CS5785 Final: Image Search Engine

Anonymous Authors

Abstract

This paper aims to investigate methods of building a large-scale image search engine using various statistical learning techniques. The search engine takes a natural language query as input and outputs a rank-ordered list of images with the most relevant image ranked highest. Available data at our disposal to complete this task included a large set of training images with associated natural language descriptions, natural language tags, and image features extracted from a pre-trained residual network (ResNet). Our team experimented with various modeling techniques and data utilization to build a search engine which identifies images given their associated descriptions. Ultimately, a combination of multiple models proved to be most successful.

Introduction

Image classification and text-based retrieval has been an area of ongoing research for the last several decades. Furthermore, as the size of digital media libraries grows at an ever increasing rate, the ability to leverage the information within these images depends on improved methods of retrieval.[1] Potential applications of computer vision-type approaches include search engines, satellite imagery, medical pathology and radiology, and facial recognition, to name a few. Recent advances in convolutional neural networks have resulted in an improved ability to classify objects present in images with a high degree of accuracy.[2] These networks utilize residual neural networks, which are based on architecture found in pyramidal cells of the cerebral cortex.[3] In contrast to traditional deep networks, residual networks incorporate skip connections that allow for combination of data before and after it is passed through layers of a more traditional neural network, in theory allowing for greater feature identification.[4]

In this project, we were tasked with using human-generated natural language descriptions to retrieve a series of images most likely to correspond to that text. For each image, we are provided that image's output features from ResNet152 along with human-curated tags.

Ultimately, our best-performing algorithm worked as follows: a Bag of Words representation of descriptions was fit to the top principal components of the ResNet152 outputs vis a multilayer feedforward neural network, and predictions were made with the 20 nearest neighbors as measured by cosine similarity. Improvements were made by incorporating linear combinations of the distance measures obtained by fitting an 2-gram representation of the descriptions, as well as distance between the description and target bag of words vectors.

Experiments/Methodology:

1. Description/Tag similarity

Given that the natural language descriptions and the human-annotated tags for images exist in similar English-language spaces, we decided our first approach would simply aim to make predictions based on these aspects alone. We preprocessed the 5-sentence descriptions to remove common words and punctuation and lemmatize them to derivative roots, and then preprocessed the tags by removing the categories for each tag. Next, we calculated the Jaccard similarity between each description and each image's set of tags. Finally, for each description, we ranked the top 20 images whose tag words had most overlap with the keywords in the preprocessed description.

2. Word2Vec Representation

We first started by converting the 5-sentence natural language descriptions into a corpus of sentences upon which to create a set of word2vec features for each unique lemmatized word, as well as calculating term frequency-inverse document frequency (TF-IDF) for each word present in the entire set of descriptions[6]. The Word2Vec model embeds text using a shallow, 2 layer feedforward neural network (source). Each 5-sentence description could then be represented in the word2vec feature space by weighting each word in the description's word2vec representation by its TF-IDF value, and taking the subsequent mean of these weighted vectors. The testing descriptions were then passed through the same word2vec model weighted by the TF-IDF values from the training descriptions. In constructing this model, several parameters were modified and tested. In all cases, we used word2vec vectors ranging from 100 to 5000 features in length. We also incorporated the testing descriptions into the word2vec library and TF-IDF calculations to minimize the number of new words encountered by the model in the testing data.

3. Bag of Words using Pool5-KNN Representation

The goal of this approach was to calculate distances from each testing Pool5 representation to all descriptions in the testing set. Each Pool5 is a compressed 2,048 dimensional feature derived from the final convolution layer of a pre-trained residual neural network (ResNet).[7] For each testing Pool5 representation we found the 15 closest Pool5 representations in the training set by k-nearest neighbors using cosine distance. We then took the 15 Bags of Words associated with those 15 closest training images and assigned their average (a 1x7090 vector) to the testing Pool5. From there we calculated the cosine distance from this Bag of Words to the Bag of Words of each testing description. After doing this for each Pool5 in the testing set, we had the distances from each testing description to each testing image. Our predictions for a given description were the 20 closest images to that description.

4. Regression and Predictions

Once we establish a method for embedding text descriptions into continuous numerical vectors, the next step is to generate a prediction for a given image and its features. This involves creating a mapping between a *p*-dimensional description vector (where *p* is the dimensionality of our Word2Vec or Bag of Words representations) to the 1000- and 2048- dimensional outputs of the ImageNet (*fc1000* and *pool5*), as well as the provided tags. Due to both the high dimensionality and sparseness of these target vectors, we first performed principal component analysis on a concatenation of these features, obtaining the top 100 features capturing >99% of the variance (Figure 1). The output of both models (LR and NN) is therefore a 100-dimensional vector.

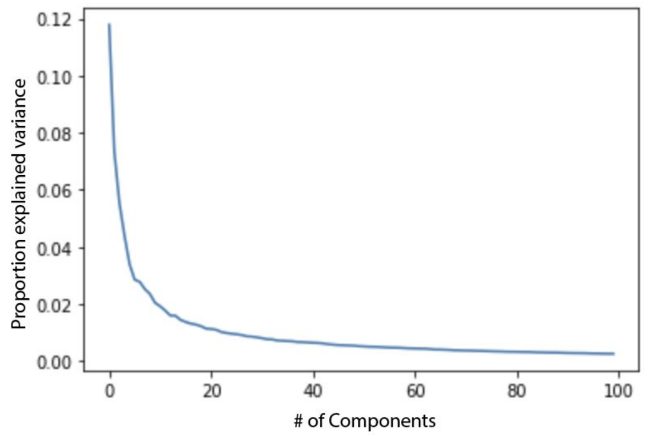

Figure 1. Explained variance by the top 100 principal components of the target ImageNet features.

We took two approaches to fitting to this target, first, fitting the word2vec and bag of words representation with multiple linear regression (LR) by least squares. Our second approach was to fit our own bag of words with a multilayer feedforward neural network (NN), built using TensorFlow/Keras. The architecture consists of two hidden layers, each of 2000 and 200 units respectively. The depth of these layers was determined by a brief hyperparameter search. Overfitting was prevented by using an early stopping criteria based on minimal improvements in held-out validation loss.

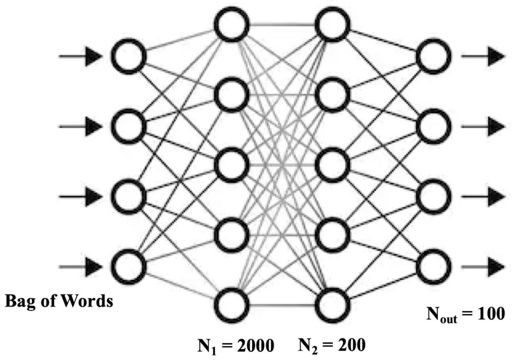

Figure 2. Diagram detailing the multilayer feedforward neural network used to predict image features from Bag of Words vectors.

As an alternative, we tried both partial least squares regression, random forest regressors, as well as a basic extreme learning machine implementation, though neither improved performance.
The predicted feature vector is then compared against the top 100 principal components of all the test images. The top 20 predictions, as measured by cosine similarity, were chosen in a ranked order for the predictions. Cosine similarity was used over euclidean distance due to its better performance in high-dimensional data.

5. Additional models
Aside from bag of words and Word2Vec, we also implemented a 2-gram model of the descriptions, and fit the target features with both the LR and NN. This model seemed to capture slightly different features in the target images. We found that we could significantly improve our accuracy if we sorted based on a linear combination of the two distance metrics. This combined model was our also our highest performing.

Results

1. Description/Tag Similarity

Ultimately, this initial approach yielded a result of 0.21 on the public testing set, with a couple of implications that guided our remaining approaches. First, it became evident that the text of the descriptions already had significant direct links to the images through their tags. However, rather than using matches of exact words and word roots, it became apparent that converting descriptions into feature vector representations that represent embeddings of larger concepts, or groups of synonymous words, might be more advantageous.

2. Word2Vec

We reasoned that such embeddings were better at capturing larger concepts in the description that map to the image features. This was validated by using the `most_similar` feature of the word2vec model.

For instance, looking for the words most closely related to “baseball” resulted in words such as “bat,” “catcher,” and “pitching,” demonstrating the model’s ability to capture conceptual similarity in words not at all related by spelling or root/base word. Nevertheless, this model is imperfect. A query for the word “birthday” returns the words “cake” and “year,” but also the word “pigs.”
However, simple regression models such as linear regression or partial least squares regression were not sufficiently generalizable to test data and feature representation of objects found in test images that might not have been present in the training data. Using a word2vec embedding of image descriptions yielded an optimal result of 0.25 on the public testing data.

3. Bag of Words and Pool5-KNN

This approach paired with linear regression gave us a score of approximately 0.27 on Kaggle. Cross-validation indicated that this score was not particularly sensitive to post processing of the bag of words model or changing the numbers of “neighbors” for each testing Pool5. The score was significantly improved by adding the feedforward NN in place of linear regression, achieving a score of approximately 0.33. By incorporating the 2-gram model into our predictions, we further improved the score to 0.37 on the public testing data.

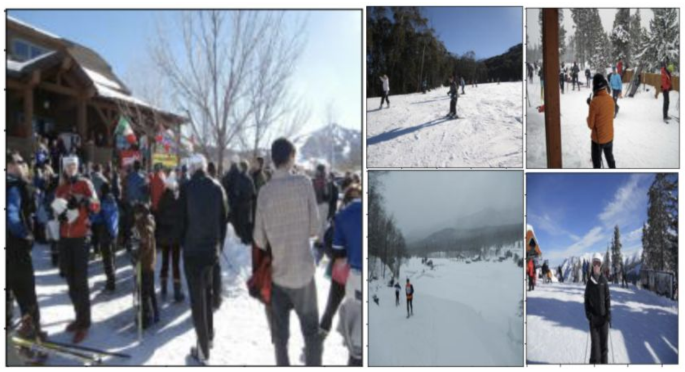
Figure 3. *Left*: An example of the model correctly predicting an image for description: "A large crowd of skiers gathered near a ski lodge" and "A large crowd of people stand in the snow outside a building.". *Right*: The next four related images demonstrating similar features.

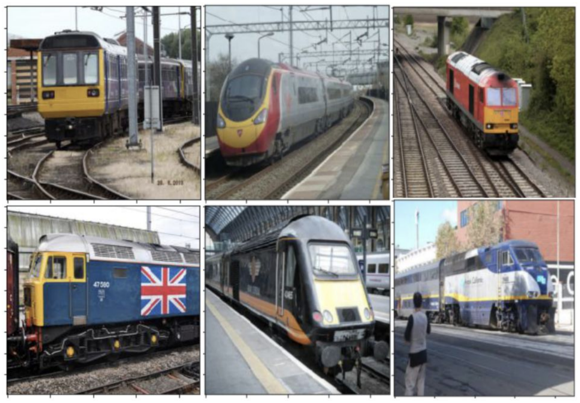
Figure 4. Example of incorrect predictions by the model for descriptions: “A passenger train that has some graffiti on it” and “a blue yellow and black train car with graffiti”. Note, that while the model seems to have captured the subject of the description, it has not picked up on the word “graffiti”.

Conclusions

27 Nothin but Nets 0.36998 9 12m

Our final model, with 0.37 accuracy, was fairly successful at identifying the correct image from the test set using the written description. As seen in the representative examples from training data, the model is likely fairly accurate in identifying a feature’s presence within an image. In the case of Figure 3, trains were correctly identified in all top images, but none specifically with graffiti on them. Thus, our approach is likely sufficient for finding objects present in written descriptions, but is not yet capable of more complex contextual relationships between objects in a description.

References

1. Rui, Y., Huang, T. and Chang, S. (1999). Image Retrieval: Current Techniques, Promising Directions, and Open Issues. *Journal of Visual Communication and Image Representation*, 10(1), pp.39-62.
2. He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
3. Wikipedia. (2019). *Residual neural network*. [online] Available at: https://en.wikipedia.org/wiki/Residual_neural_network.
4. Shorten, C. (2019). *Introduction to ResNets*. [online] Medium. Available at: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4.
5. Gupta, S. (2018). *Overview of Text Similarity Metrics in Python*. [online] Medium. Available at: https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50.
6. Ethereon.github.io. (2019). *Netscope*. [online] Available at: https://ethereon.github.io/netscope/quickstart.html.
7. Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. [online] arXiv.org. Available at: https://arxiv.org/abs/1301.3781.