Image Search Engine

Abstract

In this paper, we solved the task of image search engine problem specifying in using natural language query to retrieve relevant images. We leveraged image features extracted by Residual Network (ResNet) combined them with natural language processing methods. We applied Random Forest to map query vectors to object tag space, Ridge Regression to map descriptions to image features and proposed an ensemble algorithm to blend our models. This paper gives a detailed description of data prepossessing, modeling and experiment results.

1 Introduction

Images are widely used nowadays. It has the advantage of visual representation and it is usually adopted to express other mediums. However, we cannot access or make use of this information unless it is organized to allow efficient browsing, searching, and retrieval. Thus, image retrieval has been an extremely active discipline in recent years.

Image retrieval techniques can be mainly classified into two categories: text-based image retrieval (TBIR) and content-based image retrieval (CBIR). Text-based image retrieval is about matching users' textual query to manually annotation of the images like keywords. Content-based image retrieval is based on matching similarities among features of an image including color, texture, shape and space relationship of objects etc.

1.1 Problem Definition

The challenge is to build an image searching engine using natural language query to discover relations between textual tags and image features and return the top 20 relevant images from a large database.

- Data
  10000 samples of 224×224 JPG images are provided with a list of tags describing the objects, a five-sentence description and feature vectors extracted from pool5 and fc1000 layer of ResNet.
- Evaluation
  For each description, the model is to output the name of top 20 relevant images. The system will evaluate based on the ranking of the correct image.

1.2 Related Work

There have been numerous previous efforts in image retrieval concentrating on various aspects.

Some commercial image search engines, such as Google Image Search and Yahoo Image Search, are keyword-based image retrieval systems [1]. Liet al.[2] proposed a TBIR approach that can effectively exploit loosely labeled Web images to learn robust SVM classifiers.

The progress of extensive study on content-based image research from the early 1990s to the early 2000s has been comprehensively discussed in existing survey papers [3][4]. Specially, two pioneering works have paved the way to the significant advance in CBIR on large-scale multimedia database. The first one is the introduction of invariant local visual feature SIFT [5] and the second one is the Bag-of-Visual-Words (BoW) model [6].

2 Model Architecture

2.1 Mapping Input Descriptions to Tags

When we read the problem, we notice that we need to use our model to find out relationship between sets of sentences and words, it is reasonable to use bag-of-words algorithm to do preprocess of all description and tags. Thus, our solution is:

For the description training set and tags, we constructed a bag-of-word for each and fit them as labels into a model to predict the tags for the description testing set. Then we compared the predict tags and testing tags to find the top 20 as the result output.

2.1.1 Bag-of-Words for Tags

Since provided tags wrote as "supercategory:category" and categories are mostly nouns, we simply processed tags by splitting with colon. Furthermore, we combined noun phrases into one words because it does not make sense to split phrases into two separate word, such as splitting "dining table" into "dinning" and "table". Then we used $CountVectorizer()$ function from $sklearn.feature\_extraction.text$ to help us generate bag-of-words with the frequency of the tag. There are 80 different kind of tags in total.

2.1.2 Bag-of-Words for Descriptions

Each image have 5 or more descriptions for the content. Unlike the format of tags, each description is a full sentence and need to so some preprocessing steps before we make the bag-of-words for descriptions.

The first step is to get bag of words for all meaningful words of decriptions. We lowercased all characters, eliminated all punctuation and stop words, got the stem of words and extracted nouns. Then we used $CountVectorizer()$ to generate bag-of-words with the frequency of the tag. The flow chart of word processing are shown in Figure 1.

Figure 1: Flow Chart of BoW Process

We notice that tags are all nouns. In order to make the model reliable, the second step is to extract all nouns from the bag of words for all kinds of words.

2.1.3 Random Forest Regression

Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The reason why we chose random forest trees is that it can deal with the image problems just like we did in Assignment 4.

We fit the $RandomForestRegressor()$ of the bag-of-words of tags and descriptions as matrices into the algorithms, and predict the tag bag-of-words for testing description. That is to say, we tried to predict the tags for the image associated with testing descriptions, and then use KNN to find the top 20 most relevant images.

Specially, we built one random forest for each element in the tag bag-of-words to get the probability for the predict tag containing in each element of BoW.

2.1.4 NN(Nearest Neighbors)

The final step is to find out the top 20 most relevant images. The algorithm we used for this step is Nearest Neighbors(NN).The idea of NN is to find 20 nearest neighbors, that is the most similar 20 images from the testing data. The distance is calculated by Euclidean Distance.

Fitting the NN model where 20 with prediction vector generated from random forest, the training output is a vector of file names for each entry. These files are computed as the top 20 similar images for each test image.

In all, the whole process is shown in Figure 2:

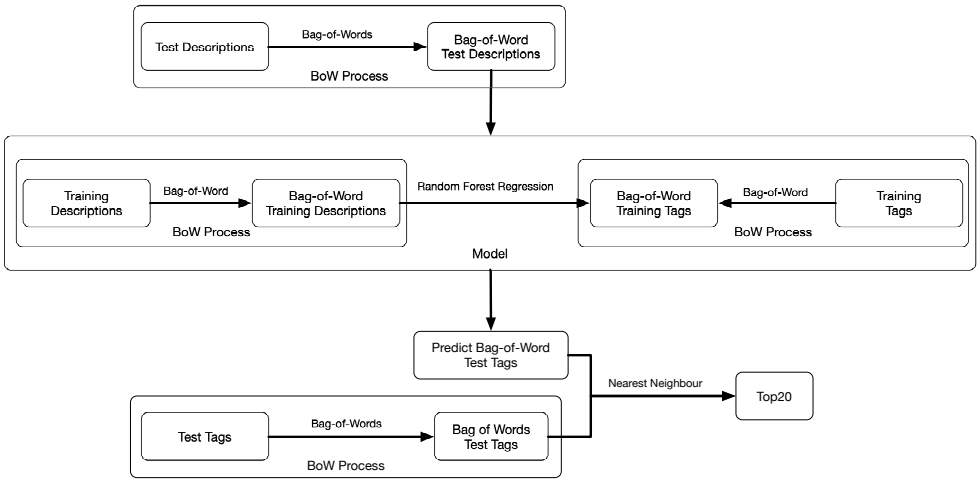

Figure 2: Flow Chart of Mapping Description to Tags

2.1.5 Results and Analysis

When we first used the tag-description model, the accuracy of the prediction is so bad that is less than 1%, which is really shocking.

We found that if the data size is too big the accuracy of random forest regression is really poor. Thus, we trained 80 different random forest regressor and each of them only use the bag-of-words of descriptions to predict the probability of only one type tag instead of training only one regressor and use it to predict all 80 tags. The results turned out far more acceptable, which is 20.5%, but it is not good, we need to find out why accuracy is so low and ways to improve the performance. We made three guesses from different angle:

- The first guess is that the dimension of bag-of-words of descriptions is so huge compared to the dimension of tags, so we used PCA to do the dimension reduction but the performance became even worse (15.9%). Obviously it is not the reason.
- The second guess is that we did not use the correct machine learning regressor to train the model, we also used Support Vector Machine(SVM) and LASSO regression, there is no improvement thus we change our guess.
- The last guess we made is that the dimension of bag-of-words of tags is so small that it cannot provide enough information for prediction, so we want to use feature (fc1000: 1000 dimension and pool5: 2480 dimension) to predict the top20 images. We will check correctness of this guess in the latter section.

2.2 Mapping fc1000 to descriptions

For this part, we need to build a model to find out relationship between fc1000 (1,000 dimensional feature from classification layer from Residual Network) and descriptions.

We constructed a bag-of-word for description and then fit the algorithm with features and BoW of descriptions to form a mapping between them. After that, we figured a solution to find the top 20 relevant images.

2.2.1 Bag-of-Words for Descriptions

Each image have 5 or more descriptions for the content. Since the description is a full sentence and need to so some preprocessing steps before we make the bag-of-words for descriptions.

The first step is to get bag of words for all meaningful words of descriptions. We lowercased all characters, eliminated all punctuation and stop words and got the stem of words. Specially, we extracted nouns and adjectives from descriptions for the reasons that fc1000 mainly capture the objects in an image and the objects are usually expressed by combinations of nouns and adjectives. Then we used $CountVectorizer()$ to generate bag-of-words with the frequency of the description.

2.2.2 Ridge regression

We still want to use random forest regression at the very beginning, while cause the huge number of data, the random forest regression takes a long time to train, in order to deal the problem quicker we select another algorithm.

Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity, where least squares estimates are unbiased but their variances are large so they may be far from the true value. Ridge Regression adds a degree of bias to the regression estimates to reduces the standard errors.

Then we fit fc1000 and BoW of descriptions as labels into Ridge Regression model to construct a mapping between them. Specially, we built two models for mapping descriptions to features and mappings from features to descriptions. In another words, we tried to figure out with which prediction variable we could get a higher accuracy result.

2.2.3 Result and Analysis

We tried two different models totally. For the first model, we fit fc1000 and BoW of descriptions as labels into the model to predict the features for the testing set and we got an accuracy of 20%.

The process of the first model is shown is Figure 3, here we simplified the bag-pf-word process to make the chart easier to read.

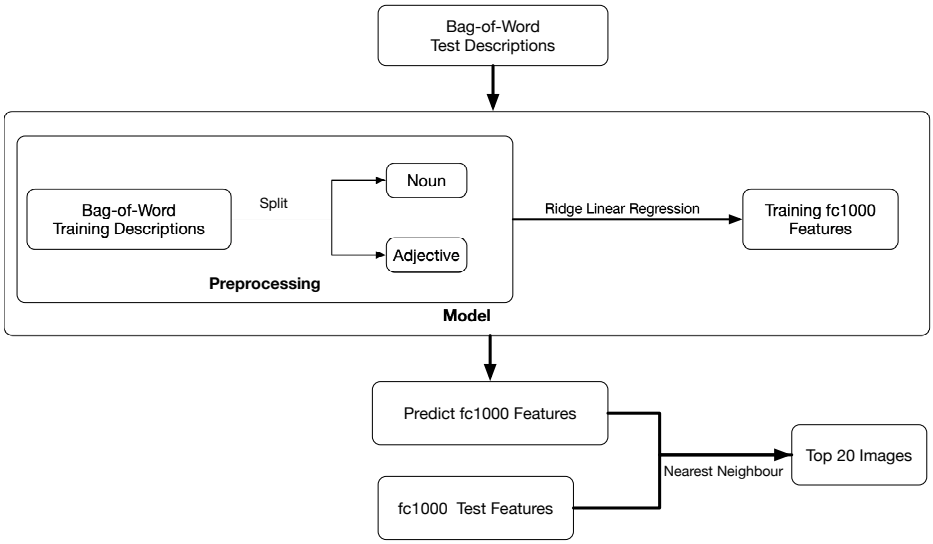

Figure 3: First Flow Chart of Mapping fc1000 to Description

The results is not as good as we thought, we do not know whether this is because we choose another model, but we guessed another reason for that is the BoW of descriptions is a sparse matrix with a much lower information density which makes the mapping from description to features less accurate even though they have more dimensions than fc1000 features. Thus, we changed our solution again:

We built the second model, we fit fc1000 and BoW of descriptions as labels into the model to predict the description for the testing set and we got an accuracy of 27%, which is much better than the previous model.

The changed process is shown in Figure 4:

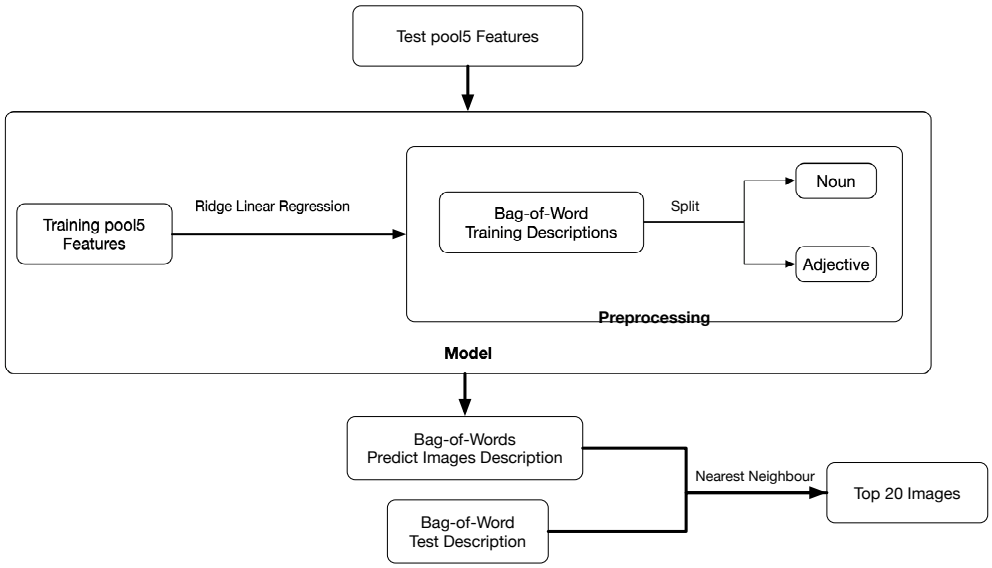

Figure 4: Changed Flow Chart of Mapping fc1000 to Description

The results proved our guesses, and tell us the fact that Ridge Regression is good to deal with the problem.

We thought the reason for the model mapping from features to description works much better than that mapping from description to tags is that fc1000 have a higher dimension and are more reliable since they are generated by machine learning algorithms.

To make the accuracy higher, we made a few attempts. We found there are a number of nouns in descriptions with a very low frequency which makes a little contribution to the prediction result but will increase . Thus, we decided to eliminate there low-frequency word and generated a new BoW of descriptions and trained the model again and the accuracy reached 32%.

For the second attempt, we noticed that the words with a very high frequency in descriptions would also affect the result accuracy since the vast majority of English words will not appear in most of the reviews, most of the feature vector elements will be 0. We need a postprocessing or normalization strategy that combats the huge variance of the elements in the feature vector. Thus, we tried the log-normalization method for postprocessing and reached an accuracy of 35%.

We also used $RidgeCV()$ with **cross validation** to select parameters and get the best accuracy which is 38%.

After that, we tried other algorithms to fit the model, such as Random Forest and LASSO. However, the result showed that Ridge Regression works the best. Now we are sure that Ridge Regression is the best method to deal with the problem.

2.3 Mapping pool5 to descriptions

For this part, we need to build a model to find out relationship between pool5 (2,048 dimensional feature from final convolution layer from Residual Network) and descriptions.

We constructed a bag-of-word for description and then fit the algorithm with features and BoW of descriptions to form a mapping between them. After that, we figured a solution to find the top 20 relevant images.

2.3.1 Bag-of-Words for Descriptions

For this part, we used similar prepossessing methods for descriptions as before, thus, we also chose Ridge Regression. However, we extracted verbs, nouns and adjectives from descriptions since pool5 mainly captured the actions of objects in an image. Then we used $CountVectorizer()$ to generate bag-of-words with the frequency of the description.

2.3.2 Ridge regression

In this part, we fit pool5 and BoW of descriptions as labels into Ridge Regression model to construct a mapping between them. Because the pool5 is like fc1000 which is a dense matrix, thus we choose the same idea as the changed model from previous section:

In all, the whole process is shown in Figure 5:

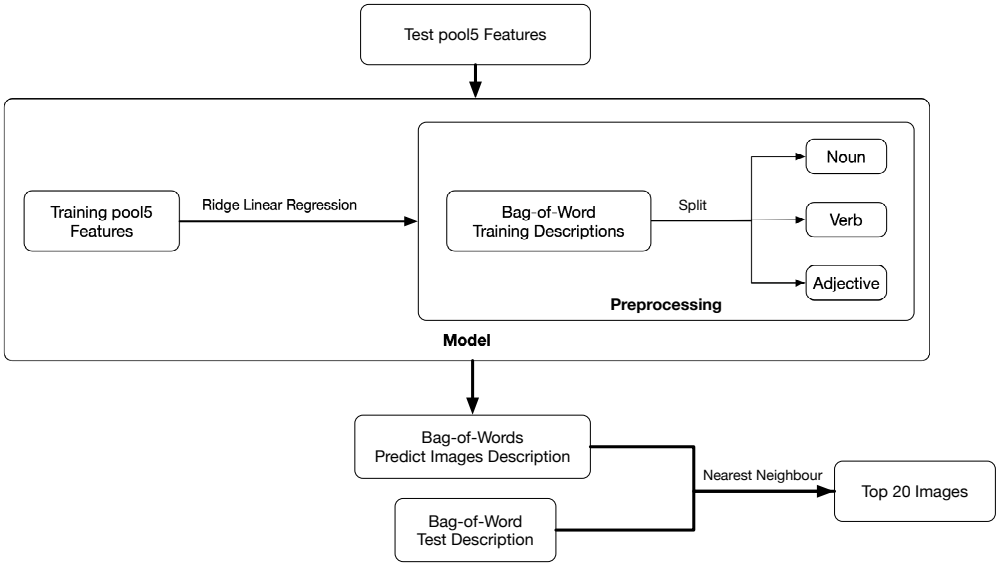

Figure 5: Flow Chart of Mapping pool5 to Description

2.3.3 Result and Analysis

Using this model, we fit pool5 and BoW of descriptions as labels into the model to predict the description for the testing set and we got an accuracy of 42.9%.

We thought that the reason why performance of pool5 is better than fc1000 is that the dimension of pool5 is far more than fc1000,(2480 vs 1000) so it may contains more useful information than fc1000.

2.4 Ensemble model

With three different models, we need to find a way to combine them together in a way that improve the overall performance.

The results from the three models are items ranking by distance from small to large. Since these distance were computed upon different model, it could be not reliable to ensemble the result directly. Thus, we first took the prediction vector from three models before Nearest Neighbour and calculated the Cosine distance to generate a new lists of distance. Then we tried several methods to merge these items.

2.4.1 First Attempt - Simple Averaging

The first attempt we took is simply averaging the distance and then rank again to get a new list of top 20 items. The result showed it did not improve the accuracy.

2.4.2 Second Attempt - Weighted Averaging

Then we guess to solve this problem, instead of simply averaging everything, we need to assign weight to the results from different results. This time the result turned out to be slightly better: 43.9%.

2.4.3 Third Attempt - Merge and Rank

The third attempt is merge and rank the result distance together and return the top 20 of all. However, the accuracy did not improve.

2.4.4 Result and Analysis

From the attempts above we can see that ensemble method can give the accuracy some improvement with right weights, though it took a lot of time to get the right weights and the improvement is really small but it is worth to do so.

3 Results

The best accuracy is 43.9%, using the result we can check the correction. Here, we randomly pick decryption from test description and list the top 10 images, the number under the images is the indexes of the images:

Ex1 Description:

A person on skis is in the air.

There is a person jumping in the air on a pair of skis.

The person gets airborne while skiing down the slope.

The skier is doing a trick in the air.

A skier is mid air on the slopes.

Top10:

(a) 1216 (b) 1660 (c) 1855 (d) 1815 (e) 429

(f) 1320 (g) 655 (h) 1122 (i) 1701 (j) 1790

Figure 6: Top 10 Images

Ex2 Description:

A small white boat sits on some calm water under a snow capped mountain.

A boat on a still lake in front of a mountain

a white boat on some water and some mountains

The boat sits in the lake below the mountain.

A lone boats sits on a lake that is adorned with mountainous reflections.

Top10:

(a) 1767 (b) 148 (c) 986 (d) 618 (e) 925

(f) 218 (g) 852 (h) 85 (i) 802 (j) 325

Figure 7: Top 10 Images

The summary of methods we used and accuracy are shown in following table, the method which is in bold font is the method we recommend and used (because there are 2 model which accuracy are very close).:

Table 1: Different Models and Accuracy

Methods	Accuracy(%)
$BoW_{description}$-> Random Forest Regression -> BoW_{tag}	20.525
$BoW_{description}$-> Ridge Regression -> fc1000	20.423
fc1000-> Ridge Regression -> $BoW_{description}$	35.760
pool5-> Ridge Regression -> $BoW_{description}$	**42.971**
Average for All 3 Models (tags, fc1000, pool5)	30.534
Weighed Add for All 3 Models (tags, fc1000, pool5)	**43.933**

4 Conclusion

In this paper, we designed an image search engine using natural language query to retrieve rank-order 20 relevant images. Here are our findings during the whole process.

- It is really useful to do some preprocess steps (like selecting specific types of words like noun, adjective or verb; lowercase; stop words; stem, etc)
- With proper parameters, we can combine different models and get a better prediction of the results. Cross validation is a useful way to select good parameters for the models.
- Although the dimension of the bag-of-words data is large, while cause it is sparse, the information density is less than dense matrices like fc1000 and pool5.

- Ridge Regression performs really great in dealing this kind of problem which is multi-colinearity problem.
- The model become more reliable when there are more features add into the model, this can be proved from our result. (Accuracy: Tags < fc1000 < pool5)

We have select the most reasonable and performed model as we can cause the limit of time, but we know that there may be some better way we can deal with the problem. Though the final project ends, we can keep learning more machine learning algorithm to make the model performs better.

Acknowledgments

The authors would like to show gratitude to Serge Belongie for his guidance ans instruction. And we thank 3 "anonymous" reviewers for their insights and their comments on an earlier version of the manuscript.

References

[1] J. van Gemert, "Retrieving images as text," 2003.

[2] W. Li, L. Duan, D. Xu, and I.W.-H. Tsang, "Text-based image retrieval using progressive multi-instance learning," Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2049–2055

[3] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 2, no.1, pp. 1–19, 2006.

[4] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," Pattern Recognition, vol. 40, no. 1, pp. 262–282, 2007.

[5] D. G. Lowe, "Distinctive image features from scale invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[6] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in IEEE Conference on Computer Vision and Pattern Recognition, 2003, pp. 1470–1477.

[7] www.github.com

[8] https://mlwave.com/kaggle-ensembling-guide