A Large-scale Image Search Engine

Jingxuan SUN (js3422), Yue WANG (yw986), Jingxuan ZHANG (jz926)

December 7, 2018

Abstract In this paper, we addressed the challenge of image search problem specifying in using natural language query to retrieve related images. We utilized image features extracted by ResNet and combined them with natural language processing methods. We employed Random Forest, 1dCNN, SVC to train and map description vectors to tag space, PLS regressions, LSTM neural network to find a sub-space for description vectors and image features. And finally proposed an ensembled model to best capture the multi-feature information.

Keywords: Image search, Natural Language Processing, Random Forest, PLS

1 INTRODUCTION

This project is CS5785 Applied Machine Learning Final Project, and the challenge is to build an image search engine system based on natural language query. In the final, system would be given a short sentence and output top 20 related image based on ranking against a pool

of 2000 candidate images.

Fundamentally, our challenge is first to build a text pre-processing function to process the description and tags. And these outputs would to fed into a set of models to train and figure out the underlying relationship between descriptions and features (both fc1000 and pool5), descriptions and tags. The reason here why we do not consider relation between tags and features would be discussed in feature observation section. Subsequently, and ideally, since different model captures different relations, we ensemble the set of models based on their weights, similarity, accuracy to capture the most relevant output in our test set.

Another rationality we consider here is that, we can not reverse the approach by mixing up the train image and test image. For example, generating a new description of each test image by finding similar image in train set and build description on those train image's description. This could achieve a higher accuracy, but when evaluating a search engine, we are supposed to use a trained model to predict based on test data only.

1.1 PROBLEM DEFINITION

Data Format:
Train set: 10K images of size 224x224, with corresponding fc1000 (1000 dimension) and pool5 (2048 dimension) features extracted from ResNet. For each image, a 5-sentence description is given, also tags information in format of [category : subcategory].
Test set: 2K images of size 224x224, with corresponding fc1000 (1000 dimension) and pool5 (2048 dimension) features extracted from ResNet. For each image, tags information in format of [category : subcategory] is given. 2K 5-sentence description is given to evaluate the performance of search engine.

Feature Observation:
For this task, we are given redundant correlated features. Each raw image is related with a raw description, a Resnet pool5 layer feature vector, a Resnet fc1000 layer feature vector

and a list of tags. The relationship between these features we believe is as follows: fc1000 feature is retrieved based on the pool5 feature, each element of the fc1000 feature represents the activation of an ImageNet class, those activated classes are reflected in the tags. Thus, we come to the conclusion that tag information should be equivalent to Resnet features in connecting descriptions with images. Therefore, we started out from using description to predict tags, to using description to predict Resnet features, then finally tried out an ensemble of the two.

Evaluation:
For each description, students are to submit 20 candidates, and the system will evaluate the results using the MAP@20 metric, returning the score based on the ranking of the correct image.

1.2 Related Work

At very beginning, we split the work to review papers on potential models both in machine learning and deep learning field.

A very natural approach is to project description on image features and find the underlying relation between the features and description, then we could generate each image in the test set a new description based on our trained model. Considering the relation underlies might not be linear, we explored techniques in deep learning field, and experimented on the feasibility of applying long-short-term neural network proposed by [1] O'Reilly. This paper mainly talked about how to generate captions based on a raw image.

One thing that bothered us very much was how to introduce the image features to come into play. Intuitively, at query time we are only accessible to descriptions, and the only thing we can do with descriptions is to retrieve tags which are also textual information. But as mentioned in feature observation, tags are very high level abstraction of the Resnet features,, especially the pool5 features, and we believe that Resnet features will offer us more detailed

and holistic information about the images. As raw descriptions are also more detailed and accurate representation of the images, we wanted to connect descriptions directly with Resnet features.

After some literature review, we found that the better way of relating one source of data to another in a cross-modal setting is doing Partial Least Square, PLS. PLS will try to find the multidimensional direction in the X space that explains the maximum multidimensional variance direction in the Y space (projection on to latent structure). The idea is similar to PCA except that it models the relation between two variables. In PLS Regression, PLSR, it also maximizes fit and minimizes misfit while maximizing correlation between X and Y[4].

2 DATA PRE-PROCCESSING

We mainly used the NLTK library for pre-processing text data. Specifically, we removed stopwords and then used stemming and lemmatizing techniques to break the descriptions down into meaningful uniformed English words.

The specific lemmatizer we used is WordNetLemmatizer, which will return words in the WordNet corpus. Unlike lemmatizers, which will always produce a dictionary word, stemmers removes prefixes and suffixes with no regard to the resulting string's semantic validity. The stemmer we used is LancasterStemmer.

After this step, there are two kinds of approaches that we took to tokenize and extract useful words: Bag-of-words and TF-IDF. Before this process, we also deal with the special case of un-connected word, for example, skate board and skateboard, by using approach similar to 2-gram to detect repeated.

For Bag-of-Words, we first computed a dictionary for every description that records the

number of occurrences of each words and then summed all the dictionaries together to get a global dictionary counting the number of occurrences of each word in the entire training set.

TF-IDF is used to calculate the weight of each word using the frequency of the specific term and the inverse document frequency, which can tell us the importance of a certain word in the context of all the documents.

In addition to pre-processing the text data, we also sorted the pool5 and fc1000 image features according to their corresponding image ID. And in the Random Forest Model, we take another step to refill missing tags for some images, which will discuss later in this paper.

3 MODELING APPROACHES

3.1 LIST OF MODELS USED

Following table address how we utilize different model to capture underlying relation between different features.

	Description -> Tag	Description -> Resnet	Ensamble
Multilabel Classification	OneVsRest with Logistic Regression		
	Logistic Regression		
	Linear SVC		
	1dCNN		
Multitarget Regression	Random Forest Regressor	PLS Regression with noun and fc1000 200c	Linear SVC + 2PLSR
	Random Forest Regressor with Tag Refilling	PLS Regression with noun and fc1000 400c	Linear SVC + Random Forest Refilling + 2PLSR
		PLS Regression with all and pool5 400c	Logistic Regression + Linear SVC + Random Forest Refilling + 2PLSR
		PLS Regression with all and pool5 2048c	
		LSTM for description generation	

3.2 Description to Tags

3.2.1 1D-CNN

We experimented with 1-D CNN in an attempt to predict tags from the query description and then use the predicted tags to find the most similar images. The structure of the CNN consists mainly of an Embedding layer, a Dropout layer, a Conv-1D layer, and a Dense layer.

As the input of the CNN is the descriptions, we first need to pre-process the training descriptions into a correct format. To achieve this, we used Tokenzier from keras.text package to tokenize the fie descriptions corresponding to each training image using the top 5000 frequent English words, resulting in a dictionary of words and their index. Then, we mapped the descriptions to words in the dictionary into sequences. After observation, we decided to limit the length of each sequence to 30, padding zeros to the beginning if the number of significant words is less than 30.

During training, we encoded the list of sub category tags to a binary list, reflecting if a single tag appears in the list. After training the CNN, we fed pre-processed testing descriptions to the model and got a ranking of the most probable tags associated with the description.

To get the sets of tags that are most similar with the predicted set of tags, we compared the predicted tags with each set of tags in the test data and computed a score for each set. If a predicted tag of higher probability appears in the compared set, then the score of this set will be increased by a larger margin. Otherwise, the score will also be decreased by a larger margin. After we compared all sets of tags in the test data, we will choose the images corresponding to the top 20 similar sets of tags as the final result.

3.2.2 MULTILABEL-LOGISTICS REGRESSION

We fit logistic regression models to predict tags from description then found the top 20 similar tag vectors as a representation of top 20 similar images.

As tags are discrete, we were able to transform the problem into multi-label classification. There are in total 80 subcategories and each image is related with several of them. We encoded the tag vector in a bag-of-word fashion and set up one logistic regression model for each tag. Then we fed the TFIDF feature of the training descriptions and corresponding training tag features to the ensemble of 80 logistic regressors to fit the model for tag vector prediction.

At query time, we passed each TFIDF feature of test description into the model and achieved according predictions of tags. To get the most similar images, we get the most similar tag vectors matched with the images we want. To get the most similar tag vectors to our prediction, we fit the KNN model with test tag vectors and retrieve the top 20 neighbors.

3.2.3 MULTILABEL-LINEAR SVC

Although using SVM and logistic regression is just one-line difference in code, the motivation of trying SVM is totally different. As SVMs will provide us the largest margin between two classes, we though it would be more robust as the model is applied to test set. The pipeline is still predicting tags from descriptions and finding the top 20 most similar test tag vectors.

3.2.4 RANDOM FOREST REGRESSOR

As an approach to project description to tags, it is important to pre-process them in a proper and refined way. How we defined "proper" is that data should be as complete as possible, also the input and label information should be equal, for example, both in "noun".

When inspecting the tags information, we found out that many images has zero corresponding tag information which makes a big loss in our model. So the first step we take here is to refill tags. Refilling tags involves three steps:

1. Sort the top 10 important features in image's fc1000 features. This is because fc1000 captures the object detection information on images, and as tags, which should also be in noun.

2. Fetch the corresponding semantic meaning of specific feature. This is done by linking tag information from imageNet with fc1000 features.

3. Conduct NLP preprocess on the new tags we retrieved. After all, we have a new bag of words of tags which size is 200, bigger and conduct more important information than original 80.

After pre-process and compress our input and output on descriptions and refilled tags, we fed them into random forest regressor. The consideration behind this is that, our input has pretty large dimension, which is 200, so random forest could do feature selection on it. In our implementation, we conduct one regressor on each tag 2K image, and append them

to form the final output as a 2000x200. After generating the tags corresponding to the test description, we conducted KNN to classify the top 20 similar result.

3.3 Description to Resnet Features

3.3.1 PLS Regressor

The pipeline for PLSR models is as follows. We fit the models with TFIDF of training descriptions and Resnet features of the training images. At query time, we let it predict the Resnet feature of a TFIDF feature of the test descriptions. Then we used KNN to find the top 20 most similar test Resnet features.

We experimented with 4 settings of the model:

- with nouns TFIDF of description, predict fc1000 feature, using PLSR with 200 components
- with nouns TFIDF of description, predict fc1000 feature, using PLSR with 400 components
- with all words TFIDF of description, predict pool5 feature, using PLSR with 400 components
- with all words TFIDF of description, predict pool5 feature, using PLSR with 2048 (all) components

The reason we only used nouns in the first two settings is that fc1000 features correspond to ImageNet classes which are noun phrases. We used all words for pool5 features because they are more detailed and accurate representation of the visual characteristics.

The result confirmed our intuition. We achieved a higher mAP@20 rate for pool5 prediction on average. Please refer to the result section for more statistics.

One thing to notice is the cost-efficiency trade-off. Although the PLSRs had a lead of accuracy from the naive logistic regression and linear SVM, the training time of PLSR til convergence was significantly longer, depending on the dimension of the data.

3.4 Ensemble Methods

As mentioned before, different models captures different underlying logistics between different type of features. So we kept exploring on how to ensemble them to find the best fit, which could used both the image features and text features.

We hypothesized that different models performs better on different inputs. Intuitively, we want to calculate the weights, which is distance in all our model. And this is our final solution to ensemble: For each model, we trained them based on our previous pipeline. After predicting all the potential tag vector, we did pairwise distance calculation on the 2K image. Thus we will be given a 2000x2000 distance matrix for each model. Multiply with the weight of each model based on their performance, we defined a multi-model system to capture all important and relatd information.

3.4.1 Results

Following is the corresponded result for each model we implemented. As clearly shown, combination of PLSR, SVC, Random Forest and Logistics Regression has highest score.

Model	mAP@20
Naive matching	0.13
1dCNN	5.892
Logistic Regression from Description to Tags	16.293
Linear SVC from Description to Tags	21.073
Random Forest from Description to Tags with Tag Refilling	26.226
PLS Regression with noun and fc1000 400c	19.214
PLS Regression with all and pool5 2048c	28.097
2PLSR + Logistic Regression + SVC + Random Forest	37.248

4 Conclusion

In this paper, we addressed the challenge of image search problem specifying in using natural language query to retrieve related images. We utilized image features extracted by ResNet and combined them with natural language processing methods. We employed Random Forest, 1dCNN, SVC to train and map description vectors to tag space, PLS regressions, LSTM neural network to find a sub-space for description vectors and image features. And finally proposed an ensembled model to best capture the multi-feature information with a satisfied mAP@20 score.

5 Acknowledge

We would like to thank Prof. Serge Belongie, for teaching us the fundamentals and the TA teams.

REFERENCES

[1] Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern

[2] Keras Documentation https://keras.io/

[3] Vens, C., Struyf, J., Schietgat, L. et al. Mach Learn (2008) 73: 185. Decision trees for hierarchical multi-label classification https://doi.org/10.1007/s10994-008-5077-3

[4] Abdi, H. (2010).2: 97âĂŞ106 Partial least squares regression and projection on latent structure regression (PLS-Regression). Wiley Interdisciplinary Reviewshttps://doi.org/10.1002/wics.51

[5] Anjali Ganesh Jivani et al, Int. J. Comp. Tech. Appl., Vol 2 (6), 1930-1938 A Comparative Study of Stemming Algorithms