RANK ORDER IMAGE RETRIEVAL USING TEXTUAL DESCRIPTIONS : CS5785 FINAL PROJECT

ABSTRACT

Image search is becoming a key frontier in today's search-based navigation of online interactions. From selecting advertising templates to generating peer-to-peer memes, using text to retrieve images has historically been difficult. This paper will outline a methodology to predict related images based on textual descriptions, and will compare preprocessing strategies using both Word2Vec text vectorization and different bag of words techniques before testing with predictive models.

PROBLEM STATEMENT

Given five sentence textual descriptions, how can we effectively predict a ranked order list of twenty images using different text extraction methods and incorporating pre-trained residual neural networks (ResNet) representation of images?

APPROACH

In our experiments, we utilized the following framework to generate predictions, iterate model types, and optimize performance:

1. **Feature Extraction** from text file descriptions
2. Using ResNet abstractions of image data to **create response vectors** for prediction
3. **Train models** using extracted text features to predict image ResNet features
4. Using a **dissimilarity measure**, compare ResNet Output to the test image data and rank the top twenty closest images
5. **Score overall** performance by final rankings and correct positioning

We will refer to each step of this process when discussing our experiments and motivations.

Note that our approaches rely on a supervised learning models, taking image descriptions and tags and translating them into target ResNet features.

DATA

For the assigned task we were given the following sources of data:

- 12,000 .jpg images (10,000 training samples and 2,000 test samples, randomly selected)
- Descriptions of all images as .txt files of length 5 sentences each
- Tags words of all images as .txt files all of varying length, in the form (category: tag)
- 1,000 feature pre-trained ResNet image abstractions for all images (fc1000 pooled layer)
- 2,048 feature pre-trained ResNet image abstractions for all images (pool5 penultimate layer)
- GoogleNews vector database (each of length 300) used for Word2Vec conversions

EXPERIMENTS

We ran two approaches to compare model performance on Kaggle data and derive insights into image feature prediction.

1. Word2Vec Approach

Preprocessing Utilizing the natural language toolkit (NLTK) package in Python we first preprocessed the text descriptions by removing stop words and lemmatizing the descriptions. This is intended to reduce the feature space dimension from incorporating unnecessary noise, and more accurately capture the sentiment of a sentence. Then using pre-trained GoogleNews vectorization we converted our textual descriptions into word vectors using the Word2Vec library.

Feature Selection We want to determine the right mix of features that can accurately, uniquely, and comprehensively depict an image (or associate a description with an image). This could include a combination of image tags, image descriptions, raw image pixels, or feature combinations. In this approach we use the 1000 ResNet features only.

Word2Vec Optimization We explored the possibility of enhancing the word2vec characterization of a sentence other than taking the mean of all word2vec outputs by word. These experiments were abandoned due to poor performance. Included experiments were: max word2vec vector, peak to peak vector, normalized average vector.

Response Vector Processing The target of our predictive models is to predict ResNet Features; However, how should we represent the ResNet features to optimize our predictions? As a result, the following approaches were taken:

- Using matrix multiplication of the ResNet features and a randomly generated matrix of size n x p to reduce the resulting feature space to p columns
- Using Singular Value Decomposition (SVD) to compress the feature space, taking a pre-specified rank approximation as the target space

Model Selection The following Models were independently used for model training:

- Ridge Regresssion

  Ridge regression minimizes squared error while regularizing the norm of the weights:

$$\beta^{ridge} = \underset{\beta_j}{\text{argmax}} \frac{1}{2} \sum_{i=1}^{p} (y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 + \lambda {\beta_j}^2. \tag{1}$$

- Random Forest Regressor

  A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Our hypothesis was that this model would help provide a simple, intuitive mapping between image features and descriptions.
- Kernel Ridge Regression

  Kernel Ridge combines Ridge Regression with the kernel trick. It learns a linear function in the space induced by the respective kernel and the data.

Scoring Methodology Cosine Similarity measure, commonly used in text comparison algorithms, was selected to provide a score or "distance" between a given text input and potential relevant images.

2. Bag of Words Approach

Our Bag of Words model represents text by defining a "bag" (multiset) of its words, disregarding grammar and word order.

Preprocessing In addition to the preprocessing done above (stop word removal and lemmatization), noun extraction and weighted bag-of-words were implemented to further extract sentence meaning.

- **Noun Extraction:** Using the innate tags marked by the NLTK program, every word not a noun could be stripped away to focus on specific image content.
- **TF-IDF:** The traditional implementation of Bag of Words weights each word in the sentence equally. However, some words may be more frequent (and therefore less important) than others. Using a Term Frequency - Inverse Document Frequency weighting, we can derive important description words of an associated image.

Model Selection Same models as before, but now feature space is much larger. This makes fitting models more computationally intensive and restricts the flexibility of fitting certain models.

- **Ridge Regresssion** this still proved to be the best regressor for Bag of Words implementation. The lambda values found through GridSearch did tend to be higher as expected but overall the added complexity via feature space still proved to be valuable in terms of predictive capabilities.
- **Random Forest Regressor** Random Forest still proved to be inadequate in terms of predictive capabilities for this scenario. We tested various depths and number of trees in order to optimize performance, but overall it did not do as well as ridge regression.
- **Kernel Ridge Regression** Kernel Ridge was an additional attempt to try and improve upon regular Ridge regression. After attempting fitting radial basis funciton (rbf), linear, and polynomial kernels there was no significant change in performance due to these improvements.

RESULTS

1. Word2Vec Approach

Model	Alpha/Depth	Performance
Lasso and PCA	0.01	0.17467
Lasso	0.01	0.13926
Baseline	N/A	0.10734
Random Forest	100	0.06073

2. Bag of Words Approach

Model	Alpha	Performance
Ridge and PCA	20	0.29191
Ridge	10	0.29190
Ridge	50	0.29045
Ridge with TF-IDF	5	0.24606

CONCLUSION

From our results it appears that a bag of words representation of the description data favors image retrieval vs. the word2vec vectorization. From our experiments it appears that the there needs to be a certain level of complexity in order to predict vectors of ResNet features and this seems to favor larger dimensions of data. Even when using techniques such as noun extraction and PCA to reduce the feature space, the improvements to performance were often negligible and often negative relative to a full bag of words representation. Other interesting observations were that robust regressors such as RandomForest did not prove to be useful. Overall performance of RF was way below that of Ridge Regression and this might be due to the complexity of the predicted output (ResNet vectors of length 50).

Future Considerations

- Our approaches naively consider each word in the description sentences to be independent from one another, which provides a simplifying assumption that can be used for ease of modeling. However, overall sentence meaning is lost. A future implementation of a sentence-based model could increase the efficacy of the algorithm by extracting and matching sentence meaning.

- Increasing the flexibility of the output rankings, i.e. not set at 20 image results, could open the possibility of running different unsupervised learning classification algorithms and allowing the model to return the closest cluster. This approach would require an investigation into the distance between clusters, how an input description maps onto the state space, and the number of clusters generated by models.

REFERENCES

[1] Code supplied by Andrew Bennett, used to create initial model of problem statement

[2] NLTK: Natural Language Toolkit, `https://www.nltk.org/`

[3] Mikolov et al., `https://www.tensorflow.org/tutorials/representation/word2vec`

[4] Scikit-learn Packages: `https://scikit-learn.org/stable/`