Text and Image Features, Vector models and Similarity/Distance measures

Akshay Shah
Kshitij Kashettiwar
Nikhil Agarwal
Ronak Tanna
Shivam Dhar
Shreyas Dewan
Shubham Verma

Abstract

In this project, we are working on the text and image features obtained by the textual and visual descriptors like tf,df,tf-idf and CM,CN,CSD,HOG,LBP to name a few respectively.
The main idea of the project is to reason how similar one entity is to the other depending on the features given. The similarity measures form an important part of the implementation in such a way that we are working on vectors and vector spaces. Primarily, we are working on how concept of similarity measures and vector models can be applied to datasets with pre computed features to be able to make sense out of them.

Keywords
TF,DF,IDF, vector models, Euclidean distance, Cosine similarity, K Means

Body

I. Introduction

a. Terminology

i. TF (Term Frequency)

1. Definition: No of occurrences of the term in the document.
2. Project context:
   No of users querying a specific term(concatenation of titles, tags, description of images)

ii. DF (Document Frequency)

1. Definition: No of occurrences of the term in the set of documents.
2. Project Context:
   No of images that mention the given queried term.

iii. TF – IDF

1. Definition: TF/DF

iv. Vector models

1. Definition: Representation of objects in the vector space in quantifiable way such that any metric can be applied on it.
2. Project Context:
   We are using the entities locations, images and users in terms of vector models in a way that distance measure can be applied to find out the similarity between concerned entity objects.

b. Goal Description

In this project, we implement 5 tasks categorized on the basis of similarity between the entities namely users, locations and images for textual descriptors and finding locations based on the visual features computed on the basis of 10 color models as described in the Bogdan et al.(2014).
The motivation and goal behind each task is described as follows
Task1
In this task, we implemented to fetch k most similar users to the given user id and model from set of TF,DF,TF-IDF which are computed in terms of query by user in terms of image title , description or tag. The motivation behind this task is to find out how are users related in the real world in terms of what location have they clicked pictures on or how have they tagged images, etc.
We also fetch the similarity score for each match and which terms are most similar.

Task2
In this task, we implemented to fetch k most similar images against given image id and model from set of TF,DF,TF-IDF which are computed on terms of image metadata like name, location, coordinates, time,etc.
We also fetch the similarity score for each match and which terms are most similar.The motivation behind this task is to find out how are images related in terms of where they are located or how are they described.
We also fetch the similarity score for each match and which terms are most similar.

Task3
In this task, we implemented to fetch k most similar locations against given location and model.We also fetch the similarity score for each match and which terms are most similar. The motivation behind this task is to find out how are locations related in terms of metadata or user description.

Task4
In this task, we implemented to fetch the k most similar locations given the color model based on the features computed by visual descriptor colour models. We also return the image pairs for the location match and corresponding similarity score. The motivation behind this task is to work with computation of vector models of precomputed features and also find out locations which are similar based on the images which are located there.

Task5
In this task, we implemented to fetch the k most similar locations based on the features computed by visual descriptor colour models. We also return the image pairs for the location match and corresponding similarity score. The motivation behind this task is to work with computation of all colour models for a given location and find out locations which are similar based on contribution of each colour model.

c. Assumptions

The key assumptions which play an important role while performing this task are basically the computed weights for tf,df and tf-idf for each terms is consistent and that term relevance is counted from the view of entire dataset. The dataset provided has less redundancy and has less noise, considering that one of the key approaches to task requires data to be consistent which is computing the centroid using KMeans clustering algorithm

II. Proposed Solution

Task1

In the task1, we get the list of users for each user object in the textual descriptor file dataset_textTermsPerUser.txt. From the list of obtained users, we extract the vector of terms and weights for each user and then construct a common vector between the given user and each other users in such a way that we only consider the intersection of terms in their term-weight vectors. This is specifically done in order to not get into **“Dimensionality Reduction Curse”** where there will be multiple redundant features and ever increasing dimensions thus leading to complications when learning from the data. After that we compute the similarity using **cosine similarity** metric because cosine works very well in case when magnitude doesn’t matter and primarily when dealing with text data when there are documents of uneven lengths. After computing the similarity between the term vectors of users, we fetch top k users who are similar based on the scores and for calculating the individual term contribution, we **normalize** the scores because we can get a good idea of which term contributed to the final score.

Task2

In the task2, we get the list of images for each image object in the textual descriptor file dataset_textTermsPerImage.txt. From the list of obtained images, we extract the vector of terms and weights for each image and then construct a common vector between the given image and each other image in such a way that we only consider the intersection of terms in their term-weight vectors. After that we compute the similarity using **cosine similarity** metric as explained above. After computing the similarity between the term vectors of users, we fetch top k images who are similar based on the scores and for calculating the individual term contribution, we **normalize** the scores because we can get a good idea of which term contributed to the final score.

Task3

In this task, we follow the same approach as task1 and task2.

Task4

In this task, we are dealing with visual descriptor in colour models like CM,CN,CN3x3,etc for images per respective locations. So we need to compute the location vectors for given input model for given location and the other 29 locations
As per the given data, each model has different sets of dimensions and some models have very high dimensions. Thus per given model, we compute the centroid vector per location which gives an good idea of how close the individual image objects are. Thus we use **K Means clustering** algorithm here which helps to compute the centroid given specified number of clusters. Thereafter we use **Euclidean distance** to compute the similarity score between the given location and other locations because Euclidean works really well for image data and most importantly, we needed to preserve the distance.

For the second part where we need to find the similar image pairs from given location to k similar locations, for this we use the brute force approach to compute the similarity between image vectors for given location and other respective k similar locations and distance measure again we use Euclidean.

Task5
In this task, we extend the idea of task4 for all the models and compute the similarity vector for each colour model for each location. Thereafter we calculate the norm of the similarity vector for each location to obtain the final similarity score for a given location pair. For contribution of each model, we need to find the norm of the similarity vector to get a standardized value.

III. Interface specification

The inputs for the task are provided in the form of command line argument where taskid corresponding to each task is followed by the specific input for that given task
The following is the explanation of input and resultant outputs for all the tasks

Task1
Input
```
1 39052554@N00 TF 5
```
Here we have task_id followed by user_id, model and k value

For this input, output which we get is as follows

Output
```
[('10117222@N04', 0.27831216351296784, ['2011', 'bok',
'garden']), ('10261566@N02', 1.0, ['2011', 'bok', 'garden']),
('10288162@N07', 1.0, ['2011', 'bok', 'garden']),
('10297518@N00', 1.0, ['2011', 'bok', 'garden']),
('10298101@N05', 1.0, ['2011', 'bok', 'garden'])]
```

Output can be described as follows
First value is user_id
Second_value is the similarity score between the given user id and returned user id
Third value is list of terms which contribute maximum to the similarity

Task2

Input

```
2  4459178306 TF 10
Task_id , image_id , model ,k (In order)
```

Output

```
[(9067739127, 1.0, ['16thcentury', 'agra', 'akbar']), (9069963392, 1.0, ['16thcentury', 'agra', 'akbar']), (9067738157, 1.0, ['16thcentury', 'agra', 'akbar']), (9067738777, 1.0, ['16thcentury', 'agra', 'akbar']), (5176517974, 1.0, ['16thcentury', 'agra', 'akbar']), (5176518500, 1.0, ['16thcentury', 'agra', 'akbar']), (5176522936, 1.0, ['16thcentury', 'agra', 'akbar']), (5176521224, 1.0, ['16thcentury', 'agra', 'akbar']), (5175918109, 1.0, ['16thcentury', 'agra', 'akbar']), (5175916261, 1.0, ['16thcentury', 'agra', 'akbar'])]
```

Output can be interpreted as list of images with each image consisting of image_id, similarity metric for terms and list of the terms.

Task4

Input

```
10 CN3x3 7
```

Output

```
[('cologne_cathedral', 208498401.95561314, [(7225656268.0, 2976189.0), (7225656268.0, 6044975521.0), (7225653660.0, 2976189.0)]), ('chichen_itza', 872570757.1442966, [(4746366668.0, 2976189.0), (2392835042.0, 2976199.0), (4746366668.0, 6044975521.0)]), ('aztec_ruins', 653447192.6909628, [(3238907655.0, 2976199.0), (3238907655.0, 6045504436.0), (4854719231.0, 2976189.0)]), ('christ_the_redeemer_rio', 476476474.9765854, [(6911141373.0, 8556157009.0), (8220413134.0, 2433677360.0), (4106630562.0, 2433677360.0)]), ('angkor_wat', 662251414.5613661, [(5061776617.0, 2976220.0), (11933493993.0, 2976199.0), (5062387464.0, 2976220.0)]), ('berlin_cathedral', 217526576.76254177, [(4016826847.0, 3292117677.0), (4016826847.0, 6045504436.0), (2707220528.0, 6045504436.0)]), ('casa_batllo', 1224603791.8435502, [(4264025324.0, 2976189.0), (4264025324.0, 6044975521.0), (480419960.0, 2976220.0)])]
```

Output can be interpreted as list of locations with each location consisting of image_id, similarity score and list of 3 image pairs contributing most to the similarity

IV. System Requirements/installation and execution instructions

It's recommended to setup a virtualenv with python 3.6

Steps to setup for execution

a) Create virtualenv with python 3.6

b) Activate virtualenv

c) Run pip install -r requirements.txt

Running instructions

Since we have 5 tasks to implement, we use task_id in the comammnd line argument as starting followed by task specific input which is explained as follows and task specific parameters will be followed by the task id

Commandline arguments (task specific)

a)Task1

task_id,user_id,model,k

Example – python main.py 1 39052554@N00 TF 5

b)Task2

task_id,image_id,model,k

Example - python main.py 2 9067738157 TF 5

c)Task3

task_id,location_id,model,k

```
    Example - python main.py  3 27 TF 5
d)Task4
    task_id,location_id,model,k
    Example - python main.py  4 10 CN3x3 7
e)Task5
    task_id,location_id,k
    Example - python main.py  5 4 5
```

V. Related Work

Computing the similarity between large sets of images is a active research area as it has many challenges. Owing to deep learning techniques like CNN is used to compute the features seamlessly, thus one would be able to compute the similarity between the images. One such reference paper would be Jian Wang et al.(2014) where it is explained that owing to challenges for learning fine grained similarity between images, the author introduces deep learning techniques to distinguish between images and the author states that they achieved far better results than on hand crafted visual features and deep classification models.

VI. Conclusion

In this project, we used different similarity measures owing to the type of the data at hand. The future scope for the project is to try and experiment different similarity measures other than cosine and Euclidean and try to make sense of the data while computing the similarity between the entities. For image to image similarity, especially the challenge of matching the images ranges from change in orientation, hue in the image, multiple objects in the images apart from main object leads us to using deep learning models for computing the features and using deep ranking we can compute the image to image similarity efficiently as stated in the Jian Wang et al.(2014)

Bibliography

B. Ionescu, A. Popescu, M. Lupu, A.L. Gînscă, H. Müller, “Retrieving Diverse Social Images at MediaEval 2014: Challenge, Dataset and Evaluation”, MediaEval Benchmarking Initiative for Multimedia Evaluation, vol. 1263, CEUR-WS.org, ISSN: 1613-0073, October 16-17, Barcelona, Spain, 2014.

Jiang Wang , Yang Song , Thomas Leung , Chuck Rosenberg , Jingbin Wang , James Philbin , Bo Chen , Ying Wu, Learning Fine-Grained Image Similarity with Deep Ranking, Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, p.1386-1393, June 23-28, 2014 [doi>10.1109/CVPR.2014.180]