

Price Match Guarantee

Determine if two products are the same by their images

Group Member:

Heyang HUANG
Zheming HUANG
Jian ZHANG
Zhenhong ZHANG
Hongji ZHOU

1. Motivation

Shopee is a cross-border e-commerce company. It initiated this competition because this type of technology can help e-commerce companies reduce the workload of checking repeated listings of products and detecting product category labels.

Two different images of similar wares may represent the same product or two completely different items. To avoid misrepresentation and other issues that could come from conflating two dissimilar products. Both customers and retailers will benefit from accurate goods list for better shopping experience and more effective sale.

To contribute more accurate product categorization, we need to combine the several algorithm model including OpenCV, KNN and EfficientNet. From this task, we have studied the latest image algorithm and their code, which broadens our horizon and extends skills of data handling.

2. Background

Compared with Option 1, our task originates from actual demand of Shopee, which combine with most popular computer vision, Neural Network technic and python coding. What's more important, process of dealing task is similar with deal with demand from Product Manager, give much contribution to make abundant resumes.

In this project, we build a Neural Embedding structure, use EfficientNet to practise our model and solve both regression and classification predictive problems by kNN. To better understanding models that we use, here are the pre-research, which including concept of Embedding, ConvNets and kNN:

a) Embedding:

Embedding (or vector space) methods find a lower-dimensional continuous space in which to represent high-dimensional complex data. The pattern of an embedding is a mapping of a discrete—categorical—variable to a vector of continuous numbers. In the context of neural networks, embeddings are low-dimensional, learned continuous vector representations of discrete variables.^[1]

Neural embedding models are a kind of embedding scheme where the vector space corresponds to a subset of the network weights, which are learned through backpropagation. Neural embedding models have been shown to improve performance in a large number of downstream tasks across multiple domains.

Neural network embeddings have 3 primary purposes:

1. Finding nearest neighbors in the embedding space. These can be used to make recommendations based on user interests or cluster categories.
2. As input to a machine learning model for a supervised task.
3. For visualization of concepts and relations between categories.

With using neural network embeddings, we can represent a great amount of data (over 30,000 in our project) using only less than 100 numbers in a vector.

b) Convolutional Neural Networks (ConvNets):

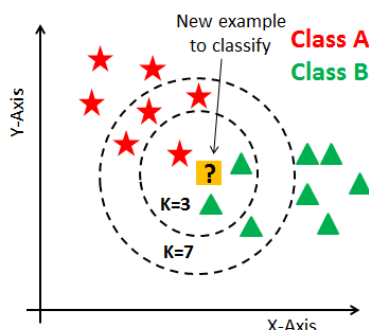
Here is come to a problem, how can we do embedding for the input images? CNN, Convolutional Neural Network, is a widely used and basic deep learning model in computer image domain. It's a basic target to handle the input images with huge dimensions of info as we have talked about. For more realism concern, there are many more effective model which is based on CNN, such as ResNets, MobileNets.

Compared with normal Neural Networks, ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. Therefore, ConNets is build for efficient image prediction and recognition.

Simple Example Architecture are as follow:

- **INPUT layer:** $[32 \times 32 \times 3]$ will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV layer:** will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as $[32 \times 32 \times 12]$ if we decided to use 12 filters.
- **RELU layer:** will apply an elementwise activation function, such as the $\max(0, x)$ thresholding at zero. This leaves the size of the volume unchanged ($[32 \times 32 \times 12]$).
- **POOL layer:** will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as $[16 \times 16 \times 12]$.
- **FC(fully-connected) layer:** will compute the class scores, resulting in volume of size $[1 \times 1 \times 10]$, where each of the 10 numbers correspond to a class score. Each neuron in this layer will be connected to all the numbers in the previous volume.^[2]

c) kNN:



kNN is a Supervised Learning algorithm. A supervised machine learning algorithm is one that relies on labelled input data to learn a function that produces an appropriate output when given unlabeled data. In supervised learning, you train your data on a labelled set of data and ask it to predict the label for an unlabeled point.^[4]

The k-nearest neighbor algorithm stores all the available data and classifies a new data point based on the similarity measure (e.g., distance functions). This means when new data appears. Then it can be easily classified into a well-suited category by using kNN algorithm.

3. Description

To find the neighbors of target products, we need to convert image and text into the forms that are useful for regression and classification.

Our data processing mainly includes two parts: Image Recognition and Text Recognition.

a) OpenCV:

OpenCV is an open-source library that includes several hundreds of computer vision algorithms. We use this package to convert provided pictures into RGB form to construct the Input layer. And it is able for following layer to decolorize the images.

b) EfficientNet:

There are several architectures in the field of Convolutional Networks that have a name, such as LeNet, AlexNet, ZF Net, GoogLeNet and so on. Here is the reason why we choose EfficientNet.

In our image processing task, balance of computation and performance is a very important part. So we may tend to choose a more swift and flexible network. And the adjustment to hyper parameters and structures are very confused and are a process full of trial and error. So we tend to choose a model that already consider the best combination of hyper parameters and structures such as width, resolution size and so on. Last but not least, training a network from zero to one is very difficult so we tend to choose a model can be pretrained and we can do fine-tuning on it to improve the performance. And finally we find out that the EfficientNet is the best choice.

As we've mentioned, the balance of network depth, width and resolution is difficult to be found out. Some researchers figure out that some combination of them, also called as model scaling method, can lead to a higher performance of both Resnets and MobileNets in Image Data set. And they also come up with a new formula to represent the whole parameters: depth, width and resolution

It means that subjective to some conditions of parameters as well as the memory and computation limit, what the highest accuracy we can get. In order to well adjust these tiny modules of CNN-based network more conveniently, the researchers construct a new framework based on the swift network, MnasNets. For example the basic version EfficientNet is shown as below.

And after validating the scaling method among ResNets and MobileNets and coming up with a new network, the EfficientNet as a baseline model of image processing model is born. And the performance of it is very good and the flexibility is very attractive to quick implementation developer. Here is the short comparison among different CNN-based networks.

That is the major reasons that we choose EfficientNets to do our image embedding job. With the help of some libs such as pytorch, we can directly get the model structure, and we can download the pretrained model file to start our fine-tuning without building the entire model from zero to one. But explain again, we will not use the FFN part of it, can just make use of the embedding convolution results.

c) kNN:

Building nearest neighbor graphs is often a necessary step when dealing with problems. There are two methods to get nearest neighbors: ϵ -graph and kNN graph. ϵ -graph is geometrically motivated, and many efficient algorithms have been proposed for computing it, but ϵ -graph easily results in disconnected components and difficult to find a good value of ϵ which yields graphs with an appropriate number of edges. Compared with ϵ -graph, kNN graphs have been shown to be especially useful in practice for advantages like quick calculate time, simple algorithm to interpret, versatile to be used for regression and classification, high accuracy and so on.

In general, the KNN classification algorithm includes the following four steps:

- 1) First, prepare and preprocess data.
- 2) Then, calculate the distance between the test sample point (that is, the point to be classified) and each other sample point.
- 3) Third, sort each distance, and then select the K points with the smallest distance.
- 4) Finally, according to the principle that the minority is subordinate to the majority, the test sample points are classified into the category with the highest proportion among K points.

So it is important how to select the K value.

d) Perceptual Hash Functions:

Perceptual hashing is the name for a class of algorithms that attempts to produce a fingerprint of image data (or other multimedia). Unlike a checksum or cryptographic hash, though, a perceptual hash is a fuzzy hash – the perceptual hash of two similar images should be similar.^[3]

In fact, transform images into hash is the process of trying to extract and summarize meaningful “features” from the image.

As we mentioned before, we divide images into block and decolorize the images, after such conversion, suitable block mean hash value based algorithms is important to improve hitting rate.^[5]

d) TF-IDF features

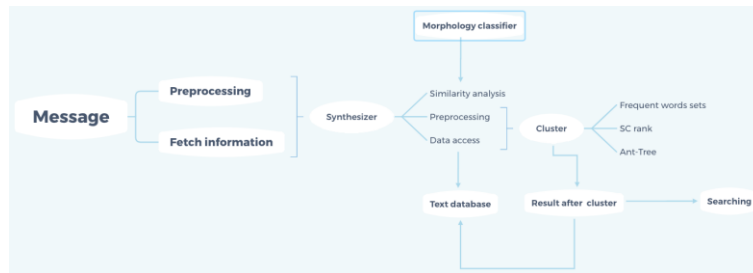
TF-IDF (term frequency-inverse document frequency) is always used in information retrieval. TF means the term frequency, and IDF means the inverse document frequency. It is a kind of statistical method to assess the importance of a word to one of the documents in a set of documents or a corpus.

The main idea of this method is that if the frequency of a word or phrase in one article is high, and it rarely appears in other articles, it is considered that this word or phrase has a good ability to distinguish categories and is suitable for classification. TFIDF is actually TF * IDF.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,k}}, \quad idf_i = \lg \frac{|D|}{|(j|t_i \in d_j)|}$$

$|D|$ is the number of all the documents.

The main process of TF-IDF is as follow.



The cosine similarity is also a important part in using TF-IDF algorithm. First find the key words of each sentence with the TF-IDF algorithm. Then take several words and put these words in a set, and compute the frequency of the words in this set from each sentence. Generate word frequency vector of each sentence. Calculate the cosine similarity of each two vectors. The greater the value, the more similar it is.

The advantage of TF-IDF algorithm is simple and fast, and the result is more in line with the actual situation. However, this algorithm cannot reflect the position information of words. In order to avoid this problem as far as possible, we should give greater weight to the first paragraph of the full text and the first sentence of each paragraph.

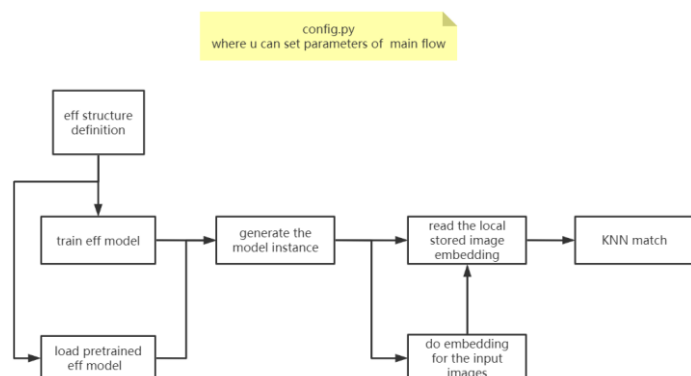
4. Implementation

a) Brief of program flow

The project code controls various parameters in the main flow of the program through the config.py file.

When we train the model for the first time, we will load the shopee data set and the effnet_b4 pre-training model for image embedding, and get an image embedding result, and then use this result to perform KNN operations to perform image match. This process costs a lot of time. To save time when the code is reproduced, we store the embedding model for the future.

Next time we execute the program, we can change the parameter `COMPUTE_CV=False` to make the program use the past embedding results to get the same fitting results. The basic structure diagram of the project code is as follows:

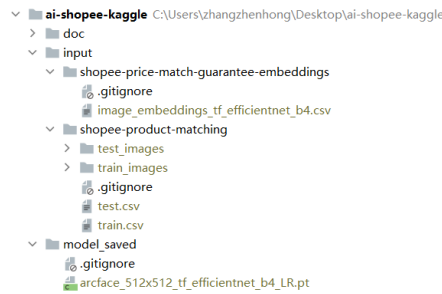


b) Code execution process and logic

1、Set the random seed of torch

When using PyTorch, by setting the random number seed, the training results of each time are fixed on the GPU or CPU so that the training results can be reproduced.

2、Data loading



(1) The training data and verification data are from the official website of the Kaggle competition, which can be obtained on the official website:

<https://www.kaggle.com/c/shopee-product-matching>

(2) image_embeddings_tf_efficientnet_b4.csv

The embedding training results are from the training results obtained by us for the first training.

(3) arcface_512x512_tf_efficientnet_b4_LR

The Effnet_b4 pre-trained model is loaded through the timm library of Python.

Embedding training results and Effnet_b4 pre-training model, we archived in Google Cloud Disk. Before the project starts, please download the relevant data set, embedding training results, and effnet_b4 pre-training model and add them to the project-specific folder.

<https://drive.google.com/drive/folders/1Bo5QrEdWOvTFKklN5d99hYdwCWZXISG?usp=sharing>

3、The main function of the program

get_image_embeddings(): Image embedding function

get_text_embeddings(): Text embedding function

get_image_neighbors(): Find the closest image collection function (look for duplicate images)

get_text_predictions(): Find the closest text aggregation function (look for descriptions of repeated images)

4、Embedding.

4.1、image part:

The purpose of this code is to get an embedding result, which is image_embeddings. It is stored as image_embeddings_tf_efficientnet_b4.csv. The code can be divided into the following 2 processes:

(1) Training embedding: Use shopee's image data set to train embedding.

(2) Use trained embedding.

4.2、text part

Use shopee's image description text set to train embedding. And get an embedding result, which is the text_embeddings of the text.

5、Model prediction

5.1、image part

Use the embedding result to perform the KNN operation. With the help of KNN method, calculate the nearest top k image sets of a certain image.

5.2、text part

Use cosine similarity to match text.

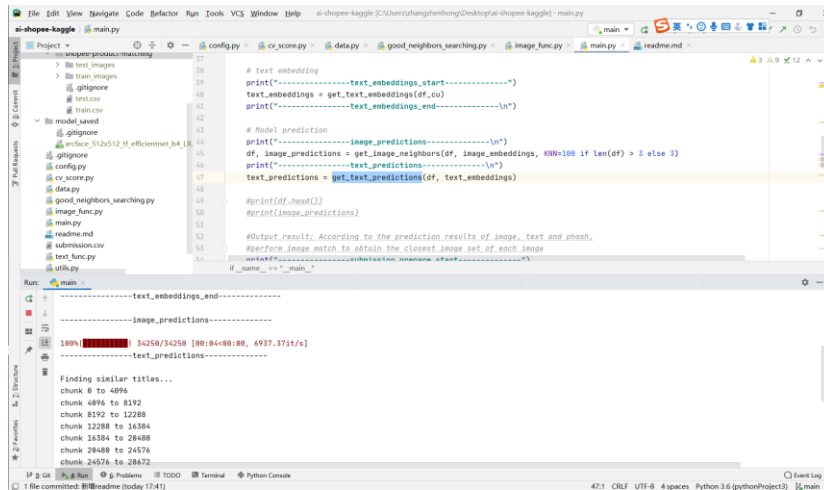
e) **Program execution environment, running screenshots and logs**

IDE: PyCharm 2020.2.3 x64

Python version: 3.6.7

Processor: CPU/GPU

OS: Windows(CPU)/Linux Docker(GPU)



Console execution log printing, please check the appendix file **console_Print.txt**

5. Data

a) Data Exploratory Analysis

The training set metadata. Each row contains the data for a single posting. Multiple postings might have the exact same image ID, but with different titles or vice versa.

b) Field description

posting_id - the ID code for the posting.

image - the image id/md5sum.

image_phash - a perceptual hash of the image.

title - the product description for the posting.

label_group - ID code for all postings that map to the same product. Not provided for the test set.

c) Preview of the first ten rows of data

Data overall situation:

Total number of data: 34250

null value: no

Types of label_group: 11014

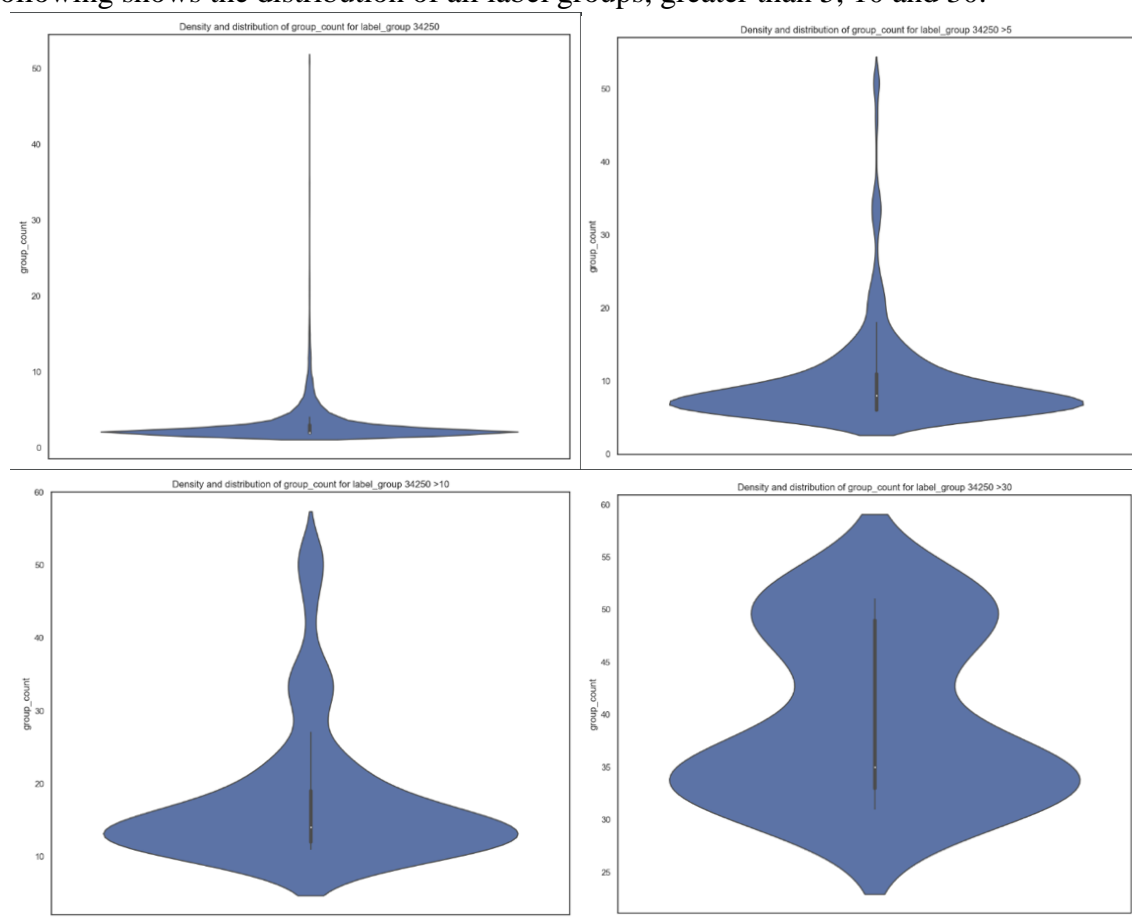
	A	B	C	D	E
1	posting_id	image	image_phash	title	label_group
2	train_129225211	0000a68812bc7e98c42888dfb1c07c	94974f937d4c2433	Paper Bag Victoria Secret	249114794
3	train_3386243561	00039780dfc94d01db8676fe789ecc	af3f9460c2838f0f	Double Tape 3M VHB 12 mm x 4	2937985045
4	train_2288590299	000a190fdd715a2a36faed16e2c65c	b94cb00ed3e50f78	Maling TTS Canned Pork Lunch	2395904891
5	train_2406599165	00117e4fc239b1b641f08340b4296	8514fc58eafea283	Daster Batik Lengan pendek - N	4093212188
6	train_3369186413	00136d1cf4e4ede0203f32f05f6605f	a6f319f924ad708c	Nescafe vc3v89dair Latte 220m	3648931069
7	train_2464356923	0013e7355ffc5f8fb1ccad3e42d92f	bbd097a7870f4a50	CELANA WANITA (BB 45-84 K)	2660605217
8	train_1802986387	00144a49c56599d45354a1c28104c	815c9bb833ab4c8	Jubah anak size 1-12 thn	1835033137
9	train_1806152124	0014f61389cbaa687a58e38a97b63f	eea7e1c0c04da33d	KULOT PLISKET SALUR /CANDY	1565741687
10	train_86570404	0019a3c6755a194cb2e2c12bfc6397	ea9af4f483249972	[LOGU] Tempelan kulkas magn	2359912463
11	train_831680791	001be52b2beec40ddcd1d2d77c7a68	e1ce953d1a70618f	BIG SALE SEPATU PANTOFEL KI	2630990665

d) The top ten most type of group:

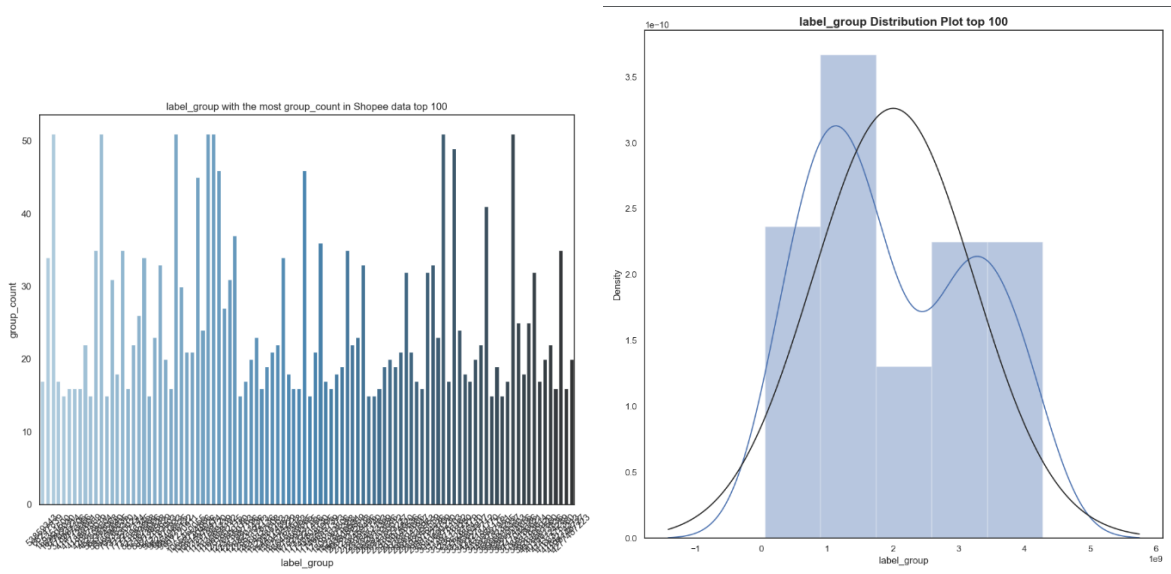
	label_group	group_count
0	994676122	51
1	159351600	51
2	562358068	51
3	3113678103	51
4	1163569239	51
5	1141798720	51
6	3627744656	51
7	3206118280	49
8	1166650192	46
9	1733221456	46

e) Data distribution

The x-axis represents label_group, and the y represents the number of a group. The following shows the distribution of all label groups, greater than 5, 10 and 30.



f) The distribution of the top 100 label groups:



6. Results and Observation

a) Output result

The code will generate a submission.csv file, the image id and the image collection closest to it. These collections represent duplicate images.

To output the result, we will perform the following three steps to get the correspondence between posting_id and matches. Where posting_id represents the original image id, matches represents the matched image id, that is, the image id that is repeated with the original image. Utilize trained embedding.

	A	B
1	posting_id	matches
2	train_129225211	train_129225211 train_2278313361 train_1816968361 train_2120597446 train_3386243561 train_3423213080 train_3805508898
3	train_3386243561	
4	train_2288590299	train_2288590299 train_1744956981 train_2406599165 train_3526771004 train_3576714541
5	train_2406599165	
6	train_3369186413	train_3369186413 train_921438619

- (1) Use phash to get the perceptual hash of the image.
- (2) Use image_predictions, text_predictions, and phash_predictions for matching.
- (3) Output image id and repeated image id set.

b) Cross validation

The score of the model in the original data set is: CV score = 0.7739374593131568.

```

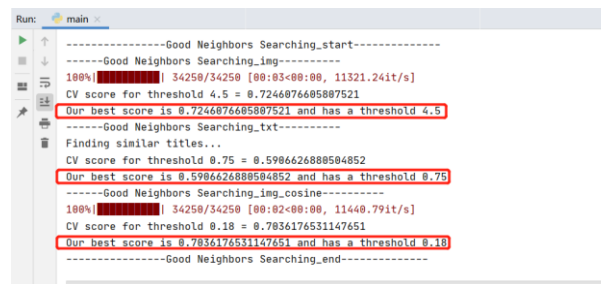
-----submission_prepare_start-----
-----submission_prepare_end,please find the submission.csv-----
-----CV_Score_start-----
CV score = 0.7739374593131568
-----CV_Score_end-----

```

c) Optimization

To find the best neighbours, that is, the highest accuracy, we want to know which threshold allows the program to obtain the highest cross_validation score. For this purpose, the threshold_searching() function is executed, and the threshold value is explored for the three situations of image, text and image cosine, and a threshold with the best cv score is

obtained.



```
Run: main
-----Good Neighbors Searching_start-----
-----Good Neighbors Searching_img-----
100%|██████████| 34258/34258 [00:03<00:00, 11321.24it/s]
CV score for threshold 4.5 = 0.7246876685887521
Our best score is 0.7246876685887521 and has a threshold 4.5
-----Good Neighbors Searching_txt-----
Finding similar titles...
CV score for threshold 0.75 = 0.5986626888504852
Our best score is 0.5986626888504852 and has a threshold 0.75
-----Good Neighbors Searching_img_cosine-----
100%|██████████| 34258/34258 [00:02<00:00, 11440.79it/s]
CV score for threshold 0.18 = 0.7836176531147651
Our best score is 0.7836176531147651 and has a threshold 0.18
-----Good Neighbors Searching_end-----
```

d) Conclusion

Through the design of the local model, we have obtained a classifier with excellent performance, and we can get an image classifier with a CV score = 0.7739374593131568. For the input data, we can get similar images through the three dimensions of image, text, and phash, to determine duplicate images.

At the beginning of the experiment, we artificially set a smaller threshold, but the CV score obtained by the program was lower. By designing the nearest neighbour algorithm, we can reverse the check to get a more threshold, improve the CV score and get a better result.

7. Discussions

In this project, we build a model that can predict which items are the same product. So as to help consumers choose a lower price in the online shopping mall.

The novelty of this project is that it does not solve the problem in a conventional direction. It includes image recognition, text recognition and matching of image or text. The difficulty is not a simple additive relationship, but involves multi-dimensional considerations. Therefore, we have also carried out a lot of thinking in terms of opening up new fields, and finally built a workable model.

After a lot of training and optimization, this model has high accuracy. But at runtime, you need to ensure high-performance equipment and spend some time.

In this project, our model can be verified to be effective. It can help many consumers to solve the needs of choosing products and obtaining discounts in online shopping malls, and at the same time, it will have a positive impact on the platform and promote the enthusiasm of customers to use the platform. In this way, the mall only spends a small amount of money to apply this model, which enables users to have a good shopping experience and thereby create greater profits.

What's more, in the process of user use, the mall can collect data and conduct training to further improve the accuracy of the model and form positive feedback. It will also help the development of academia and enable industry and academia to complement each other.

In addition, the idea of combining two algorithms in different directions is also a breakthrough. In the future of science and technology, it is possible to try to integrate more kinds of algorithms to combine to break the upper limit of a single algorithm. For this idea, there is still a lot of room for people to think about.

8. Reference:

- [1] Benjamin Paul Chamberlain, James R Clough, Marc Peter Deisenroth. Neural Embeddings of Graphs in Hyperbolic Space
- [2] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks;
- [3] Christoph Zauner. Implementation and Benchmarking of Perceptual Image Hash Functions. 2010: 7-8;
- [4] Jie Chen, Haw-ren Fang, Yousef Saad. Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection. Journal of Machine Learning Research 10 (2009), 1989-2012;
- [5] Bian Yang, Fan Gu, Xiamu Niu. Block Mean Value Based Image Perceptual Hashing, International Conference on Intelligent Information Hiding and Multimedia Signal Processing. 2006;