CS 5785 Final - Image Search

Department of Computer Science
Cornell Tech
New York, NY

Abstract

Text-based image retrieval requires pre-processing both text and image data. With a training set of 10,000 images and 10,000 descriptions, natural language processing techniques filtered out unnecessary text information, while ResNet and PCA were used for feature extraction and dimensionality reduction of the images. A neural network with one hidden layer, an elu activation function, hinge loss, and regression output produced a 45% accuracy when predicting images based on text inputs.

1 Introduction

The information gathered for training includes 10,000 images, descriptions, tags, and both intermediate and final ResNet produced features. The following diagram highlights the processing accomplished on both text and image data. This process is explained in further detail in the following sections.

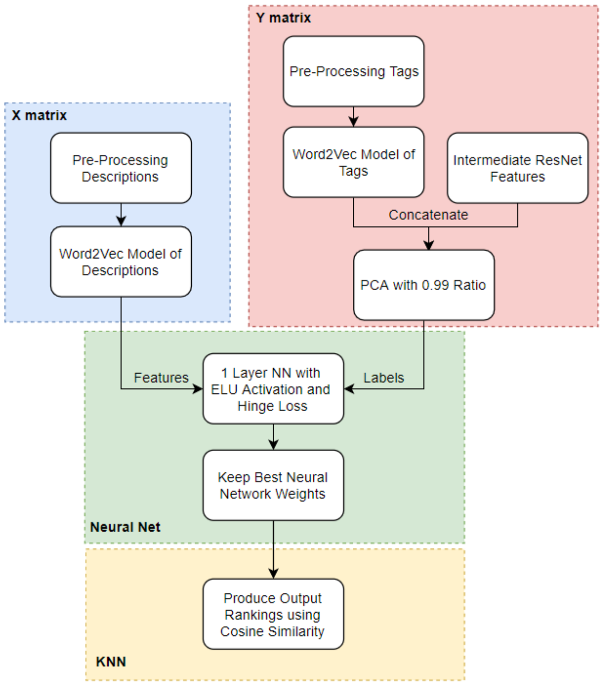

Y matrix
Pre-Processing Tags
X matrix
Pre-Processing Descriptions
Word2Vec Model of Tags
Intermediate ResNet Features
Concatenate
Word2Vec Model of Descriptions
PCA with 0.99 Ratio
Features
1 Layer NN with ELU Activation and Hinge Loss
Labels
Keep Best Neural Network Weights
Neural Net
Produce Output Rankings using Cosine Similarity
KNN

Figure 1: Model Hierarchy

1.1 Text Processing

To improve accuracy of matching text descriptions to related images, the nltk open source library was used. This includes removing stop words, numbers, white-spaces, and punctuation. This is done because they do not provide useful information in finding the context or true meaning of the sentences. It is important to note stemming and lemmatizing was not done, to allow distinctions between similar words, i.e. skateboarder and skateboard. With these distinct words, Word2Vec[1] from the open source Gensim[2] library was used to convert words into a common feature space.

Through experimentation different trained data-sets for the Word2Vec model were used including GLove, Google News and training our own corpus. The best performed model was Google News, as can be expected as its corpus was largest with 100 billion words. A non-binary approach as well as the sum of word vectors produced the highest MAP@20 score. Lastly, empty tag files were replaced with a vector of zeroes. Both descriptions and tags were run through the same NLP processing steps for consistency.

1.2 Image Processing

With image features extracted by ResNet, the difficulty of image processing was abstracted. However, given intermediate features and final features extracted by ResNet, our experimentation determined that the intermediate features were a better label for predicting images.

1.3 X Matrix

Before training the model, a precise feature matrix (X) needed to be created. By processing the description texts as described in the text processing subsection, a feature matrix was created by adding all the word vectors of a description into one feature vector.

1.4 Y Matrix

Likewise, to train the model a precise label matrix (Y) needed to be created. Through experimentation, it was found that the intermediate ResNet features better trained the model than the final ResNet features for images. Additionally, tags were processed as described in the text processing section, and converted into a summed word vector for each image. These tag word vectors were then concatenated with the intermediate ResNet feature vectors for each image. Experimentation showed this combination greatly improved prediction accuracy for the model. Finally, because this label matrix was high in dimensionality with ample noise, PCA [3] from sklearn helped reduce the high dimensional features down to 1020 dimensions while still maintaining 99% of the data's variance. This also reduced training time and highlighted the important features.

1.5 Neural Net Model

With the X and Y Matrices described above and thorough experimentation, a Neural Net with a single hidden layer, an ELU activation function, and hinge loss was found to perform the best predictions. With 5-fold cross validation the best hyperparameter values were found for the net. Finally, saving every other epoch out of 60, the MAP@20 score was run on the validation data and the best weights were saved for testing.

1.6 K Nearest Neighbors

Finally, to map predicted output to the actual images, the true labels found from the Y matrix fit K Nearest Neighbors, and for each predicted label, its 20 nearest neighbors were found with the Cosine Similarity Metric.

2 Experimentation

Through experimentation, 2 models outperformed all others. The following sections describe these 2 models in detail. Each model's hyperparameters were found using 5-fold cross validation by splitting the training data into 8,000 training points and 2,000 validation points. Both these models utilized text pre-processing, PCA for dimensionality reduction, a single layer Neural Network, and KNN Cosine Similarity.

2.1 KNN with Cosine Similarity

In order to measure our accuracy, the regression output is ranked based on its closest nearest neighbors by different distance metrics. Experimentation showed cosine similarity outperformed Euclidean Distance. This can be explained due to the fact that direction is more relevant than distance (i.e. if two vectors are similarly oriented they should be considered to be closer together, over the raw distance between each other).

2.2 Model 1: Sigmoid & MSE Loss

Model 1 differentiated between other models by using a sigmoid activation function and mean squared error loss. A sigmoid activation function $R(\gamma) = \dfrac{1}{1+e^{-\gamma}}$ is a saturating nonlinearity and has a smooth transition which expresses the data well, as shown through experimentation. We used the mean squared error (MSE) as our loss function, which is calculated as follows: $Loss = \dfrac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. This function greatly penalizes points (squared error), regardless of whether or not they are positive or negative, making it ideal in a regression setting.

2.3 Model 2: eLU & Hinge Loss

Model 2 outperformed all other models using eLU and Hinge Loss with a Kaggle score of 0.45. The eLU activation function yielded better results than the ReLU activation function. This is because it is able to produce negative values. In addition and is has a smoother transition which is able to better express the data [4]. The Exponential Linear Unit (eLU) function $R(z)$ is defined as follows:

$$R(z) = \begin{cases} z & z > 0 \\ \alpha \times (e^z - 1) & z \leq 0 \end{cases}$$

A surprising and very successful approach was found using hinge loss. Hinge loss is traditionally used in SVM models and it is defined by the equation: $hinge(x) = max(0, 1 - y_{pred} \times y_{actual})$. It increases linearly when the predictions and labels have different signs, and is zero when the sign is the same. This method was effective in positioning images in a 300 dimensional hyperspace because it ensures that each prediction direction (for all 300) with a wrong sign is punished. In combination with the cosine similarity, this fits the data well as direction is more important than magnitude.

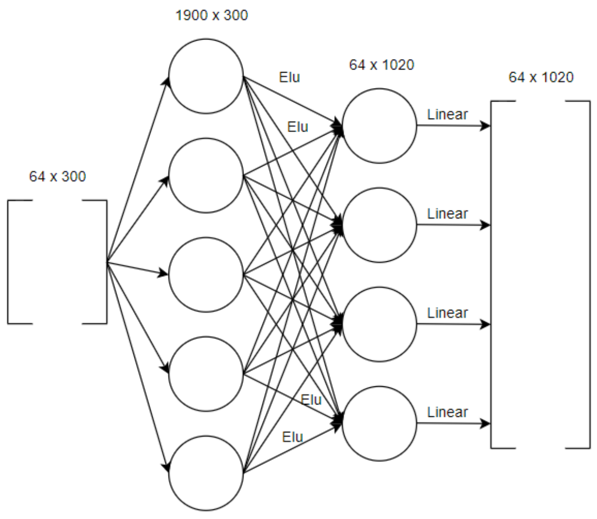


Figure 2: Single Layer Net with Batch Size 64

2.4 Best Hyperparameters

Table 1: Optimized hyperparameters after experimentation

Parameters	Model 1	Model 2
Preprocessing	Word2Vec (Google News)	Word2Vec (Google News)
Loss function	Mean Squared Error	Hinge
Hidden Layer nodes	1900	1900
Hidden Layer input dimension	300	300
Hidden Layer activation function	Sigmoid	Elu (alpha=1)
Output Layer nodes	1020	1020
Output Layer activation function	Linear	Linear
Optimizer	Adam (lr = 0.001, b_1=0.9, b_2=0.999)	Adam (")
Batch size	64	64

3 Results

After 5-fold cross-validation the result submitted on Kaggle is the best performing model found out of the five. The final model is therefore only trained on 8,000 randomly selected data points, leaving out 2,000 validation data points. This ensures the model does not overfit and that the best possible hyperparameters have been selected in each submission.

Table 2: Accuracy Results

Model Description	Accuracy
Bag of Words; Linear Regression	~0.09605
Word2Vec; Linear Regression	~0.11455
Word2Vec; PCA; 1 Layer NN ReLU; MSE Loss	~0.27091
Word2Vec; PCA; 1 Layer NN sigmoid; MSE Loss	~0.32752
Word2Vec; PCA; 1 Layer NN eLU; Hinge Loss	~0.45816

Examples of the model's limitations can be found when looking at wrong predictions of images. These become clear with compound words such as "fire truck" that are written as two words (257.txt), resulting in weaker predictions from model 2 as shown in Figure 3.

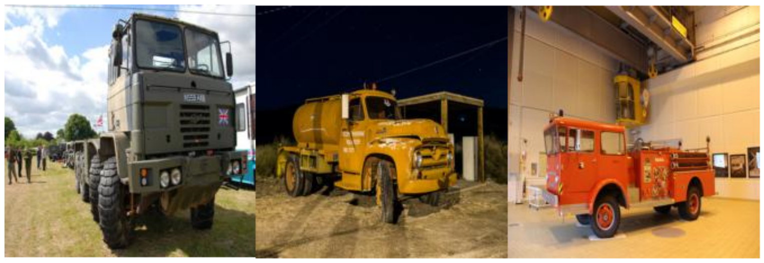
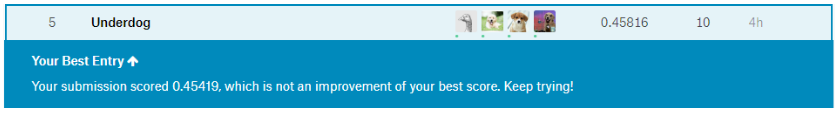


Figure 3: Fire truck description closest images (from left to right)

4 Submission



5 Underdog 0.45816 10 4h

Your Best Entry

Your submission scored 0.45419, which is not an improvement of your best score. Keep trying!

Acknowledgments

We would like to thank the entire Applied Machine Learning course staff for their insight, enthusiasm, and support throughout the semester. We would also like to thank Professor Belongie for keeping us engaged everyday in lecture and always lending his ear.

References

[1] https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92

[2] https://radimrehurek.com/gensim/models/word2vec.html

[3] https://towardsdatascience.com/principal-component-analysis-intro-61f236064b38

[4] https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html