CS5785 Applied Machine Learning
Final Report

Team Anonymous

December 2017

Contents

Abstract

We have built an description to image retrieval program in the form of natural language processing. By experimenting with combinations of various types of data inputs and five learning models, we successfully improved our image search accuracy score to 43.79%. This report is dedicated to discuss the formulation of our final model, which consists of pre-processing the images as well as descriptions and combining predictions from 3 models to determine the relationship between a image and a description.

Chapter 1

Introduction

Search engines have been assisting us in finding solutions and the contents we look for. With the blooming age of information and multimedia, social network platforms, we have been exposed to a wide range of data and the complexity of information. Currently, research is indicating that the market is in seek of a more powerful search engine for users to retrieve their target results more precisely. Customers are increasingly expecting a more intuitive way to search. Using natural language descriptions to find an image is one of the most important implementations.

Previously, image searching engines heavily relied on key words. However, keyword-based inputs has significantly limited expression from human users, who are more familiar with making natural language inquires. At the same time, searching by using sentences normally degrades the effectiveness of searching. To respond to such problem in the real world, recent researches and advanced algorithms have discovered more powerful solutions to improve the searching performance. In the following chapters, we have tested several methodologies associated with building a sentence based image search and formulated a final model.

Chapter 2

Data and Experiments

Data

Input:
We assumed following data sets are known:
A list of tags indicating objects appeared in an image.
Two layers of feature vectors, which are outputted by ResNet, a convolutional neural network from an image that represent the elements in it.
A five-sentence description, used to train and test our search engine.
A set of corresponding images.

Output
Return 20 images given a five-sentence description, ranked by their closeness to that image.

Experiment

Tools
Python Package: nltk(natural language processing), numpy, pandas, nltk.stem, matplotlib, scikit-learn

Implementation
Below is an overview of the approaches the team has tried to find the best result. More details about each step are explained afterwards:

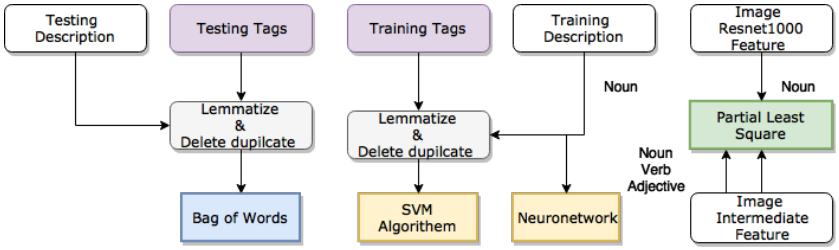

Figure 2.1: Methods and Data Sets flow

Step 1 Lemmatize and Extract Key Words

Based on the observations of the data sets, we notice that the tags and descriptions contain nouns and verbs which are not uniformed in a single form, so the first important step is to lemmatize the data set: which include all the data sets other than the image data.

There are several well-performing libraries to choose from, we decide to apply NLTK, natural language processing package of Python. Since the propriety is to extract and lemmatize all the nouns from the description which corresponds to the images. For the tags data set we have generated, there are two categories: for each individual tag, on the left side of the colon is the major category, and on the right side is the specific description for objects in the image.

Step 2 Mapping Input Features to Our predictions
During the mapping process, we have applied a couple of approaches:

2.1 Mapping descriptions to tags

2.1.1 Bag of words

First, we simply map the words in the description to the tags space in order to get the predicted tags, given testing descriptions. In the process of implementing the method, we noticed that from the description data, there exists a few words that are not matched to the tags space. At the same time, the frequency of the words appeared in the description cannot map to the tags space. To give an example of this implementation, by ranking the top words, for some words like "man" and "people" from description, we can map those into "person" in the tags space; similarly, "trees" and "flower" can be

mapped to 'plant'. By going through each tag, we give a weight of 1 for an existing tag mapped in the description, and a weight of 0 for non-existing tag.

$$\text{Description} \Rightarrow \text{Tags}$$

$$10000 \left\{ \begin{array}{c} \text{"food","cake" ...} \\ \text{.....} \\ \text{"ski","man"...} \end{array} \right\} \Rightarrow 10000 \left\{ \begin{array}{c} \text{0 1 0 0 .....0} \\ \text{.....} \\ \text{0 0 0 1 .....0} \end{array} \right\}$$

Diagram 2.1.1 Mapping result using bag of words

2.1.2 Support Vector Machine

One thing very good above using this method is that we do not have to process the words in the description too much, because the classification of SVM would automatically take care of the relationship between description words and tags words. In fact we have 91 tags in total, so we need to train SVM model for all the tags and get the prediction tag features. The mathematical expression as below:
Given a training data set of n vectors $(X_1, Y_1), ..., (X_n, Y_n)$, and find the hyper plane.

$$\vec{w}\vec{x} - b = 0$$

Set the distance and separate the data into two sets at dimension

$$\begin{gathered} \text{i for } 1 \leq i \leq n \\ \vec{w}\vec{x} - b \geq 0 \; Y_i = 1 \\ \vec{w}\vec{x} - b \leq 0 \; Y_i = -1 \end{gathered}$$

2.1.3 Neural network

Due to the complexity and the amount of data we are encountering, we then experiment with the neural network algorithm, which has been extensively used in the industry. The structure of our classifier contains 1 layer of hidden nodes.

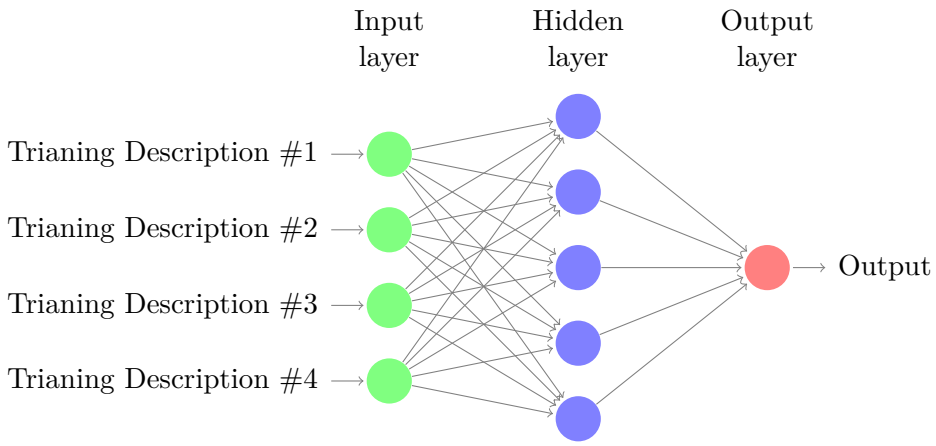

Figure 2.2.2 Single Hidden Layer Neural Network Structure

2.2 Mapping image features to descriptions

2.2.1 Partial Least Square Regression

As one of the common regression models, we applied PLS to match description to the provided features from the csv files. In other words, we process the description from the text, and generate a feature data file using our fitted PLS model, then the final result come from the Euclidean distance between two feature sets.The benefit of the model is that it usese Principal Component Regression to reduce dimensionality, and it is good at finding correlation between multidimensional data spaces.

The basic algorithm is described as below:

$$X = TP^{\mathrm{T}} + E$$
$$Y = UQ^{\mathrm{T}} + F$$

2.2.2 Random forest

We also experimented with random forest algorithm, in seek of an optimal learning model. However, due to the high-dimensional data, the training of random forest was extremely slow and the outcome is not as good as expected. Therefore, we switched to PLS for more accurate prediction.

Step 3 Measurements of Similarity between Predictions and Existing Features

3.1 Compute the norm of difference

Take the norm of two vectors, which represents the difference of two vectors. The smaller the number is, the closer these two vector matches together. In total, we tried l1 and l2 norm, and l2 norm performs better in giving a fuller representation.

$$NormSim = \|X - Y\|^2$$

3.2 Compute the cosine similarity

Between two vectors, the cosine similarity give the way to compare if two vectors' projection are overlapping on certain dimension. Plus, the real benefits of applying cosine is that a normalization process is conducted before comparison, which is a clever way to reduce noise, the mathematical expression can be represented as the following equation. However, we did not use this method as the norm distance returns better results.

$$CosSim = \frac{X*Y}{\|X\|\|Y\|}$$

Step 4 Ensemble Models

4.1 SVM and Neural Network

We tried directly blend SVM and Neural Network together by weighting there predicted tags feature. Then use the combined tags feature to compute the top 20 min distance from test tags. This approach slightly improved performance up to 10 percent. This way of blending two models is essentially combine the predicted features.

For model give different feature vectors, we cannot use the previous methods anymore. So we compute the distance for each model's prediction features and the desired features. After, get the distance (Here we implement the l2 norm distance), we give different weights to distance of each model based on their individual performance.
The final prediction would be make based on the blend distance we compute.

Chapter 3

Results

Algorithm	Accuracy	Mapping
Linear SVM	%22.4	From: Description To: Tags space
Bog of words	%13.7	From: Description To: Tags space
Neural Network	%22.7	From: Description To: Tags space
Neural Network+SVM	%24.1	From: Description To: Tags space
Random Forest	%9.8	From: Image feature To: Description
PLS$_{rednet1000}{}_{compo400}$	%33.2	From: Image feature To: Description
PLS$_{rednet2048}{}_{compo300}$	%36.7	From: Image feature To: Description
$PLS\_{rednet1000}{}_{compo400}+$ $PLS\_{rednet1000}{}_{compo400}$	%39.2	From: Image feature To: Description
$PLS\_{rednet1000}{}_{compo400}+$ $PLS\_{rednet1000}{}_{compo400}+$ $SVM + CNN + BoW$	%43.8	$From : ImagefeatureTo : Description$ $From : DescriptionTo : Tagsspace$

Observations and Highlights

From our result we can see that, the best effective single model in application is the PLS$_rednet2048_compo300$ model, and mapping image feature to descriptions using random forest isn't satisfying. When we are trying to map descriptions to tags, Neural Network out performs SVM approximately 10%, but the BoW method's performance couldn't be seen as the optimal choice in such case.

Chapter 4

Discussion

After gaining the result listed in the above chapter, we are moving on to some high lights and learning from the different approaches. At the beginning of our trials, we identified the most direct way is to match tags, since it reduced complexity and seem more intuitive, similar to looking up dictionaries.However, this method is not performing as powerful as we have anticipated, the tags can precisely give "variables", but the imagine retrieval has been affected the by the large amount of details omitted through reduction.

After realizing this issue, we started to train the text description in order to receive feature vectors, mainly by apply bag of words, and the result has been improved significantly. By then, we moved on to PLS, SVM random forest and neural network in processing 1000 image feature vectors with descriptions.

Among the methods above,random forest is the least effective, it is resulted by the problem of feature reduction, it is hard to obtain principal variables without accurate feature reduction. Due to the limitation of random forest, we integrated this approach in mixing models.

The optimal approaches we have is using bag of words in extracting nouns from 1000 image feature vectors, more precised result can be be obtained by processing intermediate image feature vectors, and expand the words to both nouns and verb.

Chapter 5

Conclusion

In the process of trying to solve the sentence image searching algorithm, we have gained a better understanding of analyzing natural language processing problems from different perspectives, identifying the key attribute can be the most challenging part of the problem.

Next, finding the right model and combining the models requires not only the theoretical background but also repeated hands-on experiments. In the future studying, we will continue to look into the reason of why PLS is the best for sentence to image retrieval question.

References

Pei-Lan, L., Lin, S. S. J., Chuen-Tsai, S. (2013). Effect of reading ability and internet experience on keyword-based image search. Journal of Educational Technology Society, 16(2)

Meng, W. (2015). A sentence-based image search engine (Order No. 10006834). Available from ProQuest Dissertations Theses Global. (1762244278).