CIS522 Project: Shopee Product Match

Waleed Algadhi
School of Engineering and Applied Science
University of Pennsylvania
walgadhi@seas.upenn.edu

Yifei Li
School of Engineering and Applied Science
University of Pennsylvania
liyifei@seas.upenn.edu

Abstract

With the rise of e-commerce platforms, product matching becomes more crucial. It can help the customers find the best deal, help the companies to offer competitive prices, and help the society defeat the counterfeits. Here, we empirically studied the Shopee product dataset with several natural language processing and computer vision approaches, reaching an outstanding accuracy of 97%.

Notebooks link: *GitHub Repository*

1. Motivation and Social Impact

In the recent years, with the rise of Internet, the amount of both customers and products on the E-commerce platforms are increasing. A recent study estimates that the worldwide B2C e-commerce has reached more than $2.3 trillion [3].

On an e-commerce platform, it's important to help savvy shoppers find the best deals without paying extra money for the same product. Hence, similar-product detection based on both images and texts can empower the platform to offer products at rates that are competitive to the same product sold by another retailer. It's harder than it might seem at the first glance, since the differences between similar products may be subtle while photos of identical products may be wildly different! Therefore, we need to resort to deep learning techniques with concentration on computer vision and natural language processing to be able to detect such differences.

Also, there's a flood of counterfeit products online, putting public trust in those businesses at risk. However, by deploying machine learning techniques that can detect similar products, figuring out the potential fake products can be made easier, causing reduction in IP violations and rebuilding mutual trust. Moreover, instead of merchants competing on more advertisements, another competition will arise on the product prices since checking similar products will be made easy for the user which will allow him to pick the product with the best price.

2. Literature Review

Oliver *et al*. [14] introduced several heuristic methods and among them random forest-based methods was the best for product resolution (also known as product matching).

Specially, as for semantic analysis, it has been found that for short text matching, the specifically-designed algorithms, like MV-LSTM and DRMMTKS, work the best, while the state-of-the-art BERT-based model has a mediocre performance [11].

There is a vast amount of research about image matching. Convolutional neural networks (CNNs) are the foundation here. With only a simple modification of the deep CNN, the binary code method can outperform some state-of-the-art methods on image retrieval [10]. The deep neural netowrks like SimNet and multi-scale CNN with image embedding have been deployed successfully on image similarity learning [2]. Besides, the entropy-based algorithms like entrypo-histogram approach play an important role here[1]. Last, model-based behavioral cloning also has a potential in detecting the image similarity [16].

To match the text, Neculoiu *et al*. concludes the usage of Siamese recurrent networks in learnign text similarity [9].

3. Dataset

3.1. Introduction

In this project we use a dataset[1] provided by Shopee. It contains 32,412 product images associated with short title descriptions for each product as training data and roughly 70,000 images in the hidden testing set. The total number of unique classes is around 10,000.

The title dataset is named as train/test.csv, which is 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. The first few lines of csv files are showed in Table 1 and their five attributes are explained as below.

[1] Source: https://www.kaggle.com/c/shopee-product-matching/data

- 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.

The image dataset is the train/test images associated with the postings. A sample of them are given in Figure 1.

3.2. Summary Statistics and Visualization

The wordcloud is showed in Figure 2. Some of them are in English while others are in the Latin letter form of Southeastern languages. Most of the product titles are short, with less than 20 words (Figure 3).

The imagesize in Figure 4 shows that most images take a box shape with equal length and width and most of the images are of a shape that is about 500x500 pixels.

Figure 2: The Wordcloud of Product Titles

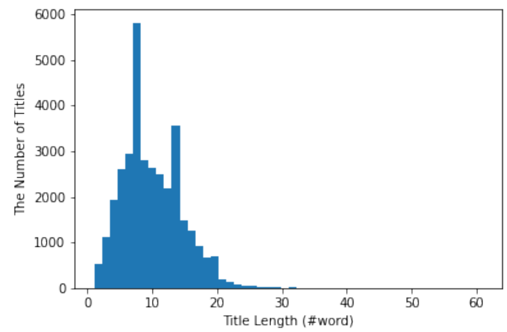


Figure 3: The Histogram of Product Title Length

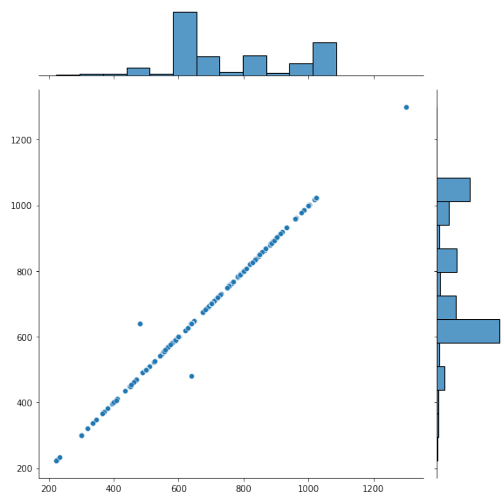


Figure 4: The Distribution of Product Image Size

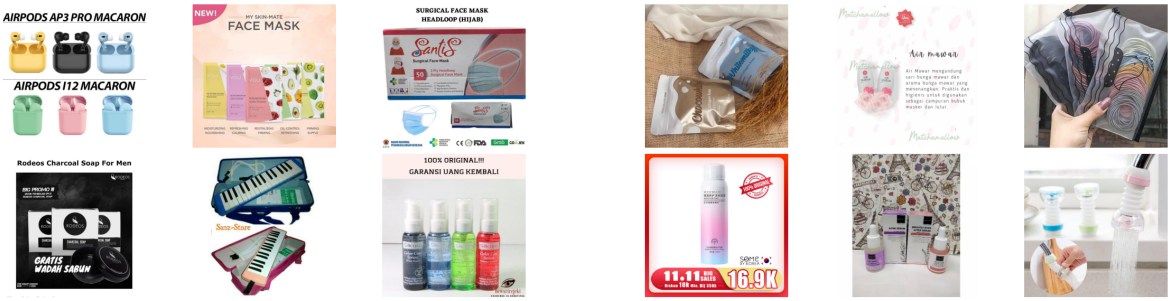
3.3. Pre-Processing and Data Augmentation

As for the product titles, we did the lower case conversion because the title meanings should be the same despite the letter case. We didn't strip symbols like slash or hashtag because all of them may contain useful information. Translation is unnecessary, as well. Although the corpus contains some southeastern Asian languages like Indonesian, the sentence embedding generated in the end would make no difference as long as the test corpus is in the same language style.

As for the product images, we transform them into a standardized format that conform to the ImageNet standards which the pretrainded models expect, which will also help to avoid any dimensionality mismatch. We also perform data augmentation which enables the neural network model to generalize better on prediction since it introduces slightly different versions of the same image to the training process every time the training loop pass by it. The preprocessing

	posting_id	image	image_phash	title	label_group
0	train_129225211	0000a68812bc7e98c42888dfb1c07da0.jpg	94974f937d4c2433	Paper Bag Victoria Secret	249114794
1	train_3386243561	00039780dfc94d01db8676fe789ecd05.jpg	af3f9460c2838f0f	Double Tape 3M VHB 12 mm x 4,5 m ORIGINAL / DO...	2937985045
2	train_2288590299	000a190fdd715a2a36faed16e2c65df7.jpg	b94cb00ed3e50f78	Maling TTS Canned Pork Luncheon Meat 397 gr	2395904891
3	train_2406599165	00117e4fc239b1b641ff08340b429633.jpg	8514fc58eafea283	Daster Batik Lengan pendek - Motif Acak / Camp...	4093212188
4	train_3369186413	00136d1cf4edede0203f32f05f660588.jpg	a6f319f924ad708c	Nescafe \xc3\x89clair Latte 220ml	3648931069

Table 1: Sample of CSV File



(a) Random Products

(b) Common Products

Figure 1: Sample of Product Images

steps include resizing the image to 256x256 pixels, center-croping it to 224x224 pixels and finally normalizing it according to the ImageNet standards. The augmentation steps include color jittering and random horizontal flip.

4. Methodology

4.1. Metrics

4.1.1 Loss Function

To learn embedding similarity, we use the triplet loss function, which achieved huge success on finding the human face similarity via FaceNet [12]. The loss formula is

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

where A is an anchor input, P is a positive input of the same class as A, N is a negative input of a different class from A, α is a margin between positive and negative pairs, and f is an embedding.

4.1.2 F1 Score

F1 score is a weighted trade-off between the precision and recall. We compare the F1 score using micro and macro parameter. The micro-metric is globally calculated by counting the total true positives, false negatives and false positives, while macro-metric is calculated on each label and then finds their unweighted. The latter doesn't take label imbalance into account and therefore the comparison can indicate the impact of label imbalance.

4.2. Zero-Shot Learning

The original Kaggle competition is formalized as a classic supervised learning problem. The images that are given in the testing set to make inference on and submit to Kaggle are all of labels that appear in the training set as well. In contrast, the learning scheme we decided to use here is zero-shot learning. We wanted to build a model that is powerful enough to distinguish and predict on classes that have never been introduced to the model during training. That is achieved by splitting the training and testing set based on the classes. This methodology is beneficial from a practical perspective since there are many products that come and go in the market and we want a model that is robust against that shift in distribution and does not cost the user the inconvenience of wasting time and resources on retraining. Our model works by extracting the features from the product image along with its text description and embedding them into a 300 dimensional vector. These vectors are fed into a k-nearest-neighbors algorithm that compares these embeddings to their closest match (1 neighbor due to under-representation of some classes) which will infer the query product's class. It is worth noting that with additional products coming into the system we do not need to retrain to the neural network model. We only need to extract the product features and add them to the KNN algorithm which

will allow new products to be compared against those already fitted.

4.3. Natural Language Processing

4.3.1 BM25 (Baseline)

BM25 is a TF-IDF-based algorithm for querying corpus and returning the ones most relevant to the query by similarity scoring. It achieves a good performance of providing the similarity rank of the sentence compared to its variants [15].

Package: rank_bm25.BM250kapi

4.3.2 Doc2Vec

Since BM25 cannot generate corpus embedding, we resort to another model called Doc2Vec. It is similar to Word2Vec but works on the phrase-, sentence, and document-level. By adding the document-unique feature vector called paragraph ID in the distributed bag-of-words method, given a context of sentences, it can predict how likely it is for each sentence in the documents being a sentence. Therefore, it can represent each sentence as a vector and thus it's much easier to feed the neural networks [8].

Package: gensim.models.doc2vec

4.3.3 BERT

The transformer-based pre-trained model called BERT is developed by Devlin, J. *et al*. in 2019 [5]. It's empirically powerful while conceptually simple thanks to its bidirectional training of transformer, a mechanism that allows BERT to learn the context of a word based on all of its surroundings. This model caused a stir in the NLP community and still maintains the state-of-the-art results.

Package: bert-pytorch

4.3.4 Faiss

Faiss is a similarity search model developed by Facebook AI Research in 2017 [7]. It assumes that the instances are represented as vectors. After a data structure of corpus is built, the model can output the Euclidean distance given a new vector to find the similarity.

Package: faiss

4.4. Computer Vision

4.4.1 K-Nearest Neighbors (ML Baseline)

We use the K-Nearest Neighbors algorithm as a baseline approach to simply map raw images their to their position in space and predict on queried images by finding the most similar one in that space according to the specified distance metric (Minkowski in our case).

Package: sklearn.neighbors.KNeighborsClassifier

4.4.2 DNN (DL Baseline)

We use a basic three layered neural network as a baseline to measure the performance of our other deep learning models against. The architecture we used is illustrated as Figure 5.

Package: torch.nn

```
FullNet(
  (linear_layers): Sequential(
    (0): Linear(in_features=150828, out_features=1024, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=1024, out_features=512, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=512, out_features=300, bias=True)
  )
)
```

Figure 5: DNN Architecture

4.4.3 CNN

Bell *et al*. [4] shows the power of Convolutional Neural Network on visual similarity. Here we use the convolutional layers followed by 2 linear embedding layers trained from scratch. The architecture we used is illustrated as Figure 6.

Package: torch.nn

```
ConvNet(
  (cnn_layers): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear_layers): Sequential(
    (0): Linear(in_features=201004, out_features=512, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=512, out_features=300, bias=True)
  )
)
```

Figure 6: CNN Architecture

4.4.4 VGG19

VGG is a CNN architecture proposed by Simonyan, K. *et al*. in 2015 [13]. It adds the depth of networks by a very small convolution filters and leads to a significant improvement on image recognition. Here we use 19 convolutional layers pre-trained on ImageNet followed by 3 linear layers pre-trained on ImageNet and fine-grained on our dataset.

Package: torchvision.models.vgg

4.4.5 ResNet152

ResNet is a breakthrough of image classification developed by He, K. *et al*. in 2015 [6]. By skipping connections and mapping identity, ResNet can avoid the vanishing gradients, degrading accuracy, or keep learning the residuals to match

the predicted with the actual. Hence, it can exploit the advantage of deeper neural network and obtain higher accuracy. Here we use 152 convolutional layers pre-trained on ImageNet followed by 3 linear layers trained on our dataset.

Package: torchvision.models.resnet

4.5. Prediction and embedding comparison

After generating an embedding for the queried product image and title, we can feed that embedding into a K-nearest Neighbors algorithms in order to compute the most similar products to its embedding based on the Minkowski distance which can be considered as a generalization of both the Euclidean distance and the Manhattan distance.

5. Experiments and Results

5.1. Model Implementation

Our approaches can be summarized by Figure 7. The NLP and CV embeddings are combined in one vector and fed into a KNN to find the best match for them.

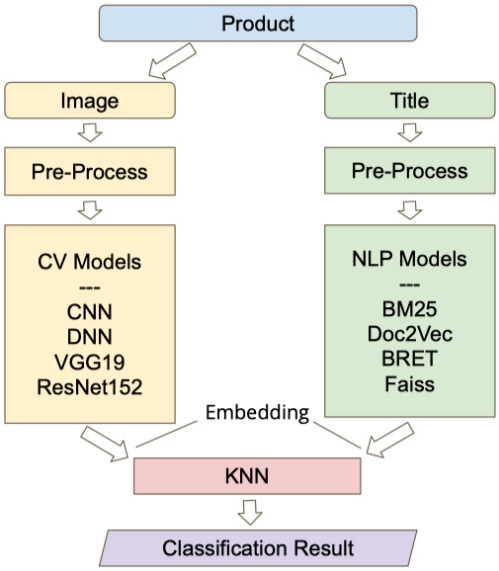


Figure 7: Overall Summary of Approaches

As for the natural language processing part, the accuracy comparison of different approaches is showed in Table 2. In this case, the Doc2Vec models work the best, while BERT has a low accuracy probably can be attributed to the fact that the text BERT is pre-trained on is quite different from the product titles we use here [11]. There's not enough time to fine-tune the Faiss model, so we pick Doc2Vec as our final model to classify the product title.

As for the computer vision part, unfortunately there wasn't enough time to train the VGG model, so we decided to compare the performance of DNN, CNN, and ResNet.

NLP Models	Accuracy
BM25	27.63%
Doc2Vec	31.29%
BERT	14.56%
Faiss	–

Table 2: NLP Models Accuracy Comparison

5.2. Evaluation

5.2.1 Loss

The validation loss follows its expected behaviour in the three neural network models (Figure 8). It starts high in the first epoch with a steep decrease in the first couple of epochs before it starts to converge. We can notice that the more complex models that are more expressive and able to capture the hypothesis better are starting with a lower loss than the more basic ones and keep maintaining that status over the following epochs.

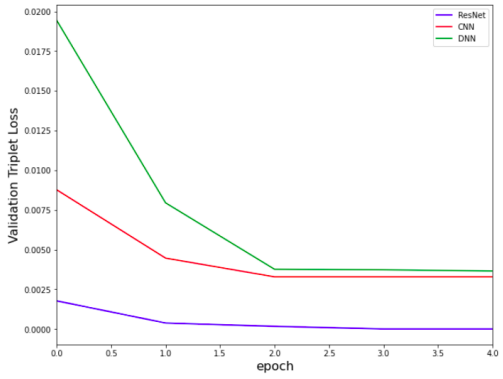


Figure 8: The Triplet Loss Curve of Computer Vision Models

5.2.2 Accuracy

The accuracy results show an advantage for the ResNet model which maintains a high accuracy from the first epoch (Figure 9). The DNN and CNN models score significantly less and do not seem to have the same robustness and consistency of the resnet model. Finally, the vanilla KNN model performed worse than all the neural network approaches with 0.1 accuracy score.

5.2.3 F1-micro

The F1-micro results show an advantage for the ResNet model which maintains a high score from the first epoch. The DNN and CNN models score significantly less and do

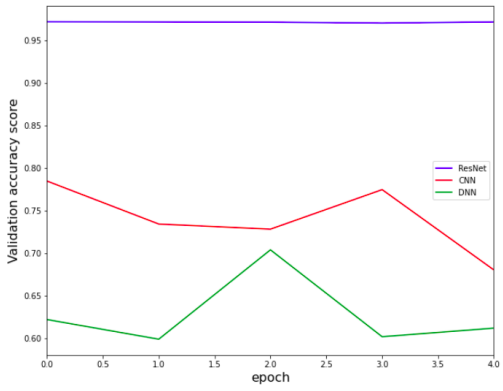


Figure 9: The Accuracy Score of Computer Vision Models

not seem to have the same robustness and consistency of the resnet model. Finally, the vanilla KNN model performed worse than all the neural network approaches with 0.1 F1-micro score.

5.2.4 F1-macro

The F1-macro results are a little bit different from the previous two metrics. However, it still shows an advantage for the ResNet model which changes slightly over the epochs but still maintains a higher score from the first epoch. The DNN and CNN models score significantly less and do not seem to have the same robustness and consistency of the resnet model. Finally, the vanilla KNN model performed worse than all the neural network approaches with 0.06 F1-macro score.

The F1-score curves are showed in Figure 10.

6. Conclusion and Discussion

6.1. Result and Analysis

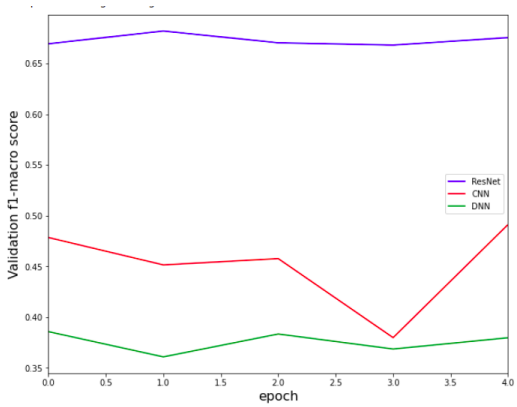
The results we achieved according the metrics mentioned above are generally high for a zero-shot learning model, yet the disparity between the high F1-micro score and the lower F1-macro score requires attention. This is can be explained by the fact that the F1-micro score, just like accuracy, does not take into consideration class imbalance in the results where, on the other hand, the F1-macro score does. This is an indication that the model has some bias to predict over-represented classes, which makes sense given the abundance of their embedding points in the KNN space which make them more likely to be neighbors of the queried product. Consequently, since they are more predicted, then their weight in the F1-micro score dominates the less represented classes which leads to that high score.

Some samples of matching products that were found by our model can be found in Figure 11.

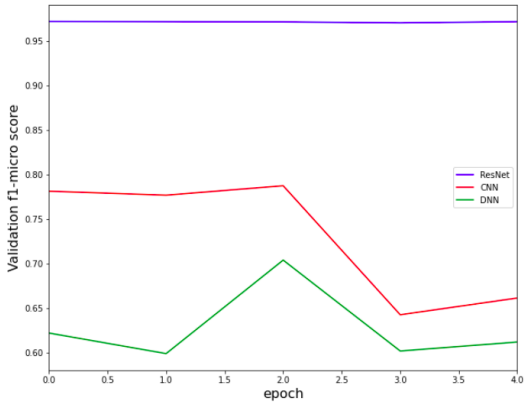
6.2. Limitations

The class label given by Shopee is not perfect. A few products that are nearly identical to humans' eyes are labeled in different classes for no reason (e.g. the 9729-th, 23462-th, and 23923-th instances in the training set). Also, a large part of test dataset is stored in the Kaggle system and cannot be accessed so it's hard to analyze all the misclassifications. To test and generalize the models better, it's recommended to try different dataset from different e-commerce platforms.

Running a single epoch on any of our models is computationally expensive (about 12 hours per single train per model). Therefore, to illustrate our results, we show evaluation metrics on the testing set on the first five epochs.



(a) F1-Score Macro



(b) F1-Score Micro

Figure 10: F1-Score of Computer Vision Models

(a) Book

(b) Phone Strap

Figure 11: Sample of Matching Product Images

References

[1] Mohammed Abdulameer Aljanabi, Zahir M Hussain, and Song Feng Lu. An entropy-histogram approach for image similarity and face recognition. *Mathematical Problems in Engineering*, 2018, 2018.

[2] Srikar Appalaraju and Vineet Chaoji. Image similarity using deep CNN and curriculum learning. *CoRR*, abs/1709.08761, 2017.

[3] Vitalina Babenko, Zdzisław Kulczyk, Irina Perevosova, Olga Syniavska, and Oksana Davydova. Factors of the development of international e-commerce under the conditions of globalization. In *SHS Web of Conferences*, volume 65, page 04016. EDP Sciences, 2019.

[4] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph.*, 34(4), July 2015.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017.

[8] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[9] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[10] Tian-qiang Peng and Fang Li. Image retrieval based on deep convolutional neural networks and binary hashing learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1742–1746. IEEE, 2017.

[11] Fatemeh Sarvi, Nikos Voskarides, Lois Mooiman, Sebastian Schelter, and Maarten de Rijke. A comparison of supervised learning to match methods for product search. *CoRR*, abs/2007.10296, 2020.

[12] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[14] Oliver Strauß, Ahmad Almheidat, and Holger Kett. Applying heuristic and machine learning strategies to product resolution. In *WEBIST*, pages 242–249, 2019.

[15] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pages 58–65, 2014.

[16] Alan Wu, AJ Piergiovanni, and Michael S Ryoo. Model-based behavioral cloning with future image similarity learning. In *Conference on Robot Learning*, pages 1062–1077. PMLR, 2020.

Acknowledgment

The authors especially thank the University of Pennsylvania CIS522 Deep Learning instructors, Dr. Ungar, Dr. Koerding, and the podmates and TAs (especially Michael Zhou), for their advice and support throughout this project.

Appendix: Feedback to Pod Teams

- Team 1 - Reddit Sentiment Analysis (95/100): As Penn students, it's nice to see a student sentiment research of social media because the mental health is significant especially during the pandemic. It's interesting to see the comparison results among the students in different Ivy League schools. Tuning the state-of-the-art BERT models and tackling the bias of different platforms are quite challenging. Good job!
- Team 2 - Emotion Recognition in Speech (95/100): Wow. A model to extract the emotional state of the speaker has huge benefit especially to the autistic people. The preliminary results of various models with the highest accuracy 65% are quite neat. Overall this project is brilliant because it involves lots of domain knowledge like voice conversion. Well done!