Final Project

Image Retrieval System

CS5785

December 13, 2021

Ophir Ehrlich (oe44)
Grace Lang (gel53)
Achal Amin (aa859)
Shailesh Mamgain (sm2252)

Abstract

Different approaches to building a large-scale image search engine were investigated. Several models and a variety of data preprocessing methods were used to achieve a large-scale image search engine, which can retrieve the 20 most relevant images given a natural language query. To this end, tags were combined with image features extracted from a pre-trained residual network (ResNet) and text description of the images to train the model. We experimented with feature extraction from the images, explored different preprocessing methods, tried different vectorization methods, and different machine learning models such as Ridge regression, Neural Networks, Partial Least Square Regression, KNN regression and many others. We also used cross-validation to select optimal parameters, and observe the performance of the model on the test set. Through continuously optimizing, adjusting the details of each model and trying different ways to ensemble, we were able to achieve a score of 0.31866 with 28 models.

1. Introduction

Image classification is an important, burgeoning field with ramifications across almost every industry from recreational to mission-critical. A specific subset of image recognition is text-based image search. This technique takes text as the input and returns the best images to the user in a ranked order. This is in many ways an NLP problem; the vectorized words are matched with vectorized pixels to identify objects, thereby opening the door to improve performance from the language side.

MarketsandMarkets, a prominent industry firm, predicts the image recognition industry will grow from $26.6B to $54.0B by 2025 with a 15.1% CAGR.[1] Applications include finding the product code for an item customers see as appealing or identifying criminals caught on camera. Particularly with text-based image retrieval, law enforcement can leverage this technology to save time crossrefrencing witness testimony with video evidence.

2. Problem definition

This paper explores several techniques for carrying out text-based image retrieval with the goal of producing 20 images in ranked order that best match a given query. The dataset used contained 10,000 samples with 4 forms of data associated with them:

1. A 224x224 JPG image
2. A description corresponding to each image.
3. Feature vectors for each image, which were extracted using the Convolutional Neural Network (CNN) ResNet, which were extracted from the fc1000 layers.
4. Tags that were labelled by humans and represent objects in the format of [category:sub-category].

The test data included 2,000 images with the same features as the training set, except there was no correspondence between an image, its features and description and tags.

A baseline model that creates features for the images to predict the ResNet representation was built upon, in order to capture the mapping between the images and their corresponding descriptions and

tags given the training set. Using the features mentioned and surrounding text-based metadata, images were matched with descriptions. Pre-processing, computer vision, TFIDF, synonym matching, and various ML approaches were experimented with to improve the baseline results.

The general framework utilized to predict the top 20 images for a given input query is shown in Figure 1. For this problem, the input query was the 5 sentence description. The natural language text descriptions and images were converted into features and represent the inputs and outputs of the machine learning models respectively. Once the image representation of a description was predicted, the closest images to the prediction were determined. Post processing techniques with the image tags were employed to rerank the top images.

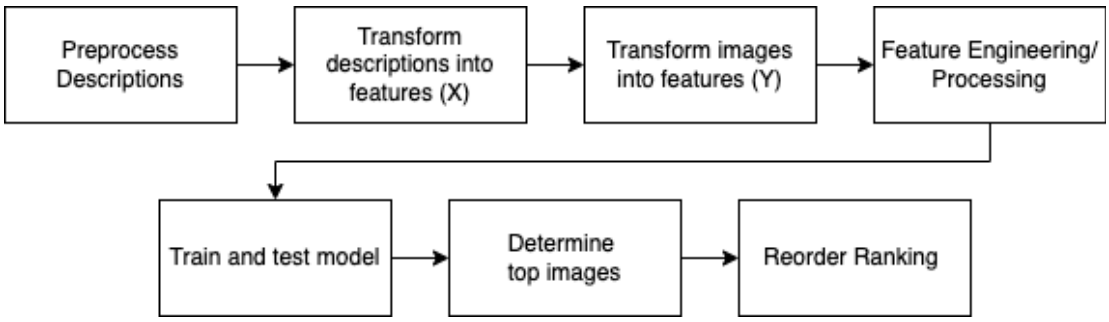


Figure 1: Problem Approach

3. Methods

The image classification end solution is a product of diverse methods attempted, of which some worked and others did not. In all known methods, images and text were pre-processed, normalized, vectorized, reduced, and compared.

3.1 Preprocessing Methods

Preprocessing was attempted both with spaCy and NLTK to find a net 0.02 nominal difference in baseline performance between the two tools. It is difficult to ascertain exactly where NLTK excelled and spaCy did not, since the two libraries have largely the same capabilities but excel at different tasks (i.e. finding synonyms vs lammentization vs POS tokenization). NLTK was chosen due to POS-level synonym methods in the *sysnet* native package that are unavailable in spaCy. Using NLTK, the following preprocessing was executed:

1. Lemmatization - Each word was lemmatized in order to contextualize the description with meaningful words and discern between nouns, adjectives and verbs.
2. Lowercase - We lowercase each description
3. Removing punctuation
4. Removing stop words - Stop words were removed using both NLTK and spaCy with similar results
5. Synonym Finding - Noun synonyms for each description were found using first Parts-Of-Speech tagging and then the *sysnet* [2] package which accepts a word and its Part-Of-Speech to create synonyms.

3.2 Extract Features From Descriptions

Several methods of extracting features from the descriptions were used. The standard baseline model vectorized descriptions using the pre-trained word2vec language model. Word2vec is a tool created by google to represent words as vectors using a sophisticated *bag-of-words* and *skip-gram* approach.[3] In order to represent 5 sentence descriptions as a single vector the baseline model averaged the vectors of all words in the sentences.

Several other custom implementations of bag of words were also used to featurize the descriptions in order to improve the performance of this vectorization:

A. **Bag of Words (BoW)** - We created a model that counted the number of occurrences of each word created from the description. This was transformed to a normalized feature vector. We also tried different experiments, some included the tags' values and some the description alone. However, this method in all experiments proved to be the least robust and gave the lowest validation score. This makes sense considering that word2vec, glove, TF-IDF and BERT are all methods built on top of a bag of words that use different methods that give better accuracy (such as weighting the importance of the words, looking for similarities etc.).

B. **Merging Tags with Description**
Tags were concatenated into a string and were merged with the description strings before vectorizing it using Word2Vec (Figure 2).

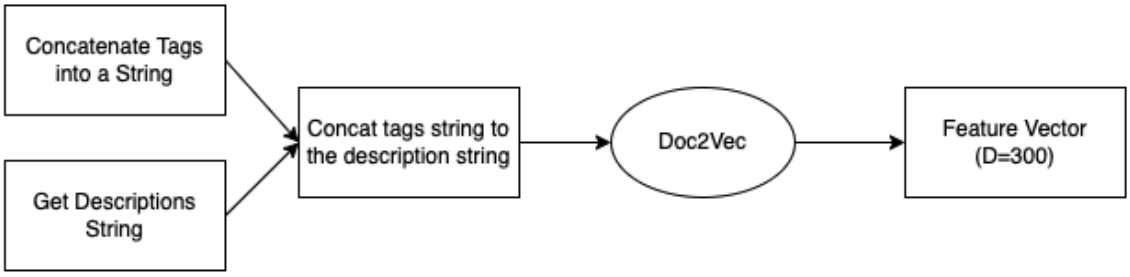


Figure 2: Tags-Description merge flow

The model improved the validation score marginally in the baseline but it actually made the results worse after we applied PCA to the feature vector as well. Instead, the final approach was extracting and appending nouns from the description at the end of the description string. This gave better results than concatenating tags.

C. **GloVe - Global vectors**
Different word vector libraries were researched, other than the GoogleNews corpus for word to vector representation. Vector transformation was implemented using Glove's pre-trained word vectors (linear substructure representation of words using unsupervised learning). Different corpus sizes were experimented, with 400k vocabulary words (D = {50, 200,300}) and with 1.9 million vocabulary words (D = 300). The best validation score was achieved with the 1.9 million vocabulary words (D = 300) corpus, which was also an improvement over the existing score.

D. Weighted word2vec using Tf-IDf

Research was conducted and it was determined that a better way to extract features would be using TF-IDF. Standard Tf-iDF vectorizer was used initially for document sentences. Input words were lemmentized for retrieval across documents. The final approach was to calculate term frequency for each word using Tf-iDF, and then multiplying it with the word2Vec vector for the term. Figure 3 describes the approach[i].

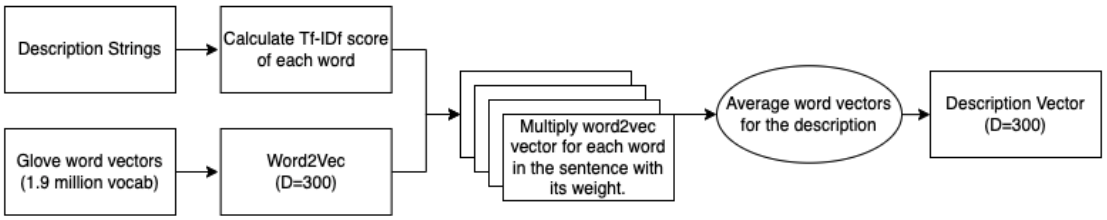


Figure 3: Tf-IDf weighted description vector

E. BERT

In lieu of the word2vec *"GoogleNews-vectors-negative300.bin.gz"* model, two huggingface BERT models were tested: 1) *Bert-base-nli-mean-tokens* and 2) *all-MiniLM-L6-v2.* The *Bert-base-nli-mean-tokens*[3] showed marked improvement over both baseline and *all-MiniLM-L6-v2* scores- it was thus implemented in all approaches.

3.3 Extract Features From Images

Features extracted from the *pool5* and *fc100* layers of ResNet deep-learned CNN represented different characteristics about each image. To expand on this dataset, one hot encoding of image tags was performed to attempt to incorporate new characteristics of the images. This led to 91 new features for the images.

Among the initial approaches was merging tag data with ResNet vectors. Key thesis behind it was to incorporate tag data in the test set to images, as the tags were closely related to the descriptions (Figure 4).

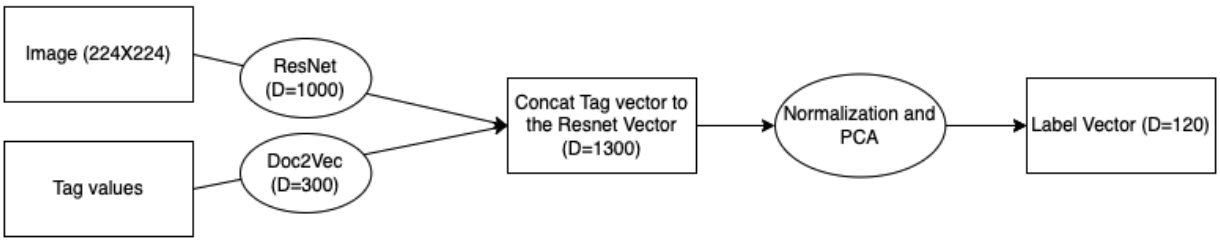


Figure 4: Resnet-word2vec merge flow

.

Another approach to create additional features was to use a different CNN other than ResNet. The VGG-16 CNN was used to accomplish this task. This neural network is used for classification and detection and was pretrained on imageNet data.[4] Features were extracted from the feature layer,

average pooling layer, one fully connected layer. In total, this created 4096 additional features to represent the images.[5]

3.5 Dimensionality Reduction

Due to the large number of features for both the text descriptions and image data, dimensionality reduction techniques were employed to capture the main trends in the data. Principal Component Analysis (PCA) on both the description and image data ensured the models used did not overfit to the nuances of the text/images, but instead conveyed the information that differentiated the data best. In addition, we used PCA in order to reduce dimensionality of the vectorized training set, X. After experimenting several models, we decided on a variance of 0.97.

3.6 Machine Learning Models

3.6.0 Lasso Ridge and Random Forest

Several regression models were used to predict the image representation of the descriptions. Linear regression models Ridge and Lasso were leveraged to trade off bias and variance. Cross validation on both these models allowed for the selection of the regularization parameters that resulted in the best prediction. A random forest model was also used to predict the image representation of the descriptions. In this case, cross validation was used to tune the number of trees, the depth of the trees, and the number of features to use at each split.

3.6.1 Partial Linear Regression Model

After some literature review, it was determined that another way to relate one source of data to another in a cross-modal is by doing Partial Least Squares (PLS). PLS tried to find a multidimensional direction in the X space that explains the maximum variance direction in the Y space. The idea reminds of PCA, but in this case it models the relation between two variables. The regression also maximizes fit and correlation between X and Y, such that each pair of observations or co-occurrences is as strong as possible. We used the PLSR both on the preliminary set of features and both after dimensionality reduction of Y using PCA. Though they have similarities, it seems that the better model was still achieved by using both PLSR and PCA on the Y dataset. This model achieved a better score than the other linear regression models.

3.6.2 Neural net - Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP)[6] from sklearn library is a neural network that serves as an extension of linear classification. Each layer assigns each feature a weight *w* and a constant bias and thresholds it with a non-linear activation function for binary classification. MLP utilizes supervised learning techniques and classifies data that is not linearly separable. After experimenting and training different configurations of this model, we found that the highest performing model included three hidden layers of size 1000.

Several other regression methods in the sklearn library were tested, such as kernel ridge regression, logistic regression, KNN regression, random forest regression, and other types of neural networks. However, these models did not perform well and therefore they were not included in this report.

3.7 Measuring Similarity

After the machine learning models predicted the image representation of the description, the closest images to the predictions were calculated. At first, euclidean distance was used to compute the closeness of the vectors. A second distance measurement, cosine similarity, was leveraged as an alternative to euclidean. Cosine distance is often used to compare similarities between two documents to detect plagiarism.[6] With a similar use case here, cosine similarities were used to detect the closeness of the two vectors by measuring the angle between them.

3.8 Post processing methods

Analysis during the training phase showed the top images were ranked on average in the top 50 after applying the predictive methods and ordering mentioned in the above sections. To attempt to surface the best images to the top of the ranking a reordering algorithm was developed. Figure 5 describes the behavior of the algorithm:

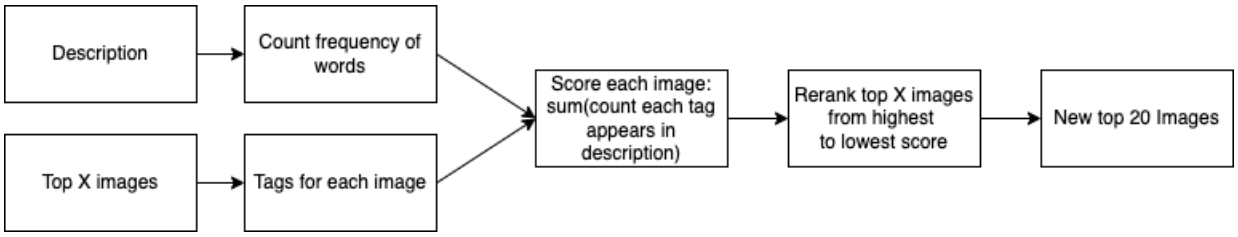


Figure 5: Reranking Algorithm

Essentially, the algorithm reorders the top images X by determining how similar the tags are to the words in the description. The tag structure is hierarchical, there is a category and an object. For example, an image is tagged as “furniture:bed” with furniture as the category and bed as the specific piece of furniture appearing in the image. For the purpose of this algorithm the category and the object were treated as two separate tags to increase the number of tags per image. Using this definition of tags, the test data had an average 6.64 tags per image and training data had an average of 6.03 tags per image. Applying the algorithm, if the word furniture appeared in the description twice, and bed appeared once the score would be 3.

Many variations of this approach were experimented with throughout algorithm generation. The main variations tried were to consider a different number of top images to rerank, and to expand the number of feature tags for an image by generating similar words. Similar words were generated through word2vec’s get_similar method and through wordnet nltk to convert between parts of speech. Additional variations were also applied, such as using SequenceMatcher instead of boolean matching.

4. Experiments

Many of the approaches discussed in the methods section were combined at various times to produce a final prediction. First, PCA on the ResNet data was implemented to reduce 1000 features down to 100, Ridge regression was used and cosine similarity was added to calculate the 20 images that are closest to the prediction. This method was then expanded on by adding the reordering of tags algorithm as a post processing technique.

Various numbers of top images to reorder were experimented with on the validation set. To understand the impact of reordering, the average change in index position and the percentage of the original top 20 that remains in the new top 20 was calculated. Table 1 tells us that on average 72% of the original top 20 remains in the top 20 after reordering when 50 top images were considered. This number decreases as the number of images considered decreases.

Table 1: Reordering Algorithm Impact

Number of Top Images Considered:	Average Change in Position	Average percentage of original Top 20
20	3.10	100%
30	4.61	84.50%
40	6.11	76.92%
50	7.59	72.15%

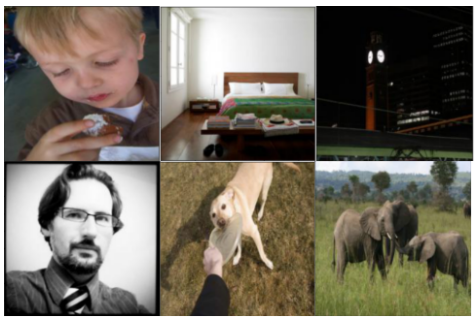
The best performing number of top images when testing on a held-out validation set was 50. This reordering approach was carried forward into future experiments because of the positive impact it had on surfacing top images.

Below is one example of how the approach updated the top image from the 31st position to the 1st position.

Description: 'small child eating donut fed another hand blonde boy looking donut hole boy looking small doughnut powdered sugar face little boy eating powdered sugar covered flood child eats part sugar frosted donut'

Top 6 Original Images:

Top 6 New Images:



It is clear none of the original 6 images match the description. Since the actual image of the boy eating a donut was in the top 50 and had a tag ‘donut’ it propagated up to the top of the ranking.

In Fig. 5 step three where the Count frequency of description words and Tags of each Image are fed to a similarity comparator that prior to SequenceMatcher was strictly binary. The resulting difference was a 0.22 decrease in validation score. This remained relatively true despite toggling the SequenceMatcher.ratio() threshold between 85% and 95%. A small change or improvement- since the motive was to inherently stem words - was expected, but the dramatic validation score decrease led to a move away from this approach.

In addition to experimenting with different characteristics of the reordering algorithm, ways to utilize different machine learning models were also investigated. Ensemble methods to combine the results of Ridge, Lasso, and Random Forest models were experimented with. Each method had an equal vote in predicting the top 20 images. The final ranking was composed of the 20 images with the most votes. Ultimately this method resulted in validation scores that were worse than the individual models themselves, so it was not pursued further. Since these models both individually and combined did not have a strong performance, the next machine learning models experimented with were partial least square regression (PLSR) and neural networks. Overall these models performed better and the results are summarized in the next section.

Other experiments were carried out to try different combinations of image features. ResNet was done on it’s own, as well as concatenated with VGG-16 features. Experiments on the validation set showed that on its own, VGG-16 features performed worse than ResNet. When concatenated with ResNet and PCA applied, the performance stayed relatively close to that of when just ResNet features were used. Ultimately the VGG-16 features had no significant contribution to the performance of the model.

5. Results

Table 3 shows the summary of the models that were run and their public leadership board score. Each model uses a subset of the methods described earlier, and results varied as a subset were applied.

Table 2: Summary of Models

Model used	**Test Set Score**	**Notes**
Bag of words + concat tags + PCA on ResNet + PLSR + Cosine Similarity + reordering	0.00886	This model did much worse than expected. Adding the tags and reordering made the data overfit, but even in the second try without concatenating the tags the model did not do well
BoW (without tags) + kernel ridge regression	0.0886	Even without the tags, the simple bag of words model did not do well compared to the

		other vectorization methods.In addition, kernel ridge did a little worse than PLSR.
Concat tags to description + Neural net + PCA on Y + PCA on X + Cosine Similarity + reordering	0.15039	Concatenating the tags and then reordering with that data decreased the accuracy due to overfitting.
Concat tag vectors to Resnet vector (Y) + PCA on Y + Cosine similarity	0.20315	Concatenating the tags decreased the score since most likely due to overfitting (usage both in the beginning and when reordering at the end).
PCA on ResNet data + Cosine Similarity + Ridge	0.20383	This simple approach performed relatively well for its limited complexity.
PCA on ResNet data + Cosine Similarity + Ridge + Reorder top 20	.20825	Reordering the top 20 images did not help the performance of the model.
PCA on ResNet data + Cosine Similarity + PLSR	0.22073	PLSR performed better than Ridge Regression.
PCA on VGG and Resnet Features + NN + reordering top 50	0.23758	Combining VGG and ResNet features decreased performance.
Reorder top 60 tags + one hot encode tags as image features + Cosine Similarity + Ridge	0.26595	One hot encoding tags did not improve performance.
PCA on ResNet data + Cosine Similarity + Ridge + reordering top 50	0.29435	Adding the reordering of the top 50 brought the model performance up from 0.20383.
BERT vectorization + Neural net + PCA on Y + PCA on X + cosine similarity + reordering	0.29711	This model replaced word2vec with BERT for the embeddings and applied the new vectorization on existing Neuralnet, PCA, cosine, and reordering.
PCA on ResNet data + Cosine Similarity + PLSR + reordering top 50	0.29755	Updating Ridge with PLSR resulted in a very small improvement.
PCA on VGG and Resent (50), PCA on descriptions - 180 + new conditions for reordering +	0.30169	This model attempted to expand the tag data to more than just nouns, to encourage

new parts of speech to tag data		more matches with the descriptions.
Tf-iDF weighted feature + Neural net (MPL) + reordering top 50	0.30466	One reason for low improvement could be the overarching role of feature reduction on the description set.
Neural net (MPL) + PCA on Y + cosine similarity + reordering top 50	0.30323	Neural net without Tf-IDf performed as good as one with GoogleNew word to vector.
Neural net (MPL) + PCA on Y + PCA on X + cosine similarity + reordering	0.31296	This performed very well when adding dimensionality reduction on X
Glove word2vec + PCA on X + PCA on Y + Neural Net (MLP) + Cosine similarity + reordering	0.31866	This was implemented with a 400k words dataset and a 1.9 million words dataset from Glove. The latter worked best.

Overall the best performing model used a neural network, cosine similarity, GloVe to vectorize the descriptions, PCA on image data to reduce dimensionality and reordering of the top 50 ranked images.

6. Conclusions

Applying different methods to the text-based image retrieval problem yielded results along a wide spectrum. A custom implementation of bag of words proved to be a weak method to featurize text descriptions on its own for this problem. More advanced techniques such as word2vec, GloVe, and BERT were necessary to capture the complexities of natural language. A method to reorder a subset of the closest matching images was useful in improving the accuracy of the model. However, through validation and testing a discrepancy was seen in the degree to which the score improved. Because the reordering is based on image tags labeled by humans and only contain a dictionary of 91 words tags alone are not always enough to surface images that are ranked just outside the top 20.

7. Future work

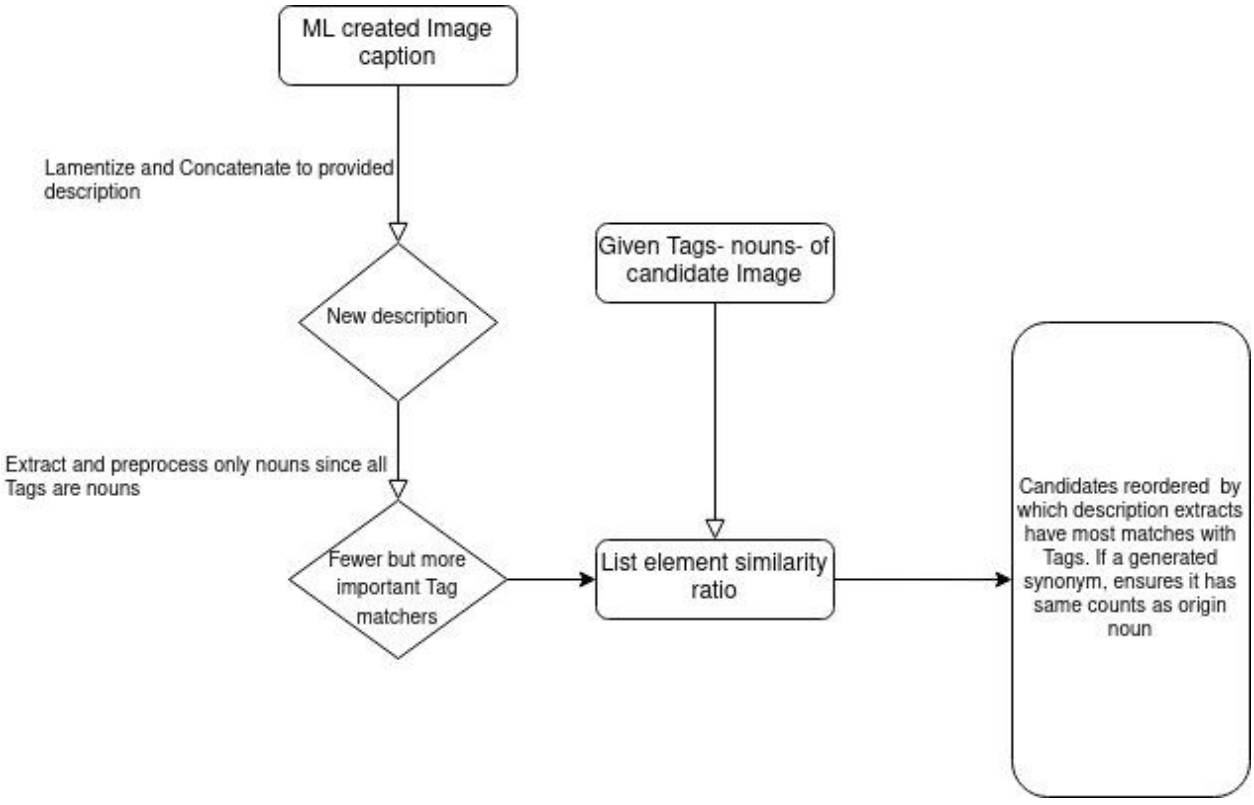
7.1 Auto-Caption Generator (attempted and abandoned)

After a literature review, various Do-It-Yourself methods for machine generated captions were found. The idea is to use an ML generated caption in conjunction with the provided description to accomplish the following:

A. First, check that the generated caption is somewhat accurate by cosine similarity or another text comparator (referenced in the appendix below) with the provided description. The provided description was cleansed to match output of caption

B. Second, **if the generated caption was relatively accurate,** extract the *set()* of nouns between both the description and caption. This *set()* function would be applied on the following existing line of code in the program- and referenced in Github- after caption nouns were appended:
des = get_description(j,train_set).split(" ") where *des* is the list of nouns from given captions.
C. The goal is to attain a better similarity score between tags and description extracted words if the description has new, accurate synonyms found in the process of ensuring a generated caption was similar to existing descriptions.

    a. Various approaches were attempted using both corporate APIs (IBM) and adjusted OSS, but due to the scale and complexity of the project, a Progressive Loading method[7] was used on the Flickr8k dataset[8] which let the images be iteratively and quickly trained with a slight performance trade-off.
    b. However, the implemented method still failed to compile on the test set due to issues with Keras despite a Google Collaboratory environment and the process complexity. Fig. 3 illustrates what was aspired:



7.2 Sub-category hard-coding

It was found that the Tags always had *'person'* as the noun, regardless of whether it was the category or sub-category. Concatenating any post-processed description output with the *'person'*

noun if it contained *['girl','boy','male','female"woman','man']* another synonyms that did not appear previously would make the Tag - Description Noun scores dramatically more accurate within this problem statement. This concept may be expanded to every other noun category that has consistency in the Tags.txt dataset, such as *furniture* and *animal.*

7.3 Sysnet Noun Synonym for Tags

Word2vec.most_similar was used, ineffectively, in creating synonyms for the Tags. word2vec does not accept a Parts-Of-Speech filter for synonyms, so it is likely more inaccurate. However, since all Tags are Nouns, the *sysnet* package could be used with greater accuracy here.

8. References

1. https://www.marketsandmarkets.com/Market-Reports/image-recognition-market-222404611.html
2. https://www.geeksforgeeks.org/get-synonymsantonyms-nltk-wordnet-python/
3. https://code.google.com/archive/p/word2vec/
4. https://neurohive.io/en/popular-networks/vgg16/
5. Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks. Russel D. Reed and Robert J. Marks (1999)
6. Partial Least Squares Regression. Henry Abdi. *The University of Texas at Dallas.*
7. https://personal.utdallas.edu/~herve/Abdi-PLS-pretty.pdf
8. https://towardsdatascience.com/image-feature-extraction-using-pytorch-e3b327c3607a
9. https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens
10. https://www.sciencedirect.com/topics/computer-science/cosine-similarity
11. https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/
12. https://towardsdatascience.com/progressive-learning-and-network-growing-in-tensorflow-e41414f304d2
13. https://www.kaggle.com/adityajn105/flickr8k/activity
14. https://nlp.stanford.edu/projects/glove/
15. http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/

9. Appendix

A1

Libraries Used:

- Sklearn
- Numpy
- Pandas
- Matplotlib
- Nltk
- Spacy

- Gensim
- Os
- Tensorflow
- Keras
- Pickle
- Re
- Difflib
- SequenceMatcher
- Collections
- PIL

A2

Github Repository with source code:
https://github.com/ShaileshMamgain/Finals/ (See individual branches for more details)