Textual Query Image Retrieval System Based On An Ensemble Model

Abstract

In this paper, we will introduce how to retrieve most relevant images in a image set given sets of descriptions. We have trained and experimented with 4 different types models to face this challenge, which are Bag of Words(BoW)+K-Nearest Neighbors(KNN), Neural Network(NN) and Partial Least Square Regression(PLSR). We also try to ensemble different models to obtain robuster and more powerful models. Finally, the best performance we achieved is the accuracy score of 0.3695.

1 Introduction

This competition challenges us to retrieve target image from natural language descriptions in sentences. We are given a testing image set and train image set, the fc-1000 and pool-5 feature vectors for each image, 5-sentence descriptions for training images, and textual tags for both training and testing images.

2 Data Description and Preprocessing

2.1 Data Description

The training and testing data is downloaded from kaggle website:

`https://www.kaggle.com/c/cs5785-fall-2017-final/data`

The data has four components in training and testing data, they are images, tags, feature vectors and descriptions. So here we will talk about how we pre-processes each part of data.

2.2 Bag of Word(BoW) Representation

2.2.1 Word Dictionary

First, we design a function to generate the word dictionary that is used in the calculation of BoW. By leveraging *Natural Language Toolkit(NLTK)*, we can analyze each description sentence in the training set and find out what role each word plays in the sentence. Intuitively, noun, adjective and verb are the most informative parts in a sentence, which are the greatest help for one trying to connect pictures with a sentence. However, preposition like 'in', 'with', etc. and article like 'the' and 'a' reveal little information about what picture this sentence tries to describe. This is why we decide to only extract noun, adjective and verb from description sentences and make them potential candidates of the word dictionary.

In addition, when we skim through all those candidates, we find some words are really uncommon and rarely used, which leads us to apply an occurrence time threshold t so that only words that appear more that t times can be a valid candidate. (we use $t = 15$ and $t = 20$ during the whole process)

Finally, we generated multiple versions of word dictionaries for training different models so that we may assemble them to generate robuster and more powerful models. We generate three versions of word dictionaries that consist of noun, noun&adjective, noun&adjective&verb respectively.

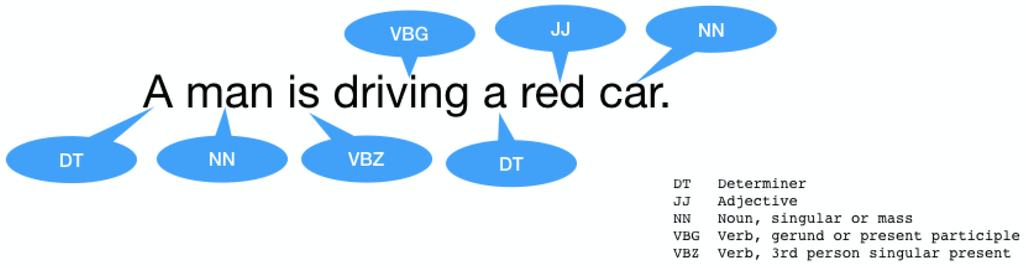

VBG
JJ
NN
A man is driving a red car.
DT
NN
VBZ
DT
DT Determiner
JJ Adjective
NN Noun, singular or mass
VBG Verb, gerund or present participle
VBZ Verb, 3rd person singular present

Figure 1: Sentence Analysis Example

2.2.2 BoW Vector

For each image's descriptions in the training and testing set, we can generate BoW vectors based on different versions of dictionaries. We call them **BoW train** and **BoW test**.

3 Model Architectures

3.1 Naive Nearest Neighbor

At first, we come up with a naive and fast lazy learning approach to solve this image retrieval problem, to get a benchmark on the lower bound of score for this problem. This approach is fast to implement because it only takes tags and description of the test set and no training is involved.

3.1.1 Model Explanation

We take the categories of the tags set and form that into a dictionary D, which has size 80. Then we take the description of the test set and map it to a binary vector with length 80 based on D: $X = BOW_D(Description_{test})$. Then we form another array of binary vector with length 80 by the original tags of test: $Y = BOW_D(Tags_{test})$.
Then we have two matrices X, Y of the same dimension 2000 by 80, where X is corresponds to test descriptions and Y corresponds to test images. Since the task is to find wanted image given description, the task now is to find most relevant Y given X. As they are in the same dimension, we find the nearest 20 Y_j's for each X_i, sorting by cosine distance.

3.1.2 Analysis

After we transform description into binary vector X, the statistics show that out of 2000 test samples, we only have 1500 unique X_i's and more precisely, a lot of descriptions are mapped into $[0]_{80}$ vector. This means the diction with 80 categories is too small to have big hits from description.

3.1.3 Performance

Kaggle submission score for this method is 0.11724, which is expected because the method is intrinsically forfeiting the scores for more than 500 of 2000 test cases when they are mapped to same vector.
It proofs the feasibility of bag of words model and it gives us the direction to find a diction that works better to map descriptions in the problem setting.

3.2 BOW+KNN

While examining the test tags data, we found that some of the tags were phrases instead of words. In the original version, the phrased were stored directly in the dictionary. Since all the descriptions were separated word by word, the phrases wouldn't be matched at all. Therefore, we split the phrases and stored every single word of tags in the dictionary to achieve a better match. Detailed comparison of accuracy will be discussed in the following chapter Experiments and Results. Another aspect of the revision mainly focused on the Bag-of-words representation. During the re- search of previous method, we realized that a weighted BoW representation was better than a binary.

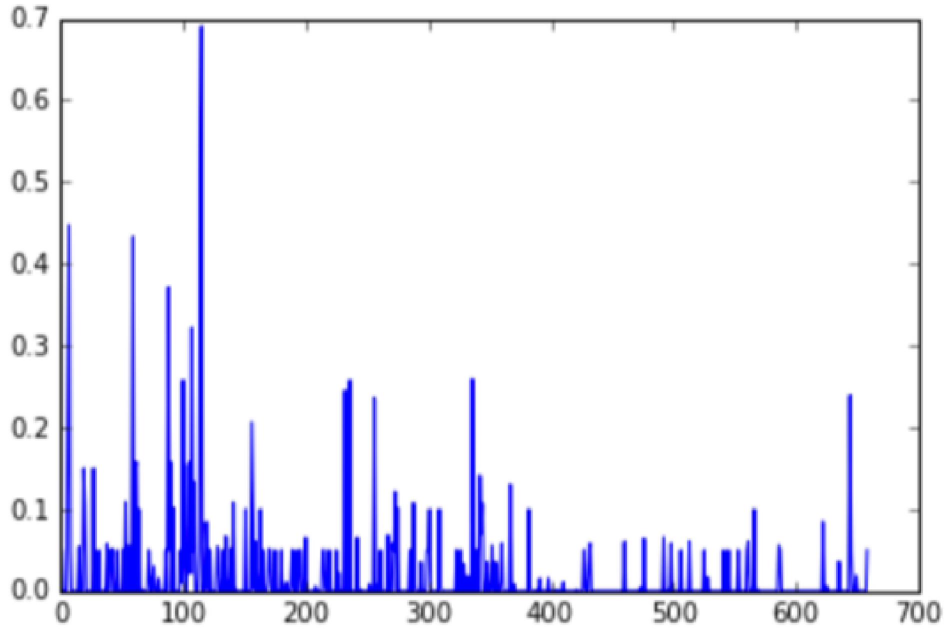


Figure 2: Procedure of Match Tags Method

One since words that appeared several times should have more impact on the match of descriptions with tags. We also realized that if a tag appeared in every image, than this tag would be useless in finding the matches. To be more specific, if every image contains an object in common, the matched image would not be found using the tag of that object. To avoid the impact of such common objects, we used a new way of calculating the bag-of-words representation. The BoW representation for each word was the proportion of the times the word appeared in the description or tag file it belonged to the times it appeared in the whole description or tag set.After we trained our BoW representation and predicted the representation of test description. The result is displayed in Figure 3.

Besides the tags, we map the image in test data with description in training data by using the feature - fc1000. We first use KNN to find 10 nearest features in the training data of each image in testing data and then do arrogation on all the description of these image. Since we have already prepared the bag of words for all the images in training data, in step, we sum each dimension of the BOW and normalize them. What we get now is a description for each of the image in testing data. Then we do KNN again to find the 20 nearest neighbor of the description and list them in a file.

3.3 Neural Network

In this section, we try to build Neural Network(NN) models that map BoW representation space onto image feature space.

We use noun&adjective dictionary to generate BoW representation for both training and testing descriptions, which would be used as model input during the training.

And then we build two NN models that regress 2048-dimensional *resnet1000intermediate* and 1024-dimensional *resnet1000*.

Two NN models have similar architecture:

- **Layer** One input layer (1000 neurons), One output layer(1000 neurons), Two hidden layers(2000 neurons each)
- **Activation** ReLu
- **Layer Type** Fully Connected
- **Dropout** Each Layer(except output layer) is followed by a Dropout layer with the 0.1 dropout rate.

Figure 3 demonstrates the architecture visually.

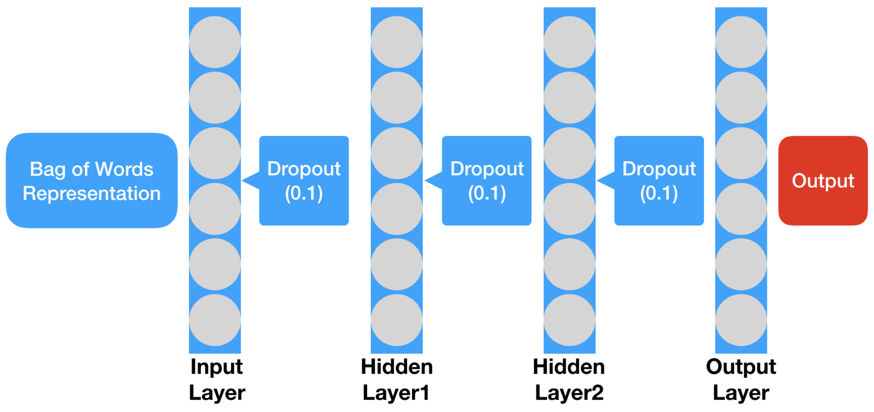


Figure 3: Neural Network Architecture

3.4 Partial Least Square Regression

After trying the first type of model, we start to think that by interpreting the data of image features, we could be able to calculate the semantic representation of the images. So it is worth trying to build a model that connect the image features with its BoW representation.

After some literature review, we find that Partial Least Square Regression(PLSR) could be a good way of building a model between one source of data and another.

We build PLSR models to map image feature space to BoW representation space so that for each image, we can know what description it fits into most. So for each description in the test set, we compare its BoW with the predicted BoWs of the images in the test set and fetch the most similar 20 images.

In the end, we trained 3 different PLSR models, each of which tries to extract different information from the image features so that they could be useful to build a robust ensemble model.

$$1000D\ Feature \rightarrow PLSR(n\_component = 400) \rightarrow BoW_{noun\_adj\_verb\_dict}$$
$$2048D\ Feature \rightarrow PLSR(n\_component = 400) \rightarrow BoW_{noun\_adj\_dict}$$
$$2048D\ Feature \rightarrow PLSR(n\_component = 500) \rightarrow BoW_{noun\_adj\_verb\_dict}$$

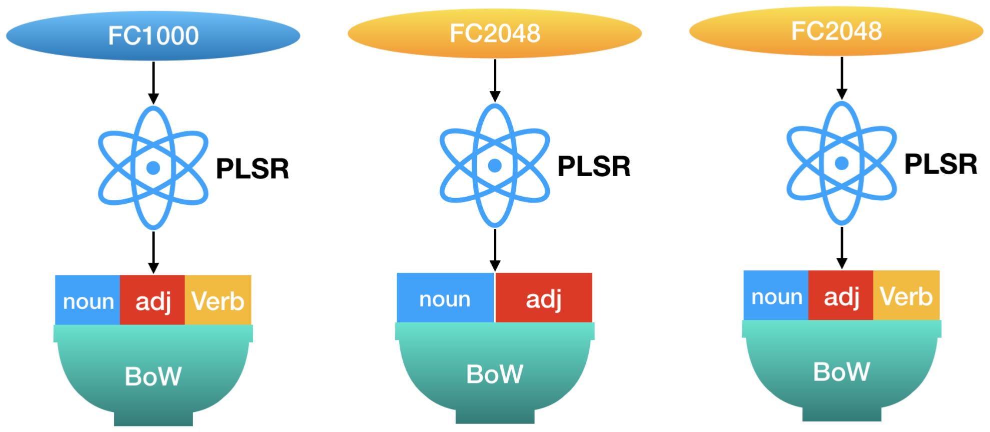


Figure 4: Partial Least Square Regression Models Overview

3.5 Similarity Evaluation

3.5.1 Neural Network Model

NN models map BoW representation to image features. In the testing phase, we map the BoW vector of each set of descriptions to a 1000-dimensional feature vector and a 2048-dimensional feature vector. Then calculate the cosine distance between these two vectors and the feature vectors of the images in the testing set, which give us a quantitative measurement of how well the images match the descriptions. We can pick the 20 images with the smallest cosine distance as the image retrieval result.

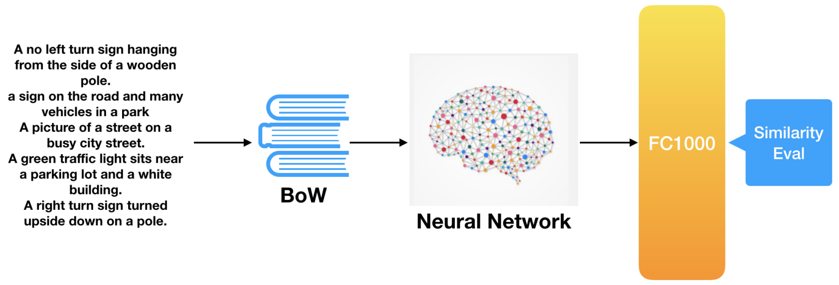

A no left turn sign hanging from the side of a wooden pole.
a sign on the road and many vehicles in a park
A picture of a street on a busy city street.
A green traffic light sits near a parking lot and a white building.
A right turn sign turned upside down on a pole.
BoW
Neural Network
FC1000
Similarity Eval

Figure 5: Neural Network Evaluation

3.5.2 PLSR Model

PLSR models map image feature vectors to BoW representation. In the testing phase, we map the image feature vectors of each image in the testing set to BoW vectors.

And for each set of descriptions, we first calculation its BoW vector and then compare it with the BoW vectors of the images in testing set based on cosine distance. Then we pick the 20 images with the smallest cosine distance as the result.

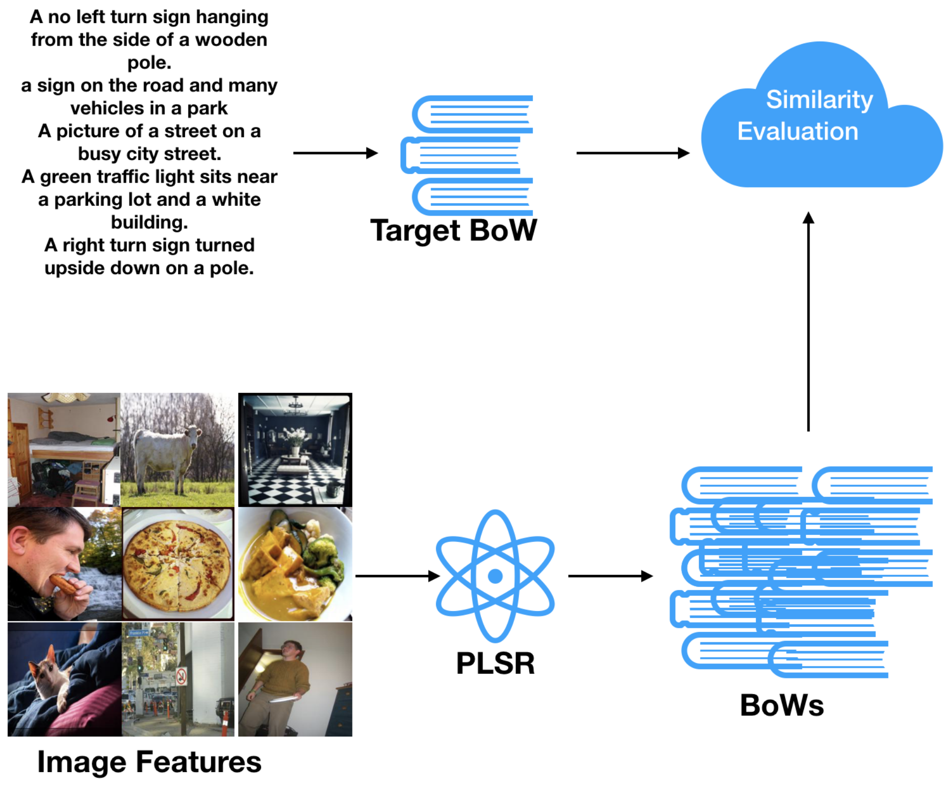
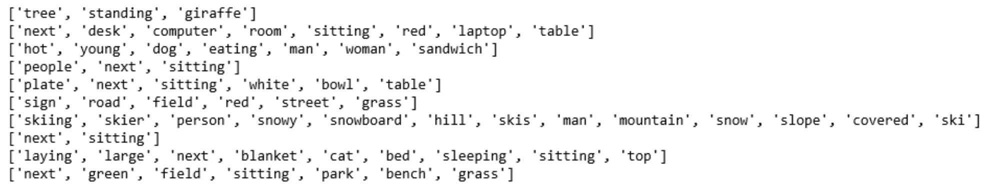

A no left turn sign hanging from the side of a wooden pole.
a sign on the road and many vehicles in a park
A picture of a street on a busy city street.
A green traffic light sits near a parking lot and a white building.
A right turn sign turned upside down on a pole.
Target BoW
Similarity Evaluation
Image Features
PLSR
BoWs

Figure 6: PLSR Evaluation

4 Result and Analysis

4.1 Validation

Here is a proof of concept our model works. After we get $X_{predict}$ from our model, which is a vector takes in value from 0 to 1 and the higher values are in an entry indicate the more confident we are that this description should be mapped to the key word of this entry. We take the threshold to be 0.3 and print out the entries we are predicting for test image 10 to 20.



```
['tree', 'standing', 'giraffe']
['next', 'desk', 'computer', 'room', 'sitting', 'red', 'laptop', 'table']
['hot', 'young', 'dog', 'eating', 'man', 'woman', 'sandwich']
['people', 'next', 'sitting']
['plate', 'next', 'sitting', 'white', 'bowl', 'table']
['sign', 'road', 'field', 'red', 'street', 'grass']
['skiing', 'skier', 'person', 'snowy', 'snowboard', 'hill', 'skis', 'man', 'mountain', 'snow', 'slope', 'covered', 'ski']
['next', 'sitting']
['laying', 'large', 'next', 'blanket', 'cat', 'bed', 'sleeping', 'sitting', 'top']
['next', 'green', 'field', 'sitting', 'park', 'bench', 'grass']
```

Figure 7: Prediction of key words in image 10-20

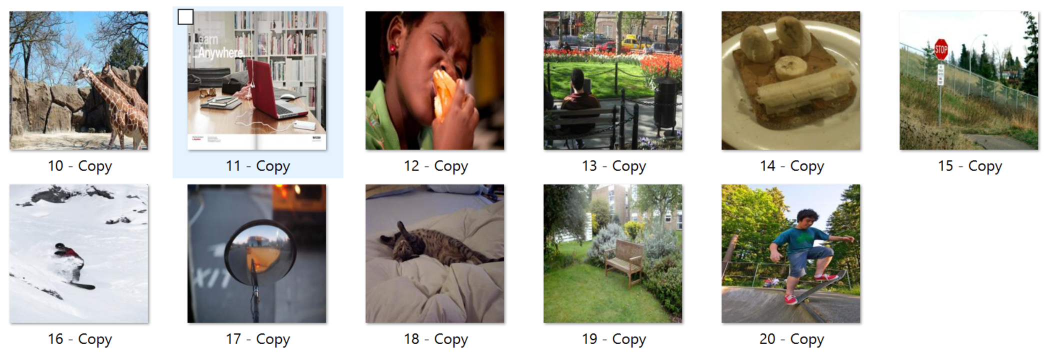


Figure 8: Actual test image 10-20

Comparing Figure 6 and Figure 7, we want to qualitative measure performance of our model's prediction from feature vector to image. For image 10, we successfully predict 'giraffe' and for image 15 we successfully predict 'sign'. This serves as a proof of concept that it works.
However, for image 12, it predicts all the key words 'hot', 'dog', 'eating', 'man', but it will be problematic to show those image for searches for dog. We see two intrinsic drawbacks here:

- 1-gram model does not include keywords involving multiple words, e.g. 'hot dog'
- Relation of object cannot be shown in the bag of words model representation, e.g. we cannot distinguish if 'man' is 'eating' 'sandwich' or 'dog' is 'eating' 'sandwich'

4.2 Image Retrieval Demo

Below we provide two demos of image retrieval: the descriptions are shown on the left hand side, and the retrieved top 5 images are on the right hand side.

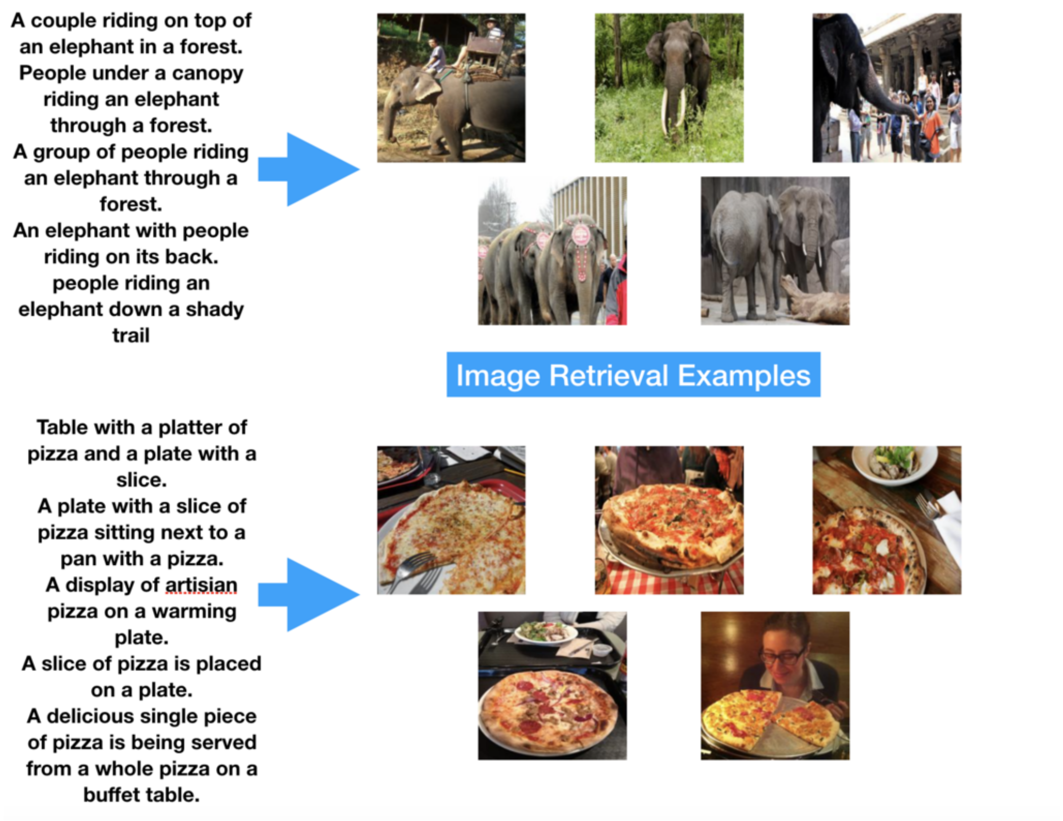

A couple riding on top of an elephant in a forest.
People under a canopy riding an elephant through a forest.
A group of people riding an elephant through a forest.
An elephant with people riding on its back.
people riding an elephant down a shady trail
Image Retrieval Examples
Table with a platter of pizza and a plate with a slice.
A plate with a slice of pizza sitting next to a pan with a pizza.
A display of artisian pizza on a warming plate.
A slice of pizza is placed on a plate.
A delicious single piece of pizza is being served from a whole pizza on a buffet table.

Figure 9: Two demos of image retrieval

4.3 Kaggle Scores

Here we report a series of our Kaggle submission scores and the best score we have is 0.36950.

Model	Performance
Naive Nearest Neighbor	0.11724
Neural Network	0.23293
BOW+KNN	0.28024
PLSR400	0.29730
3 PLSR	0.33426
2 NN + 3 PLSR	0.36950

4.4 Reflection and Future Work

To improve the drawback of bag of words model, we are planning to use doc2vec model which is based on word2vec to pertain relations of objects when mapped from description to vector. For the classifier, we are planning to implement ada-boost to get a better ensembled model to improve our score.
From this competition, we have learnt that in real application of data science, there could be multiple ways to define a problem and formulate a model. We experiment with multiple model of what is feature vector and what is label and the result differ a lot. We also have learnt data preprocessing takes a lot of time and effort and it is important to have clear constructed to code to help oneself and to communicate with others.

5 Acknowledgments & Reference

1. Natural Lanuguage Toolkit. http://www.nltk.org/;
2. sklearn
3. Tensorflow
4. Keras
5. Qibin Zhao, Liqing Zhang and Andrzej Cichocki. *Multilinear and nonlinear generalizations of partial least squares: an overview of recent advances*. WIREs Data Mining Knowl Discov; 2014, 4:104–115. doi: 10.1002/widm.1120
6. Herve Abdi. *Partial least squares (PLS) regression*. In Encyclopedia of Social Sciences Research. Sage Publications.