CS 5785: APPLIED MACHINE LEARNING

FINAL EXAM REPORT

December 8, 2017

Anonymous

Cornell Tech

Contents

DESCRIPTION TO IMAGE MATCHING

ABSTRACT

Matching search text to image results is a common use case of Machine Learning in image search. We constructed various statistical models to create mappings of text descriptions to images. For a given search query, these models output images sorted by some measure of the level of predicted similarity to the text. We explored various representations of the text data, different solution frameworks and experimental design to train our algorithm to accurately output image queries.

INTRODUCTION

Image retrieval involves indexing, storing and retrieving digital images using input search text or image. Typical approaches involve using ?context? around the images, such as tags, keywords or descriptions to retrieve images. There are also content-based image retrieval (CBIR) systems that use computer vision to extract information from images for use in image search. Clustering and image prototyping using image collection exploration and automatic summarization are other notable methods.

The Challenge

Our task was to predict relevant images given a natural language search query. We were provided with a training set of 10,000 images along with corresponding lists of tags indicating objects in the image, feature vectors extracted using ResNet, an advanced convolutional neural network, and five-sentence descriptions. For evaluation, we were given a separate set of 2,000 images and matching tags and 2,000 five-sentence descriptions. For each image, we had to predict 20 relevant images, rank-ordered by relevance. Mean Average Precision at 20 (MAP@20) was used to evaluate the results based on the positioning of the actual image match within the output.

Our Algorithm

EXPERIMENT

Data Encoding

We explored various ways to represent the text description data for use in statistical techniques, such as Bag of Words and Document Vectors.

The bag of words approach is a common feature generation tool that involves representing text data as a bag of words that constitute it. Note that bag of words is an orderless representation of the text data. How words appear in relation to one another is not modeled in this representation.

In contrast, word vectors are numerical representations of relationships between words (e.g. man: woman → king: queen). Document vectors are an extension of the concept of word vectors, and represent documents in a corpus of text as vectors in an n-dimensional space. They are created using a shallow neural network that transforms cleaned and vectorized text data into a representative numerical feature vector, also called the embedding.

We ultimately chose to use Bag of Words for data encoding as this produced more accurate results in description matching.

Approaches

Since we had to create mappings between test descriptions and test images, we explored multiple approaches to tackling this problem. **Figure 1** shows a mapping the data provided, which intuitively pointed us in a few directions.

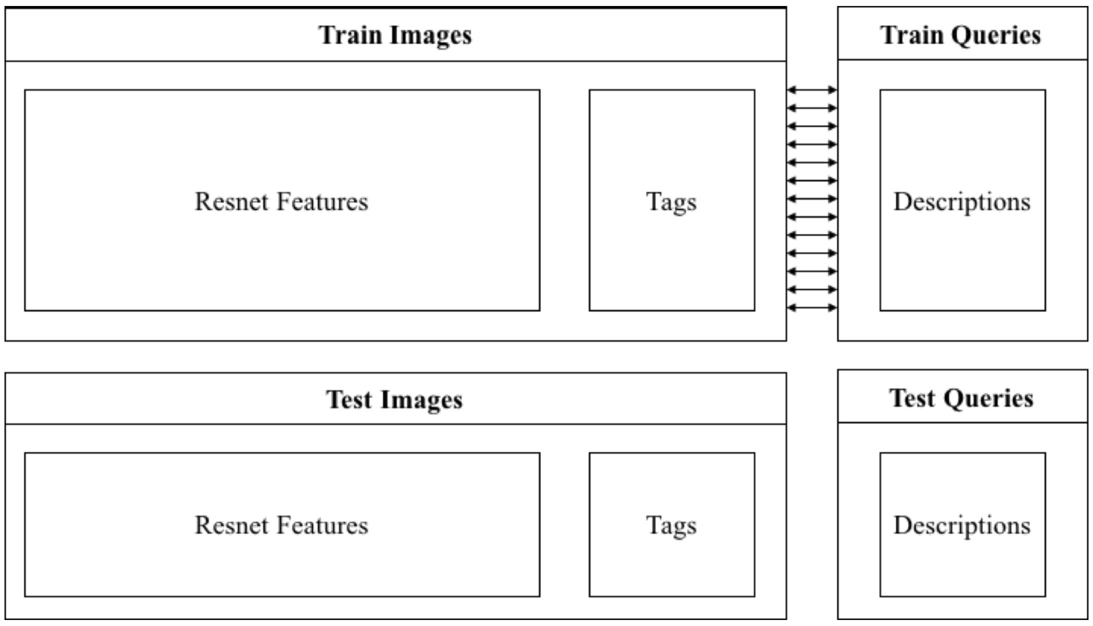

Figure 1: Schematic Representation of Data Provided

Queries to Image Approach By mapping test queries to train queries, we get the corresponding train images since there is a one-to-one relationship in the training data. For these images, if we find the most similar test images using image ResNet features and image tags provided. This allowed us to create a mapping between test queries and test images. We found that this general approach yielded better results than the Queries to Image Approach.

Test description → Closest Matching Train Description → Train Image → 20 Most Similar Test Images

Image to Queries Approach For each testing image, if we find a set of similar training images, we can access their descriptions to create an aggregate description for a given testing image. We can then calculate pairwise similarities between aggregate descriptions for the test images and test descriptions. Using these similarity scores, we can arrive at a set of top matches for each test description. This approach leaded to lower results.

Test Image → Similar Train Images → Train Description → Aggregate Description <similarity scores> Test Descriptions

Description-to-Image KNN Model

Data Representation We parsed the training and testing feature set for the final Resnet layer fc1000. We processed descriptions by removing all punctuations, lowercasing all words, removing stopwords and lemmatizing all words using NLTK's Porter Stemmer. We then created a dictionary of all words appearing more than 10 times in training and testing descriptions. We built a bag of words representation of training and testing descriptions using the dictionary. We also performed L2 normalization on the training and testing description features to normalize the vectors.

Algorithm We used scikit-learn's kNN classifier to fit training description features to image names. For each test description feature, used the .predict method in scikit-learn on the KNN classifier (n_neighbor=1) to get the closest training description based on the description feature vector. Since this directly mapped to a training image, we could get the corresponding ResNet training feature vector for that image. We then computed similarity scores for the testing image ResNet fc1000 feature vectors using a distance function to the training image ResNet fc1000 feature vector. We experimented with cosine similarity and euclidean distance as similarity measures. We found that using cosine similarity provided better scores than using the euclidean distance. For each test feature/label combination, we find by the lowest 20 scores (closest test features) and return the test image labels associated with them to be used for submission.

Image-to-Description Model

Data Representation Similar to the approach for the Description-to-Image model, we preprocessed the descriptions and built a bag of words model for the training and testing descriptions.

Algorithm For each testing image, we find two closest training images by computing the cosine similarities between our test image and the training images using the 100 ResNet fc1000 features. We then compute the aggregated description feature vector based on the nearest training images. We built a map of the testing image to its aggregated description feature. For each test description feature, we then find the top 20 aggregated description feature associated with the testing images using the euclidean (L2) distance between the aggregated description feature and the testing feature vectors. When finding the distance between the description feature vectors in training and testing data, using the euclidean distance provided better measures than using the cosine similarity function.

Similarity Mapping

Data Representation We used the same data representation as in the Description-to-Image model for this approach. In addition, we also cleaned and projected the tags data on images to a 80-dimensional bag of words vectors.

Algorithm We calculated similarities between testing description vectors and training description vectors using cosine similarities. For each testing description vector we found n_1 training description vectors that closely matched. For each of these n_1 corresponding training images, we used ResNet fc1000 + 80-dimensional tags data as features to calculate cosine similarity with testing images. We then found n_2 testing images for each of these n_1 training images with highest similarity scores, while maintaining n_1 x n_2 = 20. We found that n_1=4 and n_2=5 works best for our Kaggle submission scores. In addition to cosine similarity, we tried euclidean and euclidean-weighted-cosine similarities. We found that cosine similarities yielded the best results for matching both description vectors and image vectors.

In addition to the (n_1,n_2) approach used above, we also tried to use similarity chaining, by creating a combined similarity score as net_similarity = description_similiarity x image_similiarity. We found that this approach did not improve on the results of the (n_1,n_2) approach.

Test descriptions <similarity scores> Train Descriptions → Train Images <similarity scores> Test Images

Predicting Descriptions

If we could create a statistical model that mapped training image features to training descriptions, we could predict descriptions given a training image. We could use these to calculate similarities like in the Image-to-Query approach above. Note that unlike the approach above, this does not look at similarities between test and train images explicitly.

Data Representation We cleaned the descriptions data as above through lemmatizing, removing stopwords etc. For projecting descriptions as vectors, we used gensim's Doc2Vec module to create document embeddings out of this cleaned text with 100 dimensions and 100 training epochs.

Algorithm We used ResNet fc1000 and bag of words as input features for our model. We built a sequential neural network and trained it with training images and training description data. Given a test image vector, the goal was to predict description vector as output. We tried various configurations of the network through change in number of layers, neurons per layer, activation functions, use of drop out to prevent overfitting, various optimizers and learning rates to tune the model. Here are the specifications of the final model:

- Module: Keras on TensorFlow backend
- Input Layer: 1080 neurons, linear activation
- Second Layer: 1620 neurons, linear activation, dropout=0.1
- Third Layer: 500 neurons, reLU activation
- Output Layer: 100 neurons, linear activation

Learning rate=0.01, Optimizer= adam, Loss function: mean squared error, Epochs=50

We found that the results of the description predictions were dismal on manual review. Increasing the neural network architecture added to the computational workload, and made running it on local machines not feasible. We looked into using Google Cloud platform for training larger networks, but decided not to in the interest of limited time.

RESULTS & CONCLUSION

Approach	Public Leaderboard Kaggle Score	Private Leaderboard Kaggle Score
Description-to-Image KNN Model	0.13235	0.13631
Similarity Mapping	0.13442	0.1406
Image-to-Description Model	0.17067	0.14690
Similarity Mapping	0.22709	0.21257

We found that the Image-to-description approach yielded the best results. Using euclidean distance to find the closest descriptions based on bag-of-words allowed us to generate a more accurate representation of the target image for a given description.

After trying various representations of the data, we found that using the ResNet fc1000 features along with tags as bag of words features out performs ResNet fc1000 or tags alone in isolation in image-to-image matching.

We believe that we could improve image to image matching by using intermediate pool ResNet layers. We are still inclined towards exploring document embeddings to improve our description matching. In addition, using methods like neural networks or SVMs to model relationships between image features and text features seems like a plausible future direction.

Background & Related Work

Wang, Baiyang, and Diego Klabjan. ?An Attention-Based Deep Net for Learning to Rank.? [1702.06106] *An Attention-Based Deep Net for Learning to Rank*, 15 May 2017, `arxiv.org/abs/1702.06106`.

Li, Yuncheng, et al. ?Improving Pairwise Ranking for Multi-Label Image Classification.? [1704.03135] *Improving Pairwise Ranking for Multi-Label Image Classification*, 1 June 2017, `arxiv.org/abs/1704.03135`.

Mitra, Bhaskar, and Nick Craswell. ?Neural Models for Information Retrieval.? [1705.01509] *Neural Models for Information Retrieval*, 3 May 2017, `arxiv.org/abs/1705.01509`.

Wang, Liwei, et al. ?Learning Deep Structure-Preserving Image-Text Embeddings.? [1511.06078] *Learning Deep Structure-Preserving Image-Text Embeddings*, 14 Apr. 2016, `arxiv.org/abs/1511.06078`.

Palangi, Hamid, et al. ?Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval.? [1502.06922] *Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval*, 16 Jan. 2016, `arxiv.org/abs/1502.06922`.

Le, Quoc V., and Tomas Mikolov. ?Distributed Representations of Sentences and Documents.? [1405.4053] *Distributed Representations of Sentences and Documents*, 22 May 2014, `arxiv.org/abs/1405.4053`.

SOURCES & EXTERNAL LIBRARIES

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

John D. Hunter. *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

Wes McKinney. *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830 (2011)

Jones E, Oliphant E, Peterson P, et al. *SciPy: Open Source Scientific Tools for Python*, 2001-, `http://www.scipy.org/`

"A Gentle Introduction to the Bag-Of-Words Model." *Machine Learning Mastery*, 20 Nov. 2017, `machinelearningmastery.com/gentle-introduction-bag-words-model/`.

What is the difference between bag of words, TF-IDF, and vector space model?, `www.quora.com/What-is-the-difference-between-bag-of-words-TF-IDF-and-vector-space-model`.

C., Gab. "Bag of Words and Tf-Idf Explained." *Data Meets Media*, 7 May 2017, `datameetsmedia.`

`com/bag-of-words-tf-idf-explained/`.