Image Retrieval System Using a Combination of Text-based and Content-based Methods

Team name on Kaggle: Above Average

Abstract

We are challenged to predict relevance images given the descriptions. We have designed and experimented with both text-based and content-based methods to complete the task. This paper will introduce the TF-IDF model, Logistic-KNN model, and Image-to-Description reverse model. We have found the the connection between image features are significantly stronger than that between text descriptions, and the best-performing method is the Image-to-Description reverse model with an accuracy score of 0.244 on Kaggle.

1 Introduction

This competition challenges us to retrieve target image from natural language descriptions in sentences. We are given a testing image set and train image set, the convolutional and classificational feature vectors for each image, sentence descriptions for training images, and textual tags for both training and testing images.

Image retrieval techniques can be generally categorized into two major research areas: Text-based/Concept-based image retrieval and Content-based image retrieval, in terms of how users make queries to the image database, and in which form the images information is stored.

In Text-based image retrieval, images are annotated with a textual description (Tags) and their retrieval is based on matching the user's textual query (Descriptions) to the annotation of the image. In Content-based image retrieval, image searching and matching is based on the intrinsic features of images, for example, colors, textures, structures, etc. Text-based image retrieval requires considerable amounts of pre-annotated tags or captions of images, which is hard to obtain. Content-based image retrieval need to leverage information of the image itself, which usually calls for more complex models and larger computational capability. In this paper we present a series of different approaches of image retrieval using a combined methods of Text-based image retrieval and Content-based image retrieval techniques.

1.1 Text-based Image Retrieval using TF-IDF model

One of the fundamental bases of text-based image retrieval is to define the similarity between a textual query and a annotated textual description, and eventually search for the most related annotated image. Measuring the similarity between documents and queries has been extensively studied in information retrieval. Various methods can be used in this phrase, including cosine similarity, TF-IDF, word2vec, etc. In terms of short documents, TF-IDF is a relatively suitable and widely used method.

TF-IDF stands for "Term Frequency-Inverse Document Frequency", and the TF-IDF weight is a statistical parameter which oftenly used for evaluating the importance of a word in a given document collection.

For each term t which appears in the document, the TF-IDF weight is composed by two terms, and are defined respectively in the following way:

- TF (Term Frequency) (t): (Number of times term t appears in a document) / (Total number of terms in the document).
- IDF (Inverse Document Frequency) (t) : log_e(Total number of documents / Number of documents with term t in it)

The final TF-IDF weight is simply the product of the above two terms. As we can observe, the importance of a word increases proportionally to the number of times this word appears in the document but is offset by the frequency of the word in the corpus.

After computing the TF-IDF weight for each word that appears in a given collection of documents, we are able to construct a vector for each short document, recording the TF-IDF weights of all words that appear in this document, and eventually we can use the vectors constructed to compute similarities between documents. More details of our implementation will be explained in the experiment section.

1.2 Content-based Logistic Regression-KNN model

Considering the fact that the input query will only be text descriptions, we think it will be helpful to establish a connection between text inquiries and image feature vectors. We plan to train the model through fitting training description to the train classification feature vectors using logistic regression. Then predict the classification feature vector for each test description, and use KNN method to find the closest 20 images with similar classification feature vectors to return.

The textual description data can be very sparse, so we plan to use NLTK toolkit to preprocess the text through stemming and replacing synonyms in order to boost the intensity of data. In addition, we decide to only take high-frequency words from the descriptions to reduce noise.

Furthermore, the logistic regression requires the Y to be labels, which needs to be lower dimensional representation of the 1000-dimension feature vectors. So we plan to use PCA to dimensional-reduce the feature vectors selected by ResNet.

1.3 Image-To-Description Reverse Model

Similar to previous models that matches the closest images in training data given the test description, the Image-To-Description Reverse Model works in a different order (almost reverse order). First we preprocess the description using methods like stemming and taking high-frequency words like we did in other models. Then we gather training images that has the most similarity to each test image through comparing their classification layer of feature vectors. After that, we compute an aggregated description from these training images found, and use this description to retrieve target testing image. Finally we output the top 20 images using the test description from all the estimated description of the entire test data.

2 (Background/Related Work)

2.1 Similarity Score Calculation Between English Words

We tried to leverage the WordNet Interface (http://www.nltk.org/howto/wordnet.html) for computing the similarity between words so taht we can use that for computing the similarity between descriptions. However the runtime of that is too large to complete within a reasonable amount for a reasonable complexity of words retrieved from descriptions. So ultimately we have not retrieved an expected result from this source.

3 Experiment

3.1 Text-based Image Retrieval using TF-IDF model

Since the queries we will get from users in our image retrieval system are a series of descriptions of a certain image, an intuitive thought is to use the description obtained to search for the most relevant images tags in the database, which calls for text similarity measurements. Here we choose TF-IDF model as it contains not only information from one description text, but also the whole collection of all testing descriptions, and it's relatively suitable for short documents.

But before we start to compute the TF-IDF value for each words that appear in a description document, it is essential to notice that the description was written in natural language, which contains a considerable number of stop-words and punctuations, which makes it hard to process, and diffuses the core information contained in the text. Therefore, it's essential to pre-process description documents into a small amount of most important and meaningful words which can well represent the whole text.

During the pre-process phase, the first remove all punctuations and converted all letters into lower cases, then we use the stoplist imported from the NLTK module to efficiently remove all stopwords that appear in the text. Next, we use the whole corpus of the test description data to construct a dictionary, recording all words that appears in the corpus. Then we can convert the processed descriptions corresponding to a certain image into Bag-of-Words. Eventually we use the results obtained from above the construct the TF-IDF of the description corpus and compute the TF-IDF value vector for each description file.

For each tag file in the image database, we do a similar pre-processing step to extract essential words, and fit each processed tag file into the TF-IDF model that we constructed in the last step, using the description corpus. Thus we can compute the TF-IDF array for each tag file.

After obtaining the TF-IDF vectors for all test descriptions and tags, it's intuitive to compute the similarities between vectors. Here we simply use cosine similarity, and eventually obtained a similarity matrix of all test descriptions and tag files.

With the similarity matrix obtained above, we can easily select the top 20 most similar tag files related to each description file, and return the corresponding images to users.

3.3 Content-based Logistic Regression-KNNmodel

We first read the feature vectors from the .csv files and sorted them according to the filenames they belong to. Then we have read descriptions, lowercased all the words, stripped the punctuations, removed stop words, stemmed all the words using NLTK, and took 7 most frequent words from each description file.

After that we read the content part of all tags; built a dictionary based on the tags, and made bag-of-words feature vectors for both train and test descriptions. We decided to build dictionary from tags instead of descriptions because tags only describe objects, which are supposed to have stronger correlation with classification feature vectors.

Then we used PCA to reduce the dimensionality of descriptions to make them distinguishable by the logistic regression classifier.

Finally, we trained the logistic regression classifier by fitting the bag-of-words vectors to the dimensional-reduced feature vector data. We found the 20 images with feature vectors that have closest l2-distances to the results generated by our classifier for each test description, and submitted the result to kaggle.com.

3.4 Image-To-Description Reverse Model

We experienced that the relevance between descriptions are far less significance than that of image features. So in the end we decided to flip the order. Here are the steps:

- Map test image features to top k images features in the training images. The score are computed using L2. We use the features of fc1000 layer as this is better for matching object level words.
- Gather a top m frequent words from the k matching training images. Note the words are retrieved through lowercasing, stopwords-filtering and stemming.
- Generate a description vector for the test image as an estimate of its description.
- Perform KNN with N=20 on the test description vectors to generate top 20 relevant images.

4 Result

Here is a quick recap of the different models:

Model	**Kaggle Score**	**Notes**
Simple Bag-Of-Words Model of Test Description and Tags	0.148	This model is a good base result for reference. But is far from optimal since there is lots of data unused.
Description-to-Image-Vectors KNN Model	0.12	This model is worse than expected. The relevance between descriptions is not as good as expected. But this failure has helped us trying to reorder a little bit and came up with Image-To-Description Model in row 4.
Description-to-Image-Vectors Logestic Regression Model	0.0988	The accuracy is lower than the simple bag-of-words KNN model. This is probably because each dimension of the feature vector represents a specific content object. Reducing the dimensionality ruins the representations of the feature vectors generated by ResNet.
Image-To-Description Reverse Model	0.244	The image features relevance is higher than that of descriptions. So we expect the score of this to be higher than using description to find images.