Image Retrieval with Text Query

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

We address the problem of retrieving images with text queries in this work. We formulate it as a regression problem from image features to text features, and show that even with the simple and efficient kernelized ridge linear regression, one can achieve highly competitive test-time performance. Despite its simplicity, our trained model's MAP@20 score is 0.5259 on the unseen Kaggle test set, making our method one of the top-ranking solutions on the public leaderboard.

1 Problem Formulation and Analysis

Given a query text, our goal is to retrieve the most relevant image(s) from a image database, or in other words, to rank the images in a database according to their relevance with the text. This is a well-established problem in both academia and industry; search engine companies like Google have been offering such services as part of their product for quite some time. In this work, we tackle a smaller-scale version of this image retrieval problem. Despite being small-scale, our problem shares the core challenge as their large-scale industry counterpart: cross-modality between images and texts.

Images and texts are information from different domains. Although humans find it easy to tell whether an image and a text are relevant or not, it is very hard for a machine to do so, mainly because images and texts lie in different feature spaces, and no off-the-shelf distance metrics can be directly applied to measure their irrelevance. Hence one feasible solution is to embed images and texts into the same feature space, in which the Euclidean distance, cosine distance, etc can readily serve as the irrelevance metric. We establish our solution on this principle.

To embed images and texts into a common feature space, one first needs to decide what feature space to use. The common feature space can be that of the images, or that of the texts, or a third one other than these two. For example, if we choose the image feature space, then essentially the text features need to be translated into vectors in the image feature space, in a way that they are close to their relevant image features. This can be formulated as a regression problem from text features to image features, in the most straightforward sense. However, image features are typically of much higher dimension than text features, which means that regressing image features from text features could potentially suffer from a lot of ambiguities. Hence we take the opposite way and regress text features from image features.

We have also considered and tried embedding both the images and texts jointly into a shared feature space, but eventually gave up this method, due to the fact that our new embedding learned from the limited available data might not preserve the structure of the original text feature space in which semantically-similar texts are close to each other. This can severely hurt the generalization of our image retrieval system, as the query texts that retrieve the same image can take different forms but have similar semantic meanings.

Besides cross-modality, we think that another issue also makes this problem very challenging: partial labeling. In this task, we are provided with a bunch of images and their associated query texts to build our image retrieval system. In contrast to the image classification task where each image usually has one unique correct label, there can be multiple reasonable query texts for an image in the training set. But the training data only gives us one single query text for each image, which implies that the labeling in our training set can be partial and incomplete.

2 Method

As mentioned in previous section, we have formulated this image retrieval problem as regressing text features from image features, followed by ranking the images in a database according to the distances between their regressed text-space features and the feature of a query text. We give details of our method in this section, e.g., the image and text features we adopt, and the regression models we choose.

2.1 Image Features

Resnet models have achieved extraordinary success in computer vision tasks like image classification, semantic segmentation, etc. We use the pretrained Resnet models in PyTorch. For each RGB image in the training and test set, activations of the last two Resnet layers are extracted as features. Feature dimensions for these two layers are 2048, 1000, respectively. Indeed, there are multiple Resnet variants. We extract a 3048 dimension feature vector for each image with five Resnet variants, and concatenate them to get a single feature vector. The five Resnet variants we use are ResNet-50, ResNet-101, ResNet-152, ResNeXt-101-32x8d, Wide ResNet-101-2 listed on the PyTorch pretrained model page [4].

2.2 Text Features

Given a text, we can get its feature representation by taking the average of the word2vec vectors for each individual word. Alternatively, the bag of words (BoW) model can be utilized. In practice, we learn a regressor with each text represention being the objective, and aggregate all the regressors' predictions at test time.

2.3 Regression Models

We use an ensemble of kernelized ridge linear regressors. For each chosen text feature representation, i.e., averaged word2vec and BoW, we train a regressor that predicts a feature vector in the same text feature space given an image feature. At test time, we then calculate the Euclidean distances between the feature of a query text and the regressor's predictions for each image in the test set, which yields a distance vector. The distance vectors from different regressors are averaged to obtain the final distance vector, and the test images are ranked according to it, with smaller distance implying higher relevance with the query text.

3 Experiments

Our training data consists of 10k (RGB image, text) pairs; each pair contains an RGB image and a relevant query text. The test set has 2k such pairs hosted on Kaggle, which is not available to us.

To prevent overfitting and tune hyper-parameters, we split the 10k training examples into two parts: 8k for training, 2k for validation. Once we finish tuning the hyperparameters, we re-train our machine learning algorithm on the whole 10k training examples, and report its generalization performance with MAP@20 on the unseen Kaggle test set.

3.1 Early Experiments

Baselines: Text-to-Image Regression

The baseline solution provided by the TA is a ridge regression model from text features to image features. Its MAP@20 score is 0.1141 on the public leaderboard. We made some adaptions to this baseline to see how far we can go along that direction.

LightGBM Baseline

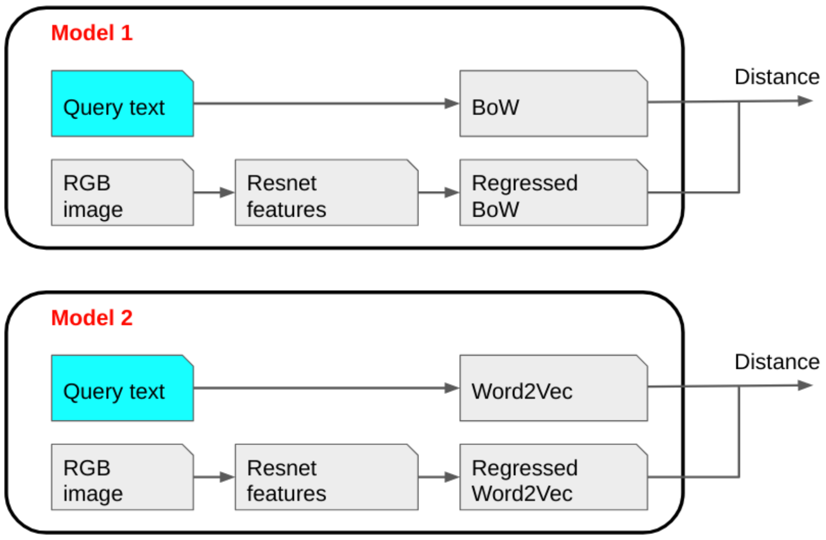

Figure 1: Overview of our method. We use two text representation methods, i.e., BoW and Word2Vec, and train a kernelized ridge linear regression model for each representation with the image features being input. During test time, each model gives us a distance measure between a query text and an RGB image in the test set. The distance measures are then combined to obtain the final result.

We first replace the ridge regression model with tree-based ensemble learning algorithms. We use LightGBM [2] in our implementation. It's a gradient boosting framework that boasts better computational efficiency than the well-known XGBoost. The MAP@20 reported by Kaggle for this model is 0.1422.

Cosine-based LightGBM Baseline

To further improve the performance, we tried several distance metrics such as L1 and cosine. Among these metrics, the cosine distance metric leads to slightly higher MAP@20 score on the Kaggle test set, which is 0.1857.

3.2 Our approach: Image-to-Text Regression

Variant of Cosine-based LightGBM Baseline

As mentioned in the 'Problem formulation and analysis' section, the drawback of using text features to predict image features is that there can be multiple mappings from low dimension data to the high dimension one. To tackle this problem, we propose a new approach to train model, which is using Resnet image features as input to predict text features. With this simple re-formulation, we are able to push the MAP@20 of the "Cosine-based LightGBM baseline model" in previous section to 0.3458.

Ensemble of Kernelized Ridge Linear Regression

By cross validating, we find that ridge linear regression with the RBF or polynomial kernels outperforms the other models. The kernel trick makes the regression algorithm more adaptable to non-linear patterns in the data. In addition, the kernelized ridge linear regression is extremely fast to train and tune, even if the dimension of both the input and output feature vectors is high. We use self-generated Resent features, BoW tag features as input, and train two regression models to predict both word-to-vec embedded texts and BoW encoded texts, as shown in Fig. 1. We use both the RBF and polynomial kernels, and end up with four trained regression models. At test time, given a query text, we ensemble the results from these four models by combining their induced distance vectors with equal weight, and ranking the images according to the combined distance vector. Kaggle shows that we manage to accomplish a MAP@20 of 0.5259 on the public leaderboard.

Methods	MAP@20
Baseline provided by the TA	0.1141
LightGBM baseline	0.1422
Cosine-based LightGBM baseline	0.1857
Variant of cosine-based LightGBM baseline	0.3458
Ensemble of Kernelized linear Ridge Regression	0.5259

Table 1: Performance comparison of different methods on the Kaggle test set.

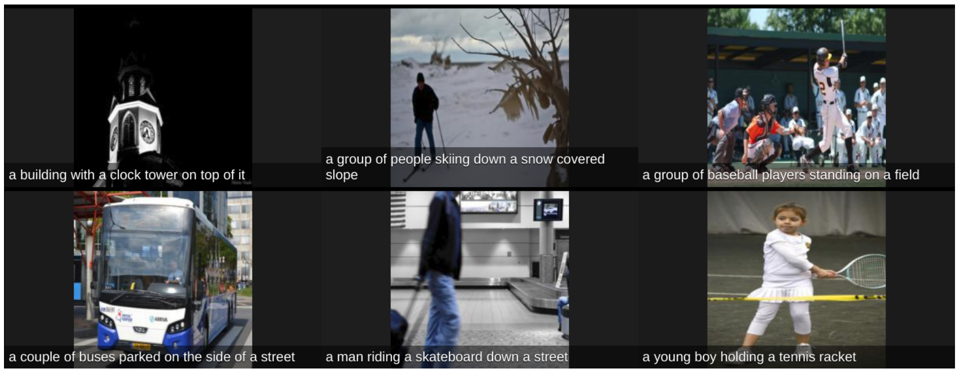
a building with a clock tower on top of it
a group of people skiing down a snow covered slope
a group of baseball players standing on a field
a couple of buses parked on the side of a street
a man riding a skateboard down a street
a young boy holding a tennis racket

Figure 2: Example captioning results on the training data by using [5].

4 Conclusion

In this work, we first formulate and analyze the problem of image retrieval with text queries. We then solve the problem with a simple and fast kernelized linear regression model. Our method has been demonstrated to generalize very well on the unseen Kaggle test set, and is one of the most competitive solutions on the public leaderboard at the time of writing.

5 Future Work

Speaking of representing an image with a text (feature), one interesting direction is to caption each image first with existing off-the-shelf image caption models like [5], and then perform image retrieval solely by comparing the query text with image captions. During our experiments, we made a few attempts along this line (some captioning results are shown in Fig. 2), but the results were not good. But we believe that this is a very promising direction to go, since captions and query texts are both natural languages.

Another interesting extension to our work is that instead of formulating a regression problem, one can employ the learning-to-rank framework [1] to explicitly optimize the ranking performance. This is more aligned with the evaluation metric, i.e., MAP@20, than simply regressing, and can potentially outperform our current method by a significant margin.

6 References

[1] **Wiki page on learning to rank** https://en.wikipedia.org/wiki/Learning_to_rank

[2] **LightGBM gradient boosting framework** https://lightgbm.readthedocs.io/en/latest/

[3] **XGBoost gradient boosting framework** https://xgboost.readthedocs.io/en/latest/

[4] **PyTorch pretrained models** https://pytorch.org/docs/stable/torchvision/models.html

[5] **Discriminability objective for training descriptive captions** Luo, Ruotian and Price, Brian and Cohen, Scott and Shakhnarovich, Gregory. arXiv preprint arXiv:1803.04376