AML Final Writeup

**Abstract**

In this paper, we proposed a solution to deal with an image searching problem….

**1 Introduction**

In this paper, we are trying to build an image searching engine that can search for relevant images given a natural language query. For instance, if a user types "dog jumping to catch frisbee," Our system will rank-order the most relevant images from a candidates’ pool.

**1.1 Problem definition**

During training, we have a dataset of 10,000 samples. Each sample has the following data available: A 224x224 JPG image. A list of tags indicating objects appeared in the image. Feature vectors extracted using ResNet. A five-sentence description.

During testing, our system matches a single five-sentence description against a pool of 2,000 candidate samples from the test set. Each sample has: A 224x224 JPEG image. A list of tags for that image. ResNet feature vectors for that image.

Output: For each description, our system will rank-score each testing image with the likelihood of that image matches the given sentence. Then the system returns the name of the top 20 relevant images, delimited by space.

For Evaluation: There are 2,000 descriptions, and for each description, we compare against the entire 2,000-image test set. That is, rank-order test images for each test description. Then we use MAP@20 as the evaluation metric. If the corresponding image of a description is among the algorithm's 20 highest scoring images, this metric gives a certain score based on the ranking of the corresponding image as follow.

**1.2 Related work**

**2 Model architecture**

**2.1 Overview of the pipline**

We are using a KNN approach to solve the problem. We split the task into 4 steps:

(1) data preprocessing

(2) building feature vectors for test queries and candidates

(3) compare pairwise similarity between query and every candidates

(4) retrieve the top 20 nearest neighbor from the candidates pool

**2.2 data preprocessing**

**2.3 building features**

We built a dictionary of unique tags from tag data of the training set. Then we built the feature vector for the test data and candidate data through Bag of Words model(??). By now, we have mapped the description data of a query to tags.

**2.4 pairwise similarity comparison**

After the last step, we got the feature vectors for the test data and candidate data. Then we compute the cosin distance or Euclidean distances between a test sample and all the candidates. After this step, we simply sort the distance and find the top 20 neighbors as the output data.