tag has two components separated by a comma. (category:object) Both category and object are counted as words in the bag. Category is useful as an additional information and can help with comparing the difference between images, so it should be kept as a word.

3. Lemmatize the word using nltk WordNet lemmatizer and further stem the word using nltk Porter stemmer. Lemmatizer can convert words in different format into one single format. For example, trees -¿ tree, likes -¿ like, ran -¿ run. The lemmatized word can increase the conversion rate for stemmer. The stemmer can convert words belong to different part of speech into the minimal stem form. For example, beautiful -¿ beauti, beauty -¿ beauti. Due to the language features, words are sparsely distributed and we want to combine closely related words to reduce the bag size.

4. Create a $1 \times N$ word vector for each image. N is the bag's size. The value for each term inside the bag is calculated using the following standard natural language processing rule.

$$x_{t,i} = tf_{t,i} \cdot idf_t$$

$$idf_t = log_{10}(\frac{len(D)}{df_t})$$

$x_{t,i}$: value in the word vector for term t and image i.

$tf_{t,i}$: frequency for term t in image i tag. Usually called as term frequency.

idf_t: weight for term t over all images. Usually called as inverted document frequency.

$len(D)$: total number of image tags.

df_t: total number of image tags containing term t. Usually called as document frequency.

1.1.2 Image Feature Vectors

We use all the given feature vectors. The feature vector from ResNet *pool5* has 2048 features. The feature vector from ResNet *fc1000* has 1000 features. We read the feature vector files using csv reader to extract the feature values.

1.1.3 Image Description

We do the same operations for the description compared with the tag. We create one single bag of word by intersecting the two bags from training and test description. The detailed procedures and calculation is the same as the tag. We get a bag with size 2776.

1.2 Model Training

We train a model that uses image's tag and two feature vectors to generate a description word vector. We simply concatenate tag (size 102), *pool5* feature vector (size 2048), *fc1000* feature vector (size 1000) into one vector of size 4050. Because there are 10000 training samples, the training X vector has size 10000*4050. The training Y vector is a description word vector of size 10000*2776.

We use ridge regression with alpha = 100 and other default parameters for sklearn. It runs pretty fast. Take about 10 seconds to train the model.

1.3 Model Prediction and Image Searching

After the model is trained, we use the model to predict a description word vector for each test image using the image's tag and two feature vectors. Then we get a description word vector for each test image. After converting the test query into a word vector, we can calculate the distance between the query word vector and all the test image's description vectors. We rank the distance in increasing orders and choose the 20 closest distance's images as the output.

2 Experiment

We try lots of experiments. We write a search engine that use a query to search the test's tags. It achieves the highest score of 0.169. In the search engine, we try bm25, language model, cosine similarity score calculation. We then try to train a model that can predict the relevant score between query and test image. The training relevant score is calculated by using distance by two feature vectors and tag vectors. We also try to use neural network, logistic regression, linear regression, random forest, kernel ridge, svr, svc, elasticNet, PLSR, PLSC, Lasso.

3 Results

We rank 4th on the public leaderboard with a score of 0.49199. The first group achieves the highest score of 0.51298. Our score is higher than all the groups' scores in the other section.