CS 5785

Dec 2017

Final Project: Images Search Network

[Abstract]

This project is to use the given training and testing data to design a model that outputs the top 20 most similar images in the given image dataset when the input is a file of five-sentence descriptions from the testing data. Techniques including Bag of Words, Multi-layer Perceptron Neural Network, and Principle Components Analysis have been used in the final model. To reach a relatively good output comparing to other previous tested models, an ensemble of the mixed methods mentioned have been implemented.

In the following sections, we discuss the models and procedures that included to reach our final prediction result in this sequence:

1. [Preprocessing]: Descriptions – Tags

2. [Neural Networks]: i) Preprocessing

– Word to Vec

– Distance Calculating Models: Bag of Words

ii) MLP Regression

3. [Dimensionality Reduction]: PCA

4. [Ensemble Model]: i) Descriptions – Tags + MLP

ii) Descriptions – Tags + PCA based MLP

[Preprocessing]

Score Calculator Depending on the connection between Description and Tag

In order to find the relation between descriptions and tags, we perform this preprocessing procedure. In the later mix model, we take this description – tag relation into consideration, and mixed it with the MLP model to predict the image ranking.

This preprocessing is not a model that aimed to train data but a data cleaning procedure.

The procedure includes the following steps:

1) From all the tags in both training and testing data, we collect different classes they have, and summarize them into a dictionary;

2) Then for each class in the dictionary, we summarize all unique words that has appeared in this class, so we have our complete dictionary for all the tags;

3) From a single description of 5 lines in the testing data, we first subtract the keywords from a single description (getting rid of the punctuations, stop words, and rewrite some of them, say ‘woman’ and ‘man’ were rewrote into ‘person’). Now, fit the cleaned description into the format of tag dictionary, so that we transform all the descriptions into tags, we denote them as “DTags”;

4) From each DTag we created, calculate the its score with each image’s tag by this formula:

$$Score = \frac{\#of\ DTag\ entries\ appeared\ in\ the\ Tag}{\#\ of\ entries\ in\ DTag} * \frac{\#of\ Tag\ entries\ appeared\ in\ DTag}{\#\ of\ entries\ in\ Tag}$$

The score will be in range of [0, 1].

***Mistake Summary:**

In this preprocessing step, we have tried a different way of calculating the score which is:

$$Score = \frac{\#of\ DTag\ entries\ appeared\ in\ the\ Tag}{\#\ of\ entries\ in\ DTag}$$

With this method, too many similar scores appeared. So, this method cannot provide sufficient ranking reference. We abandoned this.

[Neural Networks]

Multi-layer Perceptron Regressor

Since we want to find a predicted feature vector given a file of five-sentence descriptions, we are going to use a regression method. The reason why we are using MLP regressor is that the relationship between descriptions and image feature vectors is not linear. Since when the data has some nonlinear dependencies, neural network usually performs better than other regression method, MLP regressor was implemented in the model.

(i) Distance Calculating methods

During training the model of description-vectors learning predicting features and rank the distances, it involves a question of which distance calculation methods to choose. We have tried two different methods: 1) cosine similarity, 2) Euclidean distance. We quickly found out although these are different distance calculating techniques, the result of ranking the distances is very similar.

(ii) Frequency-counter + MLP

After preprocessing ranking, we then tested the ensemble model of frequency-counter and MLP. Since there are different parameters to try in the multi-layer perceptron neural network, we made several attempts to find a relatively satisfying output. What is more, there are two types of feature vectors given in the training data (one is 1000 dimensional features, the other one is 2048 dimensional intermediate features). Since the 1000 dimensional features is the output from the original image features from a neural network model, and the 2048 dimensional features is the layer before the output layer in the exact neural network model, it is reasonable to try to do MLP with both features vectors. Hence, we did several different combinations of feature vectors with different MLP parameters. The table below shows some of our trials of different combinations of feature vectors and parameters.

From the table below, we can tell that from the different combinations we have tried, having one hidden layer with neuron number 1,500 gives the best result.

Methods	Scores
Frequency-counter + fc1500 + 1000 nodes (1 hidden layer)	0.31285
Frequency-counter + fc100 + 1500nodes (1 hidden layer)	**0.34668**
Frequency-counter + fc1500 + 1500nodes + 1500nodes (2 hidden layers)	0.31891

Chart0. Mixed Model w/o PCA

***Mistake Summary:** Word2Vec

Word2Vec Model:

To prepare data for the neural network, we have to first translate text descriptions into numbers. There are three methods that came into our mind. First is to use the word to vector

(Word2Vec) developed by Google. The second one is to develop a vector that is either multi-hot or frequency-counter. And the last one is the frequency-counter word-to-vector methods. In other words, we instead of treating words that appeared multiple times in a description file, we also gave attention to the frequency of occurrence of the words.

We first tested the Word2Vec to find the best predicted feature vector. Then we compare the distance between this predicted feature with the filtered feature vectors provided by the preprocessing indexes. We quickly found that this method didn't give us an accurate enough result. Hence, we implemented the same logic with multi-hot counter word-to-vector model. However, the output was still not satisfying enough.

Finally, we tested the frequency-counter word-to-vector model. By combing this model with neural network, we started to have some relatively higher scores. Therefore, we chose to use this frequency-counter word-to-vector method to provide a baseline for further investigation.

Reason of failure:

Since Word2Vec is trained by Google, and the context is very likely to be different from what we are trying to do in this project. Google's word2vec applies better when we want to find associations between words. Google provided an example: *'vec('Paris') – vec('France') + vec('Rome') results in a vector that is very close to vector vec('Rome').'*

However, the description vectors we want is simply a conversion from text to numbers, nothing more. Hence, although Word2Vec has this nice nature of association, it might cause more noise when applied to our project. Multi-hot vector and frequency-counter methods, on the other hand, are designed specifically to train the model with feature vectors as output. Hence, without the noise of association, the later methods outperform Word2Vec in this project.

[Dimensionality Reduction]

PCA based MLP Regressor

Since there are too many dimensions in the feature vector: each intermediate feature with 2048 entries and each feature with 1000, the relative score calculated by cosine similarity will be greatly affected by noise. In order to better perform distance calculation given by cosine similarity, we decided to use PCA to reduce the dimensionality of the features data.

We have tried several different dimension values in PCA model to perform MLP regression. Please refer to the following chart:

PCA(n_components)	1,000	600	300	**150**	120
MLP score on Intermediate feature (dim = 2,048) after PCA	0.26366	0.28650	0.28760	**0.30444**	0.30123

Chart1. PCA based MLP predicting intermediate feature

PCA(n_components)	600	**100**
MLP score on feature (dim = 1,000) after PCA	0.24213	**0.25951**

Chart2. PCA based MLP predicting feature

Thus, we decided to use the model of PCA (n = 150) based MLP over intermediate features and the model of PCA (n = 100) based MLP over features to perform future mix of models.

[Mix Models]

Description-Tags Preprocessing + PCA based MLP Regression Model

We have tried several different coefficients of the mixed models that shown in the following chart:

	Intermediate feature	feature	Description-Tags	score
Mixed Model Ratio	1	1	1	**0.375**
	2	0	1	**0.39164**
	2	2	1	**0.37224**
	3	0	1	**0.39105**

Chart3. Mixed Model With Different Ratio

***Mistakes Summary:**

In transforming the feature data into lower dimensions, we have tried with training and testing data, respectively. But then, the results turned out to be very inaccurate. After printing out the dim-reduced vectors, we figured that the with different PCA model will return different transformation output. Thus, we retrained the training and testing data together in the PCA model to transform them so that to help them remain the same pattern as before the dimension reduction happened.

[Ensemble Model]

The collection of all the ensemble models we have tried are listed as the following:

Mixed Model Ratio

w/o PCA			
Intermediate feature	feature	Description-Tags	**Score**
1	1	1	**0.31285**
2	0	1	**0.33913**
0	2	1	**0.34426**
2	2	1	**0.34725**
w/ PCA nodes = 1,500			
Intermediate feature	feature	Description-Tags	**Score**
1	1	1	**0.3750**
2	0	1	**0.39164**
2	2	1	**0.37224**
3	0	1	**0.39105**
w/ PCA nodes = 2,000			
Intermediate feature	feature	Description-Tags	**Score**
2	0	1	**0.39206**
w/ PCA nodes = 3,000			
Intermediate feature	feature	Description-Tags	**Score**
2	0	1	**0.39749**
3	0	1	**The Highest**

Chart4. Mixed Model Summary

[Summary]

In this project, we have tried four different models, including preprocessing ranking, frequency-counter + MLP models, frequency-counter + MLP + PCA models. Within each model, we have tweaked different parameters in order to get better results. After many attempts, we have finally come to the mix model of frequency-counter + MLP (2000 nodes) + PCA (dimension =150). We ran MLP dimension 2048 feature vectors, mixed the model with a ratio of 1 (preprocessing): 2 (fc2048).