# Joint Learning of Image and Text Features through Ensemble

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

The challenge presented is to create a search algorithm that matches a caption/description to an appropriate image. Ten thousand training examples are given, and two thousand are provided through Kaggle for validation. To attain the highest $map@20$ score, multiple machine learning model were experimented with. Partial least squares was determined to be the best primary method - surpassing the performance of autoencoder, neural networks, support vector machines, and kNN. To extract the highest performance, the outputs of several methods were combined together as an ensemble algorithm. The best score achieved was 0.41977.

## 1 Introduction

### 1.1 Challenge Overview

The goal of this challenge is to search for relevant images given a natural language query. An example given is that if a description states "Dog jumping to catch a frisbee" then a series of images should be returned, ranked most likely to represent the dog jumping. Several additional samples of images and their captions are shown in Figure 1 below.

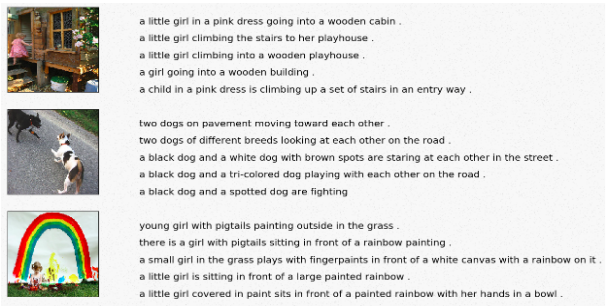


Figure 1: Examples of images and their associated captions (1)

To create a machine learning algorithm training data is provided that includes the raw images, features for each image as extracted from the ResNet algorithm, image tags, and the corresponding image description. Within the testing set each medium of data is again provided, but the descriptions no longer correspond to the image. During testing time, for each description 20 images should be generated ranked by their likelihood to match the description. The metric used to determine performance is match@20, the equation given below where i is the rank of correct image. If the correct image is not in the recommended 20 a score of 0 is given.

$$map@20 = \frac{20 + 1 - i}{20}$$

The testing procedure is automated though the online Kaggle platform. A maximum of four submissions can be made per day. Accompanying the model, a report is to be generated in the style of a conference paper that becomes peer reviewed. The overall grade for this challenge is assessed both on model performance and report quality.

### 1.2 Background Work

The problem of matching an image to a caption is a merger between the larger fields of image recognition and natural language processing. For the beginning of the 21st century the field of image recognition was akin to most other fields, in that it required careful feature selection and could then be classified using a variety of algorithms. With the release of imageNet in 2009, researchers had the resources to train deep neural networks - which quickly surpassed the performance of previous models. Current state of the art models include ResNet, which has over 100 layers and was used to produce the image features in this challenge (4). Alternatively, the field of natural language processing has been continuously improved upon since the advent of computers in the 1950's. However machine learning approaches revolutionized the field starting in the 1980's by applying basic statistical models (5). Neural networks in the past decade have caused another revolution, with the transition of word embedding techniques using simple bag of words approaches into context aware word2vec (6). While the historic poor performance in each of these fields likely prevented their merger, the exceptional performance of modern deep learning approaches has lifted this blockade.

The inception of the image caption problem occurred when the Microsoft COCO Caption dataset was released in 2013 (2). With over one and a half million captions describing 330,000 images the dataset provided a rich resource for training the resource heavy deep learning algorithms. Around this time Yahoo released the Flickr datasets, which provided another complementary source of images with associated captions. The main type of problem that emerged was developing captions that matched the image, inverse to the problem tackled herein. Taking advantage of the well developed model for each feature, the typical framework for solving this problem was to process the image into rich features and then submit those into a text generation algorithm - as shown in Figure 2. The image understanding part uses independent image recognition, convolution neural nets and the text generation part uses recurrent neural networks, such as long short term memory. Many examples using these methods are published on arXiv, and Google has even released a website that offers its algorithm img2txt (7).

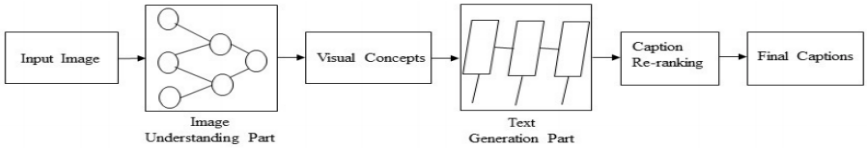


Figure 2: The typical process of assigning a caption to an image (3)

Alterations need to be made for this process flow to be turned from a captioning machine into a matching machine. Instead a linear flow of data, the image feature selection and text generations must be considered simultaneously. A common combination is shown in Figure 3.

Important caveats exist when considering this problem practically on the the large training set scale, and on the scale for this class. First is that the ground truth captions themselves are not entirely accurate. The caption one human gives the image may be rather different than the caption another gives, and both captions can still be considered correct by a third human. This variance in the actual caption for an image can be seen within the precision recall curves shown in Figure 4. A second problem that emerges is that all of the methods described so far use deep learning on thousands of images. These models are likely highly tuned and carefully constructed, realities that are difficult to replicate on a relatively smaller dataset in only one week. While few methods described in literature do not take a deep learning approach, it is likely that relatively simpler methods will perform better.

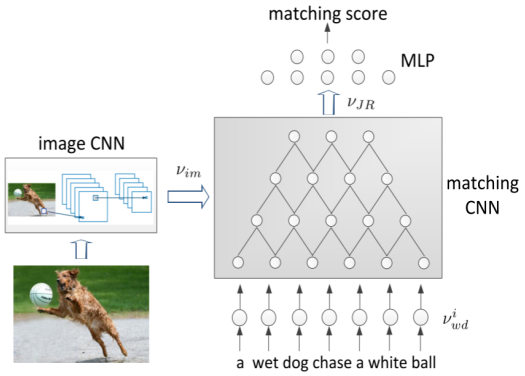


Figure 3: The typical process of matching images to captions (8)

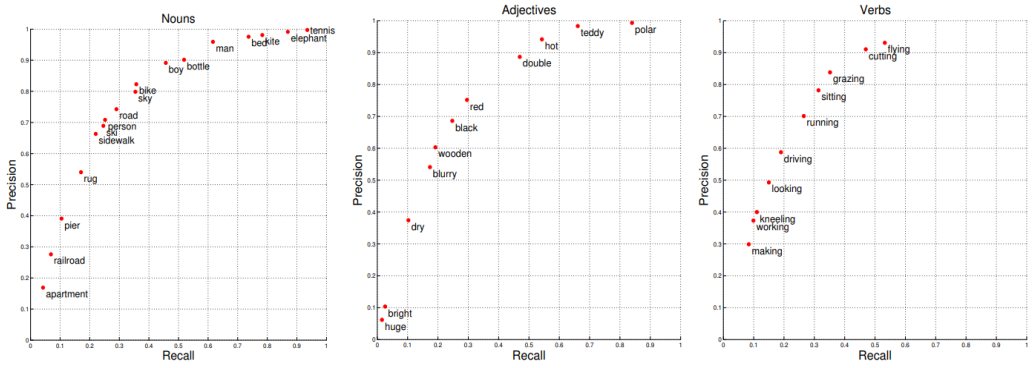


Figure 4: Precision recall curves for the correlation of a series of human judging another's image caption (2)

## 2 Methods and Experiments

To build our final model, we first tested a variety of feature extraction methods and image retrieval algorithms proposed by previous studies to perform similar tasks. After examining the performance of each model, we selected the best performing ones to build an ensemble model.

Section 2.1 describes the feature extraction methods we tested, Section 2.2 to 2.5 image retrieval algorithms, ensemble models and our final ensemble model in the section 2.6.

### 2.1 Feature Extraction

#### 2.1.1 *Feature extraction from the images.*

Feature extractions from the images was already provided as outputs of the ResNet algorithm. The feature matrix from layer fc1000 contained 1,000 features capturing the presence or absence of 1,000 different objects in the image.

The matrix from the pool5 layer contained 2,048 features. Since this is extracted from the last convolutional layer of Resnet, it captures a wide range of features present in the image, in addition to objects.

In our model, these feature matrices were subsequently reduced for use in several applications. Principal component analysis and latenet dirchlet analysis were both methods used to provide this reduction.

#### 2.1.2 *Feature extraction from the descriptions.*

Since the descriptions are natural languages, they may contain significant amount of noise and useless information. To extract the most useful information, we created a bag-of-words model for the text descriptions. We performed the following preprocessing steps:

1) Remove punctuation → 2) Lowercase-converting → 3) Remove stop-words → 4) Lemmatization → 5) Optional: Filter for nouns or non-nouns → 6) Create a dictionary of processed words from the training set

Then, we represented each description as a feature vector of words in the created dictionary, whose entries are TFIDF transformed counts of each word in each sample.

In parallel, we also tried to train a dense word vector using negative sampling as the Google Word2Vec, but we found that instead of training a task-specific word embedding it is better to utilize pre-trained dense word vectors such as GloVe_6B_300d which is commonly used in Natural Language Processing. We average the 300 dimension word vectors in both descriptions to form a description vector. We noticed there are some unseen words in the description that were likely caused by misspelling for example *"chirch"* which probably was misspell of word *"church"*. Nonetheless, we found these were rare such cases and we simply omitted them.

*Explanation: TFIDF transformation normalize the word count by dividing frequency of a word by the frequency of documents containing the word. This normalization reduce the impact of high-frequency words, which are believed to be less informative than less frequent words. GloVe 6B is trained on 6 billion tokens and is widely used. It is suitable for the task of text feature extraction in this challenge as most of the descriptions contain commonly used words that were not specific to a genre of text. We chose the 300 dimension pre-trained word vector to capture the meaningful text information in the description.*

#### 2.1.3 *Feature extraction from the tags*

We created a bag-of-words model for the tags associated with each image. We performed the following preprocessing steps:

1) Remove punctuation → 2) Lowercase-converting → 3) Remove stop-words → 4) Lemmatization → 5) Create a dictionary of the processed tags from the training set.

Then, we represented the set of tags for each image as a feature vector of words in the created dictionary, whose entries are either 0 or 1, indicating the absence or presence of a tag, respectively.

GloVe pre-trained word vector was also used to create dense tag vectors as described in the previous section. We also noticed that some of the training and test images were missing the tags, in that case we simply treat them as 0 vectors as there is little ways to impute the missing tag values.

### 2.2 Tag-based Image Retrieval

The tag-based image retrieval model seeks to harness the textual information contained in the tags annotating each image. We first map the bag-of-words representation of descriptions (2.1.2) to a bag-of-words representation of tags (2.1.3) using linear Support Vector Machine (SVM). Specifically, we trained separate SVM classifiers to predict the presence or absence of each tag feature from bag-of-words representation of descriptions.

We used mahalanobis distance to calculate the similarity between our input and each test sample. As an independent model, we report the images corresponding to these 20 most similar tags in the test set as our output.

*Explanation: We used Mahalanobis distance because it captures the covariance of features. Since certain objects often appear together in images, we assume tag features are dependent. Using mahalanobis distance accounts for the covariance in our features.*

### 2.3 Image Feature-based Image Retrieval

The image feature-based image retrieval model attempts to map image features and text descriptions into a common subspace. After mapping, the model calculates the similarity between a mapped input

and all mapped images. We generated models using image features extracted from pool5 layer and fc1000 layer of Resnet separately.

Previous studies proposed various methods to perform subspace learning with image and text data: deep learning methods, Partial Least Squares (PLS) (11), Canonical Correlation Analysis (CCA) (9), and Bilinear Model (BLM) (10). We will focus on the first three methods in this paper.

### 2.3.1 *Partial Least Squares*

Partial least squares (PLS) models the relationship between two set of variables by finding latent variables that explains the most covariance between them. PLS can be extended to regression problems (PLSR). Specifically, PLSR maps one set of variables into a latent subspace and regresses the other set of variables on these latent variables.

In our case, we used PLSR to map the image features into the text space, and then calculated the similarity between an input text and all mapped images.

a) PLS using fc1000 layer features

   In this model, we used PLSR to map image features extracted from the fc1000 layer to the bag-of-words model (BOW) for descriptions (see 2.1.2). The BOW model contains only nouns. We chose to include only nouns because fc1000 features are trained to identify objects in natural images.

b) PLS using pool5 layer features

   In this model, we used PLSR to map image features extracted from the pool5 layer to the bag-of-words model (BOW) for descriptions (see 2.1.2). The BOW model contains all the informative words regardless of their part-of-speech. We chose to include all words because pool5 features capture a variety of features in addition to objects, which may reflect verbs, adjectives etc. in the descriptions.

### 2.3.2 *Canonical Correlation Analysis*

Canonical correlation analysis (CCA) aims to find linear mapping between two sets of vectors in low dimensions at which the correlation structure between the two vectors are maximized. We used the CCA to jointly mapping the ResNet image features or the tag features with the description vectors. Similar to the PLS, we used fc1000 and pool5 layer features to conduct the CCA between image and text (GloVe 300d vector) using 400 and 750 dimensions the same as PLSR.

We also performed CCA using tag features with 100 dimensions. In both PLS and CCA, we chose these numbers of dimensions as a fact based our observation of the PCA using fc100, pool5 and tag features. As depicted in Figure 5, we observed that the variance explained by each Principal Component generally dropped quickly. In order to capture good amount of variances we select 100, 400 and 750 PCs to surpass the "elbow" of the decreasing eigenvalues.

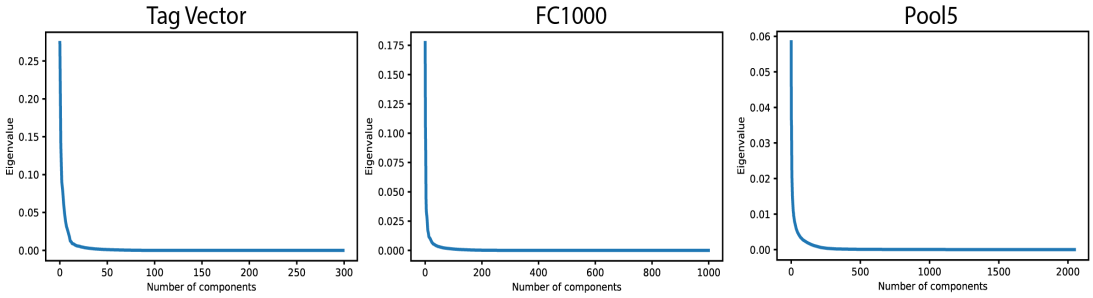


Figure 5: PCA of tag, fc1000 and pool5 features.

### 2.4 Description-based Image Retrieval

In order to use the descriptions as an retrieval apparatus, they were first transformed into numeric vectors using several methods. One of the more basic methods, bag of words has previously been described (see 2.1.2). However, the bag disregards possible phrases or n-grams. To encode these semantic relationships severalethods were employed, including: the word2vec algorithm applied to the more unique words of each sentence, hashing and tokenization algorithms from the keras package, and the sent2vec algorithm. The ultimate product from each method is the same, a matrix of descriptions with multiple numerical features attempting to best correlate with the true meanings of the descriptions.

#### 2.4.1 *k-Nearest Neighbors*

Under the k-Nearest Neighbors approach each testing description is compared to each of the training descriptions. Through cross validation cosine distance was determined as optimal. The k closest images that corresponded to these training descriptions were chosen, and their vectors were combined in a weighted average. Lastly this representative training image was compared to all testing image, and the closest by cosine distance were ranked highest.

### 2.5 Joint Image-Description-Tag Matching

An intuitive approach to the problem might be to use all possible information in one model, thereby taking advantage of all possible signals.

#### 2.5.1 *Two Branch Neural Network*

Published methods have shown that by combining both the descriptions and images into a common, multimodal space can be an efficacious method for description matching. One of the simplest join method is the two branch neural network, in which the descriptions and images are processed in a separate neural network, their outputs combined in the last step in an inner product. Training to produce a maximum margin in this inner product will create a classifier (shown in figure XX.).

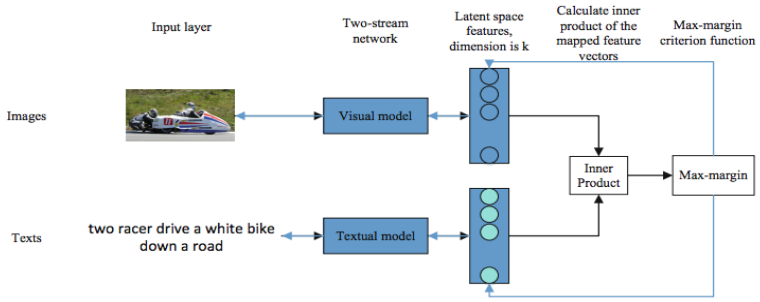


Figure 6: Schematic of a Two Branch Neural Network for image-description matching.

#### 2.5.2 *Autoencoder*

Autoencoders are a specific structure in neural network that is able to learn to reconstruct the original input data. It is widely used for denoising and it has a nice feature that the encoder can act as a dimensional reduction tool. Here we used it as a transfer learning tool that transfer the features in image domain to the text domain.

The structure of the autoencoder we used is shown in Figure 7 below and we used Keras to build and train our autoencoder. We concatenated all the fc1000, pool5 and tag features (GloVe 300d) to form a feature vector of size of 3348 as input to the autoencoder. The autoencoder has 5 layers, each contain 1024, 512, 128, 256 300 neurons with "relu" activation function except for the last layer that we used "linear" activation function to produce a 300 dimension which corresponds to the text description vector. We used batch size of 256 and divided the training data into 9000 for autoencoder training and 1000 for validation.

We monitor the validation loss that uses an early stopping strategy for over-fitting. In addition, we also tried adding "dropout" layer to prevent over-fitting. However the performance of this autoencoder is not as good as our other model especially PLSR. We observed that the validation loss stopped decreasing after first 20-30 epochs. This may be caused by the fact that we used the linear activation function in the final output layer and *mean-squre-loss* as the loss function. Since the description vectors we generated using GloVe are most likely normally distributed, it may be better suited for Variational Autoencoders to learn the underline Gaussian distribution parameters.

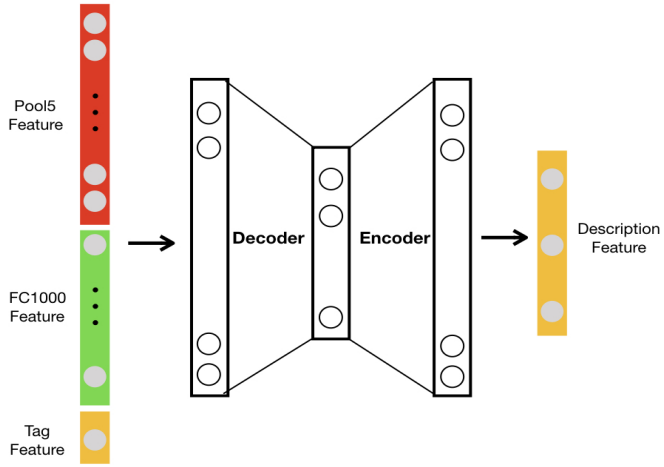


Figure 7: Autoencoder Structure

### 2.6 Ensemble

After we trained each single model in Table 1 as described above, we performed ensemble of different model combinations. The first ensembling method we tried simply averaged the 0-1 normalized similarity score from every model and ranked the images accordingly. The second ensembling method tested different linear combinations of different models to generate the final similarity score.

## 3 Results and Discussion

The testing $map@20$ score for each of the models described in the methods section is shown in the table below.

Several key observations can be made from the table of Kaggle score results. The first is that the PLSR methods works far better than any other. The methods that do not work as well can be split into two groups. The first are those that cannot capture the high dimensionality within the data, including KNN and tags only methods. The second are methods that are complex and thereby need large amounts of tuning in order to perform optimally, including 2-stream NN and the autoencoder. PPLSR takes advantage of both groups of methods - it has few parameters that need tuning, but is still advanced enough to capture a large proportion of the data's variance.

Even though the PLSR methods work well individually, they perform even better when ensembled. In general, ensemble methods can provide this added boost of performance by taking advantage of each method's strength in understanding the variance in one aspect of the images or text. For example, the KNN may be able to determine the rare pictures of zebras well, but is not very proficient at separating pictures of people. Yet adding the KNN to several methods will create a majority vote, similar to the KNN process itself, that hopefully promotes the correct answer while the incorrect answers wash out as weak noise.

The ability of ensembles to capitalize on several methods is ultimately limited by the individual strengths. A weak method ensembled with a strong method will lead to mediocre results. This princi-

Table 1: Model Performance on Kaggle

| Models and Scores | |
|---|---|
| Method Name | Kaggle Score |
| SVM with description and tags (1) | 0.23019 |
| GloVe PLSR with description and tag (2) | – |
| GloVe PLSR with description and pool5 (3) | 0.32273 |
| GloVe PLSR with description and fc1000 (4) | 0.34572 |
| GloVe KNN (5) | 0.22455 |
| KNN (6) | 0.10812 |
| 2-stream NN (7) | – |
| Autoencoder (8) | 0.12824 |
| GloVe CCA with description and tag (9) | – |
| GloVe CCA with description and pool5 (10) | – |
| GloVe CCA with description and fc1000 (11) | – |
| Ensemble of methods 1-7 with average distance | 0.32809 |
| Ensemble of methods 1, 3 and 4 with average distance | 0.37008 |
| Ensemble of methods 9, 10 and 11 with average distance | 0.18990 |
| Ensemble of methods 2, 3 and 4 with average distance | 0.40332 |
| Ensemble of methods 1, 2, 3 and 4 with weighted distance | **0.41977** |

ple leads to the results shown in the table, with the ensembles that include all methods performing worse than those which just include the partial least squares. Therefore, the ensemble method that uses the PLS individual methods and SVM and tag-based method is our final model for submission and review.

We noted that before ensemble our best model only hit the 0.32-0.34 Kaggle score, but after ensemble we achieved Kaggle scores between 0.37-0.42 which exceeded the best performed single models. For visual examination, we showed an example below (Figure 8), the description contains actions like *leaping* and *catching*.

If we were using PLS with the tag feature alone, we saw that none of these images captures the action term and specifically in third figure in row 1, there is a man with *white frisbee* which matched the descriptions. However if we included the image features from fc1000 and pool5 layers, we saw that now most of the images revealed the action term *leaping* or *catching*. This example demonstrated that by just using the tag feature, it is possible that we only get the static description such as *white frisbee* but including the pool5 layer we can further capture the action descriptions. This observation explains why ensemble model using fc1000, pool5 and tag features exhibit superior performances. The results from ensemble models demonstrated that aggregating knowledge and crowd sourcing from multiple learners have an advantage in jointly matching between image and text features.

Similar to Boosting and Bagging, our result showed that by aggregating the different models, ensemble methods outperform the its constitutional base learners. Our final model consist of ensemble using SVM and Partial Least Square using tags (weighted by 0.6) and Partial Least Square Regression using all fc1000 and pool5 features (weighted by 1) as shown in Table 1. Our final model reached a test set performance map@20 score of **0.41977** on Kaggle.

## 4 Conclusion and Future Work

Through building a search engine that matches an caption to an image, we have learned principals of natural language processing, such as bag of words, and image analysis, such as the ResNet algorithm. In a broader sense, this project has also proven that the best model for a machine learning problem should have enough complexity to capture the variance in the data, but not too much complexity that it cannot be easily tuned. If the Kaggle competition were longer than a week, some of these more complex methods would be attempted. For example, there are publications that show correlation latent Dirichlet analysis can perform well for image captioning - yet there is no simple model available for implementation. Nevertheless a final score of 0.4, indicating that the correct image is


Figure 8: Ensemble Prediction Example

reproduced in approximately the top 10 for every caption is an impressive achievement. Especially when considering the time frame, and the relative experience of the programmers.

## Acknowledgments

We thank the TAs Ryan Benmalek and Valts Blukis, and Instructor Prof. Serge Belongie for their very helpful office hours and insightful lectures.

## References

[1] Yan, F., Mikolajczyk, K. Deep Correlation for Matching Images and Text. The Computer Vision Foundation, 2015.

[2] Chen, X., Fang, H., Lin, T., Vendatam, R., Gupta, S., Dollar, P., Zitnick, L. Microsoft COCO Captions: Data Collection and Evaluation Server. ArXiv, 2015.

[3] Hossain, Z., Sohel, F., Fairuz, M., Laga, H. A Comprehensive Survey of Deep Learning for Image Captioning. ArXiv, 2018.

[4] He, K., Zhang, X., Ren, S., Sun, J. Deep Residual Learning for Image Recognition. ArXiv, 2015.

[5] Jones, K. Deep Residual Learning for Image Recognition. International Encyclopedia of Linguistics, 2001.

[6] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. Distributed Representations of Words and Phrases and their Compositionality. Proceedings of NIPS, 2013.

[7] Vinyals, O., Toshev, A., Benigo, S., Ehran, D. Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge. ArXiv, 2015.

[8] Learning to Link Images with their Descriptions. Heuritech Blog, 2015.

[9] D. Li, N. Dimitrova, M. Li, and I. K. Sethi, "Multimedia content processing through cross-modal association," in International Conference on Multimedia. ACM, 2003, pp. 604–611.

[10] A. Sharma, A. Kumar, H. Daume, and D. W. Jacobs, "Generalized multiview analysis: A discriminative latent space," in Computer Vision and Pattern Recognition. IEEE, 2012, pp. 2160–2167.

[11] R. Rosipal and N. Kramer, "Overview and recent advances in partial least squares," in Subspace, latent structure and feature selection. Springer, 2006, pp. 34–51.

## Utilized open source libraries

GloVe 6B 300D

NumPy

Pandas

SciPy

NLTK

Keras

sklearn.featureextraction.text.TfidfTransformer

sklearn.svm.LinearSVC

sklearn.crossdecomposition.PLSRegression

sklearn.neighbors.KNeighborsClassifier

sklearn.modelselection.KFold

sklearn.ensemble.RandomForestClassifier

sklearn.cross_decomposition.CCA

sklearn.decomposition.PCA