CS376 Project Milestone

Product Matching on E-Commerce Platforms

Hoang Trung Dung	-	20180853
Le Viet Thanh Long	-	20190780
Nguyen Tuan Kiet	-	20200734
Cao Viet Hai Nam	-	20200817

I. Introduction

1. Motivation

With the rapid development of many e-commerce platforms, especially in the Covid-19 pandemic, people go shopping online a lot. Retailers usually want to drive more customers to buy their products by offering the cheapest prices. One of the methods for achieving that goal is product matching, which collects the prices of the same product sold by many other shops, and gives a retailer information to offer a competitive rate to customers. Our team would like to conduct an application project for the product matching problem. Specifically, we want to build a machine learning model predicting which items are the same products from their images, which can improve the price collecting process significantly faster.

2. Dataset

Our dataset for this project includes about 30,000 product images and a CSV metadata file from a Kaggle competition mentioned in the reference part. Each row in the CSV file contains the following data for a single image posting:

- *posting_id*: the ID code for the posting.
- *image*: the image id in the images folder.
- *mage_phash*: a perceptual hash of the image.
- *label_group*: ID code for all postings that map to the same products.

For example, one row in the CSV file is:

posting_id	image	image_phash	label_group
train_129225211	0000a68812bc7e98c42888dfb1c07da0.jpg	94974f937d4c2433	249114794

II. Methodology

1. Siamese neural network and Triplet loss

a) Siamese neural network

Inspired by the famous face recognition model FaceNet, we are going to train a Siamese neural network to learn the feature embeddings of the input image. The architecture of these networks is built upon an image classification model removing the last classification layer.

b) Triplet loss

Triplet loss is applied to enforce the learning process in Siamese neural networks. This objective function requires three inputs: a reference input (anchor), a matching input (positive), and a non-matching input (negative). The goal of triplet loss is to minimize the distance from anchor to positive while maximizing the distance from anchor to negative. The triplet loss of anchor input A, positive input P and negative input N is described as follows:

$$\mathcal{L}(A, P, N) = \max\Big(\|\,\mathrm{f}(A) - \mathrm{f}(P)\|^2 - \|\,\mathrm{f}(A) - \mathrm{f}(N)\|^2 + \alpha, 0\Big)$$

where *f* is the Siamese neural network; α is a margin threshold between positive and negative pairs.

c) Triplets categorization

Based on the distance between anchor to positive and negative samples, we can classify triplets into three categories:

- Easy triplets: the triplet loss is 0, which means the positive is much closer to the anchor than the negative.
- Semi-hard triplets: the positive is closer to anchor than the negative, but still within the margin.
- Hard triplets: the negative is closer to anchor than the positive.

d) Triplet mining

To produce triplets as input for each batch, there are two different approaches:

- Offline triplet mining: Produce triplets offline, for example, at the beginning of each epoch.
- Online triplet mining: Compute useful triplets on the fly on each batch of inputs.

The latter approach is more efficient, and thus we decided to use it in our project. And for this approach, there are two strategies to calculate triplet loss on each batch:

- Batch all: select all valid triplets and calculate the average triplet loss of the hard and semi-hard triplets.
- Batch hard: for each image, only select the positive farthest away and the negative nearest to the anchor.

The batch hard strategy yields the best performance and is thus being used in our project.

2. Image Matching

The feature embeddings generated from a trained Siamese neural network will be normalized and then fit into a K-Nearest Neighbor (KNN) model to find the *K-Nearest Neighbors* for each embedding. The neighbors which have distances to the embedding less than *distance_threshold* - a tunable hyperparameter - would be considered a match and then the labels of each such match will be compared..

3. Title Matching

From the title field available in the dataset, we can also use the NLP model to extract feature embeddings from text and the rest of the process is similar to image matching.

Feature embeddings can be

- A simple TF-IDF vectorizer (each title is like a bag of words), or
- Contextual embedding (for example, BERT has the ability to 'extract' meaning out of words by breaking it into components)

4. Evaluating result

For evaluating our model, we will use the F1 Score to be the metric. The mean is calculated in a sample-wise fashion, meaning that an F1 score is calculated for every predicted row, then averaged.

III. Experiments

1. Exploratory Data Analysis (EDA)

We first explored the tabular data of annotation files to find out NaN values or abnormal rows. Fortunately, the data is rather clean and there are no NaN values to care about. We checked column-wise unique values and observed that only the posting_id column has unique values and the rest have duplicate values. There are 34,250 rows in the training annotation files, which consist of 32,412 images grouped into 11014 different label groups. Thus, there are duplicates in images, hashes, and titles, not just label groups.

The title of each product is mostly a short string describing the name or characteristic of the product written mostly in Malaysian, Indonesian, German, and English. In addition, preprocessing the text is necessary to exclude many special characters (eg: \, #. @ …) which are irrelevant to the context of the title.

We also observe some noises in the dataset. In the annotation files, we found several cases of assigning different label groups to the same images.

2. Title Matching

Because Shoppee is widely used in South-East Asia, the product titles contain many kinds of language. For that reason, we use the multilingual text model with the ability to handle all the target languages. The multilingual model is trained on the concatenation of Wikipedia in 104 different languages with the pre-trained task being BERT's mask language model. Three text encoders are selected from Hugging face to do experiments:

- BERT base multilingual model (bert-base-multilingual-cased)
- DistilBERT base multilingual model (distilbert-base-multilingual-cased)
- Paraphrase multilingual sentence transformer (paraphrase-multilingual-MiniLM-L12-v2)

For the general structure of the procedure, we first initialize the text model and tokenizer scheme. After preprocessing text, we forward it to the text model and utilize a dense vector from the last hidden layer to extract text embeddings. It is worth mentioning that mean pooling is applied to text embeddings afterward taking attention masks into account for correct averaging. Finally, for the evaluation, we compare the distance between embeddings using cosine distance or using the KNN model. If the distance of 2 embeddings is within a reasonable threshold, we match the corresponding titles to the same group

We also use the TF-IDF (Term Frequency-Inverse Document Frequency) method to do title matching. Thus each sentence in the title field would have its own vector.

The result of the experiments we obtained is shown below:

Model	Best F1-score val
BERT base multilingual model	57.74%
Paraphrase multilingual sentence transformer (PMST)	58.74%
DistilBERT base multilingual model	59.72%
TF-IDF	62.25%
Combine TF-IDF + PMST + image_phash	68.27%

Overall, using embedding extracted from the products' titles can yield reasonable results on product matching problems, especially when combining several methods together. The BERT-based model seems to produce worse F1 scores compared to the TFIDF method. The disadvantage of extracting embeddings of the last layer is that it can be heavily biased towards the actual task of BERT, which is MLM (Masked Language Modeling) or NSP (Next Sentence Prediction). Since even the very early layers are involved in extracting syntactic and grammatical senses out of the sentence, therefore, it may make more sense to take the average of the embeddings across all layers rather than plucking the embeddings of the last layer or last few layers.

3. Image Matching With Pretrained Model

Before training the Siamese neural network, we first apply pre-trained image classification models to create a baseline for our later experiments.

To conduct experiments, we used the pre-trained image classification models from *timm* library, and NearestNeighbors model from *sklearn* library. 32450 data rows were split into a training set and a validation set, with a ratio of 8:2.

After some tuning on the training set, we obtain *distance_threshold* = 0.6 to be the optimal value. We tested on 4 image models and got the following results:

Model	Accuracy on ImageNet	F1-score on Shopee dataset
ResNet-18	69.76%	60.47%
ResNet-101	77.37%	67.64%
EfficientNet-B0	77.1%	67.74%
ViT-B/16	77.91%	66.67%

From the result above, we can see that pre-trained models on ImageNet show a good result when transferred to product matching problems.

4. Image Matching With Fine-tuned Model

We trained the Siamese neural network with the triplet loss objective function. Each model is trained for 50 epochs with an early stopping mechanism, with appropriate tuned hyperparameters and optimizer. The same image augmentations are used in each experiment, which consists of Random Resized Crop, Random Color Jitter, Random Grayscale, Random Horizontal Flip and Random Rotation.

Model	Optimizer	Learning Rate	F1-score	Improvement
ResNet-18	SGD	0.001	71.63%	11.16%
ResNet-101	Adam	0.0001	70.84%	3.20%
EfficientNet-B0	Adam	0.0001	70.53%	2.79%
ViT-B/16	Adam	0.0001	72.46%	5.79%

The results from the above table indicate that all the chosen Siamese neural networks yield improved results after fine-tuning, up to 11.16% for ResNet-18. In addition, Vision Transformer model ViT-B/16 outperforms others on the feature extracting task, yielding the best results for both the ImageNet and the Shopee dataset. Another noticeable point is that ResNet-101 and EfficientNet-B0 yield lower scores than ResNet-18 when transferred to feature learning on the Shopee dataset. This might indicate several overfitting on these two deep CNN models because ResNet-18 does not perform as well as ResNet-101, EfficientNet-B0 on the ImageNet dataset.

IV. Conclusion

In conclusion, the Siamese neural network achieves excellent results even using only pre-trained weight on ImageNet, and the results can be improved further when fine-tuning further on the Shopee dataset. In general, feature embeddings extracted from images yield better results than feature embeddings extracted from product titles. We also propose several ways to achieve better results:

- Using different loss functions: Arcface, Barlow Twins
- Combining feature embeddings from text and image approaches.
- Applying different model structures

V. Contribution:

- Cao Viet Hai Nam: Prepare Exploratory Data Analysis for the dataset. Responsible for the title matching task. Perform experiments with different NLP models for the extraction of text embeddings.
- Nguyen Tuan Kiet: Participate in preparing the project's document. Implement an inference framework to evaluate the pre-trained model and training results. Do experiments to train ResNet-101 and EfficientNet-B0.
- Hoang Trung Dung: Building PyTorch training pipeline, performing experiments on fine-tuned ResNet-18 and ResNet-101, participating in making slides and project presentations.
- Le Viet Thanh Long: participate in preparing the project's proposal, slides making, project presentation, project final report. Do experiment with ViT-B/16.

VI. References

1. Shopee. *Shopee - Price Match Guarantee competition*. Kaggle.

2. Schroff, F.; Kalenichenko, D.; and Philbin, J. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
3. Alexander Hermans, Lucas Beyer, and Bastian Leibe. *In defense of the triplet loss for person re-identification*. 2017 CoRR.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
5. Tan, M. and Le, Q. V. *EfficientNet: Rethinking model scaling for convolutional neural networks*. 2019 ICML.
6. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. *An image is worth 16x16 words: Transformers for image recognition at scale*. 2021 International Conference on Learning Representations (ICLR).