Ridge Regression with L2 Nearest Neighbor Search for Caption-Image Retrieval

December 6, 2018

Abstract

Online search engines have become widely used and incredibly useful in the digital age. With the advent of machine learning, modern information retrieval is not limited to text based lookup anymore. In this paper, we experiment with multiple state-of-the-art machine learning models to create cross-modal image retrieval for text input.

1 Introduction

The combination of natural language processing and computer vision have become more intertwined in recent decades with the advancement of machine learning, GPUs, and TPUs. One of the important topic in recent years is cross-modal retrieval. There are many papers introduced recently on cross-modal retrieval methods [1]. However, most focused on caption retrieval for image and video [2] [3]. Recently, with the advancement of recurrent neural networks, the community has focused more on caption generation for image and video. This paper focuses on image retrieval for caption input. We develop a novel Ridge Regression with L2 Nearest Neighbor Search algorithm that achieves state-of-the-art accuracy for caption-image retrieval task.

1.1 Project Overview and Data

In this paper, we will discuss the steps we took to create an caption-image retrieval model that takes in short sentences as queries and returns the top 20 images that best fit the query. We use a data set that is comprised of images, image descriptions, image tags, and image feature vectors extracted from the last convolutional layers and the last fully connected layers of Deep Residual Network, a state-of-the-art convolutional neural network [4]. Our training data set is made up of 10,000 different images, each image with five different sentences that describe it. We have 2,000 testing samples. The ReNet feature vectors have 1,000 features for the ones extracted from fully connected layers and 2,049 features for the ones extracted from convolutional layers.

1.2 Motivations

The task requires to match image feature space to textual space. This motivates three key types of methods. First, keep everything within the original feature space and only do comparison within the feature space, this inspires our first method of Nearest Neighbor Search. The second is regression model that intakes all the feature vectors regardless of the original feature space and compute the likelihood of the sample being positive, this inspires our second approach of multilayer perceptron regression. The last approach is to transform either image vector to text vector, or vice versa, this insires our last and main approach for the paper. All of these methods were attempted by us in this paper.

2 Model Design

In this section we explain the data processing strategy and the model architecture we designed.

2.1 Data Processing

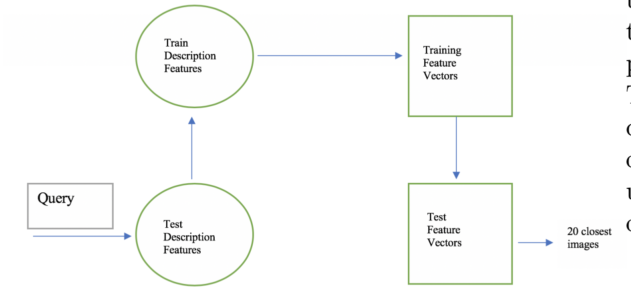
As with all machine learning work, the most important work is data processing. We focus specifically on description parsing since the image features are already extracted using ResNet and the tags are in the same texual space as the description and be parsed using the same technique as description parsing.

In this paper, we explore three different pre-processing approaches. In the first method we develop a uni-gram bag of words implementation that captures all the words in the descriptions whille lemmitize and eliminating stop words fromthe descriptions. The second approach was Word2Vec, a popular word embedding model that utilizes shallow two-layer neural networks. We used a model pretrained by [5]. The third pre-processing approach is a complex Bag of Words model that utilizes part of speech tagging to remove words that does not serve as nouns in a sentence [6]. Both the training image descriptions and testing image descriptions are pre-processed.

2.2 Nearest Neighbor Search

In this model we used a Bag of Words representation to pre-process our data. The pre-processing includes lemmatizing the sentences and remove all stop words. The resulting bag of words vectors had 6,000 features. The features are then normalized using L2-normalization on each sample.

With our pre-processed data, we began the kNN process. The idea of this kNN is to keep every feature vector in its original feature space to eliminate the loss of information when features are transformed between feature spaces. The input test description is compared to the train description to find the most similar description. The train image corresponding to that description is then retrieved since the mapping for training set is known. The train image is then compared to all the testing image to find the most similar test image. For each comparison, 5 to 20 samples are chosen to proceed with a result of 25 to 400 samples. These samples are then ranked by weighting the texual space distances and the image space distances.

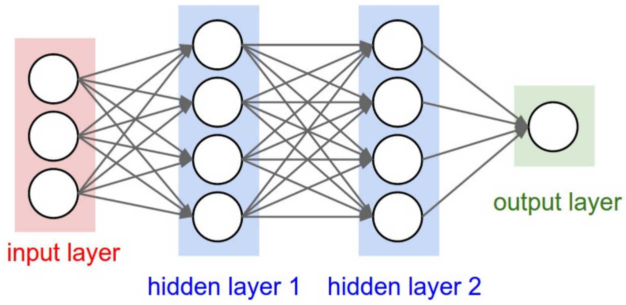


2.3 Neural Network

After completing the kNN process and receiving our predictions, we realized a score of about 12% accuracy. However, we believed that we could improve upon this, and implement a more accurate model. Thus, we changed our approach. Neural Networks are well renowned in the field of Machine Learning for their high accuracy. The approach we chose to use was a Multi-Layer Perceptron model. This is a feed forward style of neural network with at least 1 hidden layer. MLP models use non-linear activation functions and are trained using back propagation techniques.

The using an Neural Network was not the only thing that we changed in this model. We also adjusted the way that the data was being pre-processed. From our Bag of Words approach, we quickly moved to using a Python add-on package called gensim. This package has a very powerful word pre-processing package called Word2Vec. Both the training image descriptions and testing image descriptions are pre-processed and put into a Bag of Words feature vector format. The total length of our BoW vector from our Image descriptions came out to be 300.

While neural networks are known to be more accurate, they are also much greedier. They take up both a lot of time and a lot of computing power. This was a large inconvenience due to our limited time and limited computing power. The use of GPU would be ideal in a situation such as this. Initially, we attempted to train a multi perceptron model with 2 hidden layers. We experimented with both the number of neurons in each hidden layer, as well as the number of hidden layers. In addition to this, we experimented with PCA dimensionality to try and pinpoint a more accurate model. PCA is a process of unsupervised dimensionality reduction. The process uses orthogonal transformations to reduce data in n dimensional space, down to that of one's choosing. In this process, we initially chose to use a PCA process on our ResNet vectors to reduce our 1,000 features to 300.



Multilayer Perceptron Model

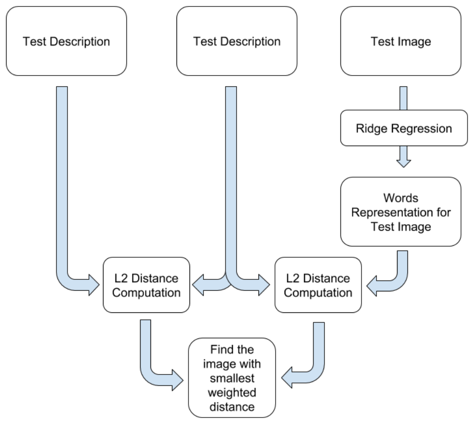
We began to realize that we could not create a neural network large enough to get accurate results. We did not posses the proper resources, and no matter how much effort we put in to adjusting the model and its parameters, we were not able to significantly increase the accuracy.

2.4 kNN With Regression

Despite our many attempts to find a model that performed better than the original kNN, we were unsuccessful. Each trial resulted in an accuracy that was less than our previous model. With the decrease in accuracy from our multi-layer perceptron model, we decided to go back, revisit and modify our kNN model. There is not a lot that can be done to change the kNN algorithm, however we were able to pre-process the data, more and experiment with parameters differently. Instead of using a Word2Vec, or our original Bag of Words model, we began by adjusted our data pre-processing procedure.

Our new pre-processing used another Bag of Words method. This new method keeps only nouns from the test and train images descriptions, removes any pluralization, then uses a TF-IDf Vectorizer to adjust to 1- and 2-gram models. Finally, it removes any words that occur above and below a certain threshold. Initially, we started with a 1-gram model and we removed any remaining words that appeared less than 10 times. However, we tuned these parameters to increase accuracy

In addition to our word processing, we introduce a new method of data processing. We hypothesized there might be a better technique to process the ResNet vectors than PCA for dimensionality reduction. We decided that it may be beneficial to use a regression technique to adjust the ResNet vectors to the list of their associated features in English, and then map those features into a bag of words vector. From there, kNN can be used as in our very first attempt.



3 Experiments

In this section, we look deeper into the three different models, and make adjustments in an attempt and maximize our overall accuracy. We look at how changing different parameters and inputs within each model affects their overall accuracy

3.1 kNN

The euclidean distance between the bag of words vectors and the ResNet image vectors do not necessarily match each other, so we add weights to the distances to increase accuracy. In this case there is no majority vote, it takes only the closest neighbor.

3.2 Neural Network

For validation, we split the training data set of dimension 10,000 into a sub set of 8,000 and 2,000. We started with an MLP model that has 3 layers and 680 different neurons in each layer. This MLP used a high adaptive learning rate with a ReLU activation function and termination if the loss decreased by less than .0001 in ten iterations. The first variable we tested was the performance of the model with and without conducting PCA on the ResNet vectors. We also tested the size of the dimension that PCA should reduce or vector space to. In additioin, we experimented with the number of layers in our Neural Network, and the number of neurons in each layer. The table below shows a summary of the different trials.

Test #	Model	Hidden Layers (Total Layers)	Neurons	PCA	Accuracy
1	MLP	3 (5)	680	300	0.014
2	MLP	3 (5)	1380	N/A	0.001
3	MLP	4 (6)	680	300	0.0135
4	MLP	3 (5)	1360	300	0.0015
5	MLP	3 (5)	580	200	0.013
6	MLP	3 (5)	780	400	0.0085

Multilayer Perceptron Resutls

We initially selected 680 neurons to match up with our 680 inputs. 300 from the bag of words vector, 300 from our PCA reduction, 80 from our tags. We tried doubling the number of neurons with respect to our number of inputs, as well as a few other techniques. Our results were not satisfactory. Our accuracy fluctuated around 1%.

3.3 kNN With Ridge Regression

This experiment was set up with a new pre-processing technique as described back in the introduction of this paper. Our new Bag of Words model is much more complex, with variable inputs, type if n-gram, norm type, binary or not, and minimum number of appearances. We started with 1-gram model, Euclidean norm, non-binary, and minimum number of appearances of 10. However, different to our word2vec process, our bag of words vectors now initially contained 986 different features.

First, by changing only one variable at a time, we tuned our pre-processing to have the best accuracy of all the models. The Ridge Regression on our ResNet feature vectors, as well as our kNN procedure also had features that we tuned. We tuned the alpha of our ridge regression as well as the weights on our images and tags to produce the highest accuracy. Below is a chart of the different results we received as we tuned our model.

Test #	Regression Type	Alpha	Min_df	Binary	Weights (image, tag)	Accuracy
1	Ridge	10	10	F	(1,0)	.179
2	Ridge	10	10	F	(10,1)	.1885
3	Ridge	10	10	F	(23,1)	.194
4	Ridge	10	10	T	(23,1)	.2185
5	Ridge	10	10	T	(13,1)	.2245

kNN with Ridge Regression Tests

In these experiments, we find that the accuracy of our model greatly increases when the pre-processing word vectors are binary. Furthermore, we see a significant increase in accuracy when there is a higher ratio of weight on the distance between image feature vectors to weight on the language vector distances.

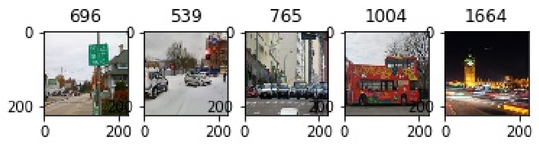
The accuracy presented in the above tables are validation accuracy calculated by $\frac{20+1-i}{20}$. The actual accuracy that our best performing model actually achieve is 0.40401 accuracy on the Kaggle Competition submission.

4 Results and Analysis

Our experimentation shows that the results for advanced computation methods can sometimes be superfluous. Word2Vec and and Multilayer Perceptron are all very powerful methods that the community relies on. However, our results shows that their performance is sub par to the simplest kNN and regression method. While our greatest achieved accuracy is below 50%, the kNN with regression model performs extremely well given the difficulty of the problem.

Let us compare the accuracy of our models visually. First let us look at the the sentence query for a given search in our testing data, and we will display the top 5 image outputs from each model. The query is:

an orange fire truck parked in a wharehouse
An antique truck displayed in a work garage.
A red fire engine is parked in the fire station.
A red fire engine is inside of a building.
The old model fire truck sat alone in the room.



Top 5 image predicted using our kNN model

We see that our kNN model without using a ridge regression does not produce seemingly accurate results. We can see that one of the given photos contains a red vehicle, but it is not even a fire truck. This model alone would not be sufficient for a real search engine.

Top 5 image predicted using our neural network model

The results generated by our Neural Network model are seemingly even worse than that by our kNN model, and they are. Our Neural Network results are almost exactly the same as if photos from our data set were randomly selected. We can see a red fire hydrant, but that image may arrive just out of random, not because the words fire and red appear in the query.

Top 5 image predicted using our kNN with Ridge Regression model

Our kNN ridge regression model outperformed the others. The model was over 40% accurate. Every photo that our model produced had something to do with the query that was searched.

The image features are taken from the last layer of ResNet, each of these features fundamentally corresponds to one of the 1,000 ImageNet classification class. With this observation, parsing the description with a bag of words that only collects singular nouns allows us to generate a feature vector that lies in similar subspace as the ResNet features. A simple ridge regression then maps the similar sub-spaces to each other with little loss of generality. The resulting feature vector proves to approximate the description very well. The tags further strengthens the prediction by adding what the image did not capture. The weight for the tags distance is usually best at around $\frac{1}{13}$ compared to the weight for image distance given our other parameters. This further confirms the validity of our model design.

References

[1] et al Wang, Katie. A comprehensive survey on cross-modal retrieval. (arXiv:1607.06215), 2016.

[2] et al Anderson, Peter. Bottom-up and top-down attention for image captioning and visual question answering. *A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society*, (arxiv:1707.07998), 2018.

[3] et al Yu, Youngjae. End-to-end concept word detection for video captioning, retrieval, and question answering. *A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society*, (arxiv:1610.02947), 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] Google LLC. word2vec, 2013.

[6] Ewan Klein Steven Bird and Edward Loper. 5. categorizing and tagging words, 2014.