Leveraging Machine Learning for Automatic Product Matching in E-commerce

Steven Zhang, Ziyun Yu

Abstract

The growth of e-commerce has led to increased competition among online retailers. Product matching is a way to offer competitive rates, but it's challenging to identify similar products among the vast number listed. To address this, machine learning methods like ResNet, DenseNet, EfficientNet, and BERT models are being used to analyze product features and attributes. In this paper, we propose a novel approach that leverages ResNet and BERT models to improve accuracy and efficiency. Our model uses deep learning-based architecture to analyze multiple sources of product information and generate similarity scores. This can help retailers offer products at competitive rates and improve the customer experience on e-commerce platforms. Code is available at https://github.com/qingYzhang/Leveraging-Machine-Learning-for-Automatic-Product-Matching-in-E-commerce.

1 Introduction

To improve the efficiency of product matching and identification in e-commerce, machine learning methods have been increasingly used to analyze product features and attributes. Our proposed approach combines ResNet and BERT models and leverages a neighborhood blending technique and a formula that balances the contributions of each model. Moreover, we address the issue of different measurement units in product descriptions and use a technique we call "chiseling" to force groups of products into a desired distribution. We also explore the use of the ArcFace loss function for improved accuracy in the embeddings. Our experiments demonstrate that this approach outperforms previous methods and has the potential to help retailers offer products at more competitive rates while enhancing the customer experience on e-commerce platforms.

2 Dataset

The dataset is from Kaggle and contains metadata and images for a set of postings from Shopee, an e-commerce platform in Southeast Asia and Taiwan. The goal of the project is to identify products that have been posted repeatedly, even if they have different titles or images. The dataset includes a training set and a hidden test set, with roughly 70,000 images in the test set. The training set includes posting ID, image ID, perceptual image hash, title, and label group. The label group is an ID code for all postings that map to the same product. The test set does not include label group information. The project will require pre-processing of the data, feature extraction, and machine learning algorithms to identify similar products. The final output should be a list of posting IDs that match each other, up to a maximum of 50 matches per posting.

3 Solution

3.1 Train

3.1.1 Loss Function - ArcFace

ArcFace is used as the loss function in the training part of our model. ArcFace is a widely-used loss function in deep learning for face recognition, but it has also been successfully applied to other data modalities, including text and images. The primary advantage of ArcFace is its ability to enhance the discriminative power of the learned feature embeddings, making them more separable and easier to classify. This is achieved by incorporating an additive angular margin into the softmax loss function, which encourages the model to learn more discriminative features by maximizing the angular distance between the feature embeddings of different classes. [1]Compared to other loss functions, such as the traditional softmax and triplet loss, ArcFace has been shown to be more effective in large-scale recognition and classification tasks with diverse classes. By incorporating ArcFace into both our text and image models, we were able to significantly improve their accuracy and efficiency for product matching and identification tasks.

3.2 Image Model

3.2.1 DenseNet

DenseNet is a popular deep-learning model for image recognition that has been widely used in various computer vision applications, including product

matching and identification. However, our experiments show that using DenseNet with a ResNet backbone for our image model results in a significantly lower F1 score compared to using ResNet alone. This outcome raises the question of whether the added complexity of DenseNet is necessary for our task or if it is actually hindering performance. Further analysis suggests that the high diversity of classes in our dataset may have contributed to this lower performance of Densenet. The Densenet architecture has a high number of parameters due to its densely connected layers, which may result in overfitting when dealing with a large number of classes. Additionally, the feature maps produced by Densenet can be highly correlated, which can lead to redundancy in the learned representations and further hinder performance in a large-scale classification task. Therefore, we do not choose DenseNet for the image model.

3.2.2 ResNet

ResNet is a widely adopted deep learning model for image recognition tasks, known for its ability to train very deep neural networks without the risk of vanishing gradients. This is due to the use of residual connections, which allow for the reuse of learned features and facilitate the flow of gradients through the network. In product matching tasks, ResNet is particularly effective in learning discriminative features that capture the fine-grained visual details of products. This is crucial for the accurate matching of products with similar appearances, such as shoes or furniture, where subtle differences in texture, color, and shape can greatly affect the matching results. Additionally, ResNet has a relatively small number of parameters compared to other deep learning models, which makes it less prone to overfitting and more suitable for large-scale image recognition tasks. We finally reached the highest F1 score with ResNet which is the image model we choose at last.

3.3 Text Model

3.3.1 Bert

We use BERT as the text model in our model. BERT is a popular deep learning model for natural language processing tasks, known for its ability to learn rich contextual representations of text. In product matching tasks that involve text, BERT has been shown to be particularly effective in capturing the subtle differences in product descriptions that are crucial for accurate matching. This is important for product-matching scenarios where the product title or description can be complex and varied, such as in the case of fashion or electronics products. Additionally, BERT has a relatively small number of parameters compared to other deep-learning models, which makes it less prone to overfitting and more suitable for large-scale text-matching tasks. Our experiments show that using BERT for product matching tasks results in a high F1 score, demonstrating the effectiveness of this model in handling complex and varied product descriptions.

3.4 Embeddings

3.4.1 KNN

Nearest neighbors are a popular approach used in various machine learning applications, including image and text matching. The idea behind nearest neighbors is that similar data points will be located near each other in a high-dimensional space. In the case of image and text matching, we can represent each image and text as a vector in a high-dimensional space through a process called embedding. Once we have these embeddings, we can calculate the distance between them to determine their similarity.
To determine the optimal value of K (the number of nearest neighbors to retrieve), we experimented with different values of K ranging from 10 to 100. Our results showed that a value of 50 provided the best balance between precision and recall. Smaller values of K resulted in a loss of precision, while larger values led to a loss of recall. We attribute this to the fact that smaller values of K may not capture enough of the most relevant neighbors, while larger values may include too many dissimilar items in the search. Thus, a K value of 50 allowed us to strike a balance between the two and achieve optimal performance.

3.5 Combining Outputs

3.5.1 Take Mean, Max, Min

To combine the outputs from the image model and text model, we tried using $D = max(D_{Resnet}, D_{Bert})$, $D = min(D_{Resnet}, D_{Bert})$, $D = (D_{Resnet} + D_{Bert})/2$ to combine the results. However, the F1 score only reaches 0.7 to 0.8 which is much lower than our expectations.
a) Mean: Taking the mean of distances would calculate the average distance between the ResNet and BERT embeddings. However, this approach assumes that the ResNet and BERT models contribute equally to the final representation. In reality, some models may perform better on certain types of data or capture different aspects of the data. By simply taking the mean, we might lose the specific characteristics of each model.

b) Max: Taking the maximum distance would consider only the largest distance between the ResNet and BERT embeddings. While this approach em-

phasizes the largest discrepancy, it may ignore the valuable information provided by the other distances. This can be a disadvantage if the ResNet and BERT models have complementary strengths and weaknesses.
c) Min: Taking the minimum distance would consider only the smallest distance between the ResNet and BERT embeddings. Similarly to the maximum approach, this may overlook important information contained at larger distances.

3.5.2 Take $D = D_{Resnet} + D_{Bert} - D_{Resnet} \times D_{Bert}$

To improve the performance of mean, max, min, we choose the formula $D = D_{Resnet} + D_{BerT} - D_{Resnet} \times D_{Bert}$. It has some advantages as follows.
a) Enhanced Complementary Information: By adding the distances obtained from ResNet and BERT embeddings ($D_{Resnet} + D_{BerT}$), we can incorporate complementary information from both models. ResNet and BERT capture different aspects of the input data, and combining their distances allows you to leverage the strengths of each model. This can lead to better discrimination between similar and dissimilar instances.
b) Non-linear Interaction: The multiplication term ($D_{Resnet} \times D_{Bert}$) introduces a non-linear interaction between the distances. This non-linear operation can help emphasize or de-emphasize certain regions of the feature space, depending on the relationship between the distances. It enables the model to capture more complex patterns and relationships between instances, potentially improving the overall performance.
c) Robustness to Noisy Embeddings: If one of the models (D_{Resnet} or D_{Bert}) provides noisy embeddings for a particular instance, the multiplication term ($D_{Resnet} \times D_{Bert}$) can help mitigate its influence. Multiplying a large distance by a small distance results in a smaller value, reducing the impact of the noisy embedding. This can make the distance calculation more robust and less sensitive to outliers or inaccuracies in the individual embeddings.
d) Flexibility and Adaptive Behavior: The formula allows the model to adaptively weigh the importance of the ResNet and BERT distances for each pair of instances. It combines the distances flexibly, taking into account the specific characteristics of the instances and their embeddings. This adaptability can be beneficial when dealing with diverse datasets and complex relationships between instances.
Overall, the combination formula you described captures complementary information, introduces non-linear interactions, and provides robustness and adaptability, which can contribute to improved performance compared to alternatives such as taking the mean or max distances. However, it's important to note that the effectiveness of the combination formula may vary depending on the specific dataset and task, so experimentation and evaluation of your particular problem are essential to validate its performance. That's why we choose this formula at last and get the highest F1 score.

3.6 Neighbor Blending

To further improve the performance, we implement Neighbor Blending, which successfully makes our score jump from 0.883 to 0.910. The goal of neighborhood blending is to use the information from neighboring items to improve the embedding of the query item and produce a clearer cluster. This is achieved by computing a weighted sum of the neighborhood embeddings, where the similarity between the query item and its neighbors is used as the weight. This sum is then added to the query item's embedding, resulting in a blended embedding. The benefit of using Neighbor Blending in the model is that it can combine the strengths of ResNet and BERT models for better accuracy and efficiency in product matching. Specifically, it involves computing the similarity score between a given product and its nearest neighbors based on both ResNet and BERT models, and then blending these scores to produce a final similarity score.

3.7 Close Matches Removing

Close matches can arise due to differences in product packaging, labeling, or even the use of different measurement systems ie 150ml vs 300ml. To address this issue, the process of unmatching based on measurement units has been used in our method. This process involves removing close matches that differ in measurement units from the list of potential matches based on a similarity threshold. By doing so, the accuracy of the product matching process is significantly improved, leading to more reliable and trustworthy matches. Through this step, our F1 score rises.

3.8 Chiseling

To make the model more robust to unseen testing data, we enforced a desired distribution of groups. This method involves first hypothesizing that the distribution of group targets in the test data is similar to that in the training data, and then removing matches from the submission until the desired distribution is achieved. This approach has several benefits. Firstly, it can help to reduce overfitting, which occurs when the model is too complex and learns to fit the noise in

the training data rather than the underlying patterns. By manipulating the distribution of the target labels, the model is forced to learn a more robust representation of the data that is less prone to overfitting. Secondly, it can help to improve the generalization performance of the model by making it more resilient to changes in the underlying data distribution. This is particularly important in real-world scenarios where the data distribution may shift over time.

The process of "chiseling" the submission to match the desired distribution involves a combination of trial and error and domain expertise. It requires an understanding of the underlying data distribution and how it relates to the problem at hand. By manipulating the distribution of the target labels in this way, it is possible to improve the accuracy of the model and achieve better results on the task of matching product listings.

Results and Discussion

Our experiments show that using a KNN search with a parameter of 50, the ResNet and BERT models perform best. Combining their results using the formula $D = D_{Resnet} + D_{Bert} - D_{Resnet} \times D_{Bert}$ improves the F1 score, achieving a maximum score of 0.9. Neighborhood blending and removing close matches also significantly improve the F1 score, elevating it to 0.908.

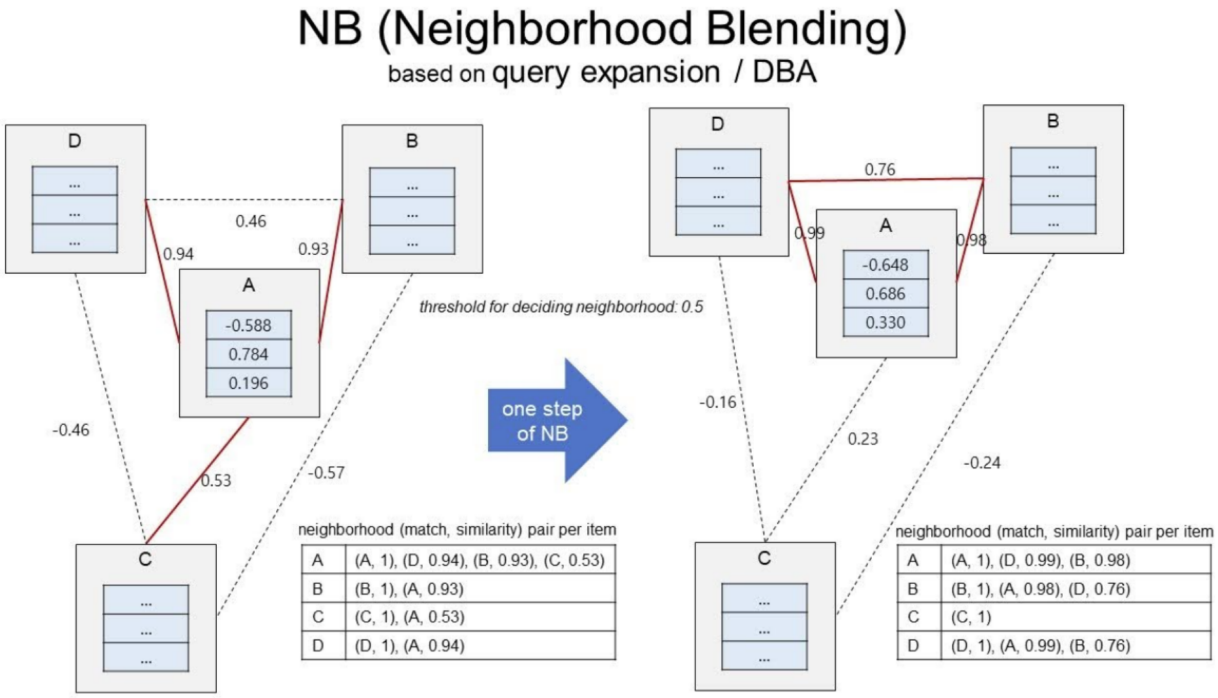
Overall, our findings demonstrate the effectiveness of combining multiple models and preprocessing steps in improving the accuracy of the product-matching process.

Future work could explore the use of other models, such as EfficientNet, NFNet, and preprocessing steps to further improve the accuracy of the matching process.

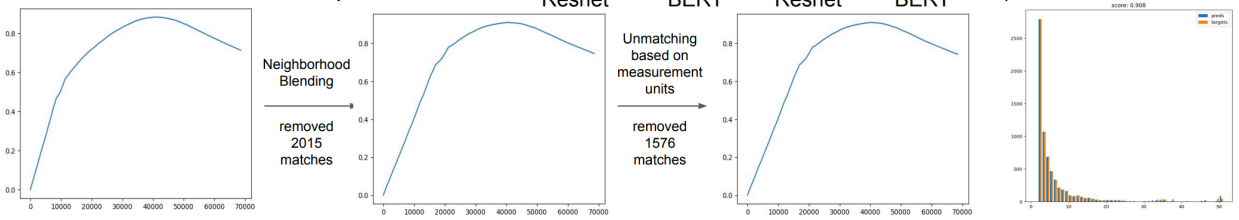
References

[1] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4, 2018. Available: `https://arxiv.org/abs/1801.07698`

Appendix:



Default (Resnet, D = D_{Resnet} + D_{BERT} - D_{Resnet} * D_{BERT}, 50)



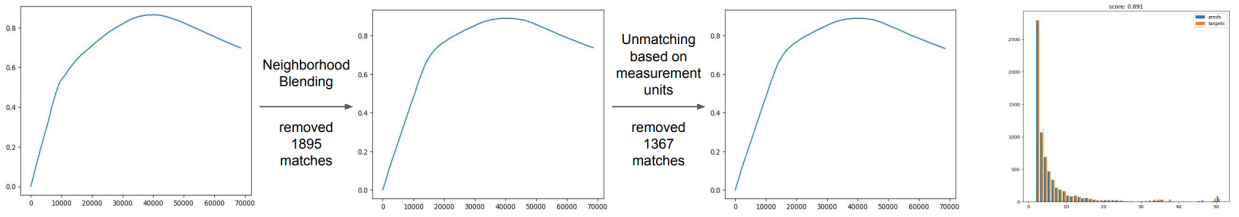
0.883 at 5.959281921386719 pairs or 0.765 threshold

0.909 at 5.971979141235352 pairs or 0.891 threshold

0.910 at 5.889959335327148 pairs or 0.891 threshold

F1 Score: 0.908

Combine the Neighbours by Taking Max



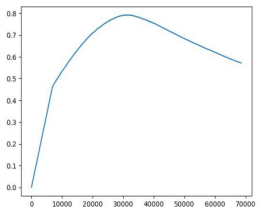
0.865 at 5.9159369468688965 pairs or 0.686 threshold

0.889 at 5.935055255889893 pairs or 0.796 threshold

0.890 at 5.863105773925781 pairs or 0.794 threshold

F1 Score: 0.891

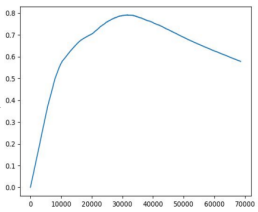
Combine the Neighbours by Taking Min



0.792 at 4.484675884246826
pairs or 0.380 threshold

Neighborhood Blending

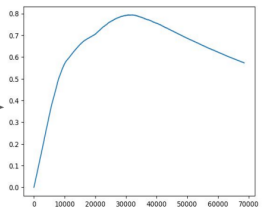
removed 2651 matches



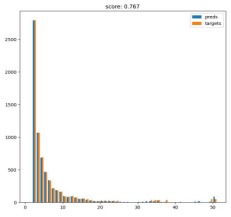
0.791 at 4.600116729736328
pairs or 0.386 threshold

Unmatching based on measurement units

removed 2789 matches

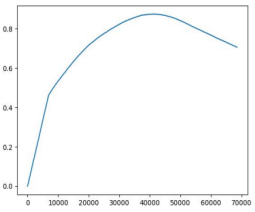


0.793 at 4.497664928436279
pairs or 0.386 threshold



F1 Score: 0.767

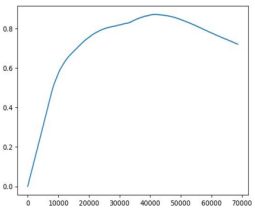
Combine the Neighbours by Taking Mean



0.875 at 5.950963020324707
pairs or 0.459 threshold

Neighborhood Blending

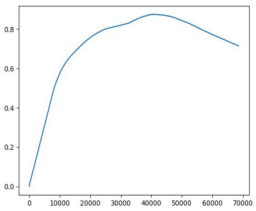
removed 2067 matches



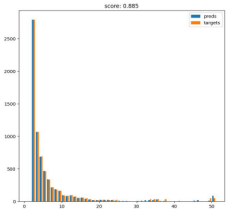
0.871 at 6.074722766876221
pairs or 0.486 threshold

Unmatching based on measurement units

removed 1953 matches

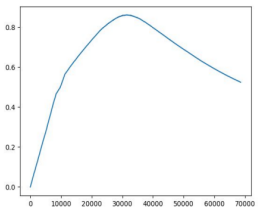


0.874 at 5.939579486846924
pairs or 0.485 threshold



F1 Score: 0.885

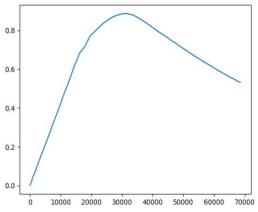
Find the Top 10 Nearest Neighbours for Each Row



0.861 at 4.585522651672363
pairs or 0.766 threshold

Neighborhood Blending

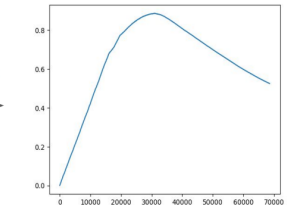
removed 2662 matches



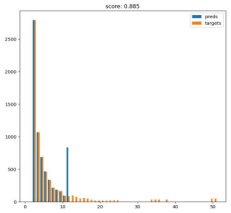
0.888 at 4.538966655731201
pairs or 0.899 threshold

Unmatching based on measurement units

removed 2281 matches

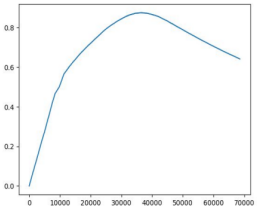


0.887 at 4.529188632965088
pairs or 0.893 threshold



F1 Score: 0.885

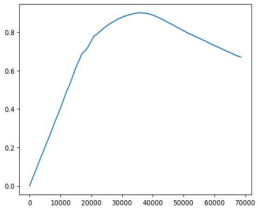
Find the Top 20 Nearest Neighbours for Each Row



0.874 at 5.317133903503418
pairs or 0.766 threshold

Neighborhood Blending

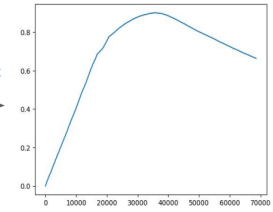
removed 2464 matches



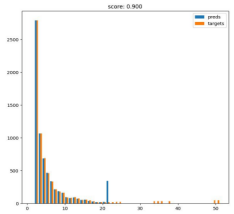
0.902 at 5.222125053405762
pairs or 0.899 threshold

Unmatching based on measurement units

removed 2092 matches

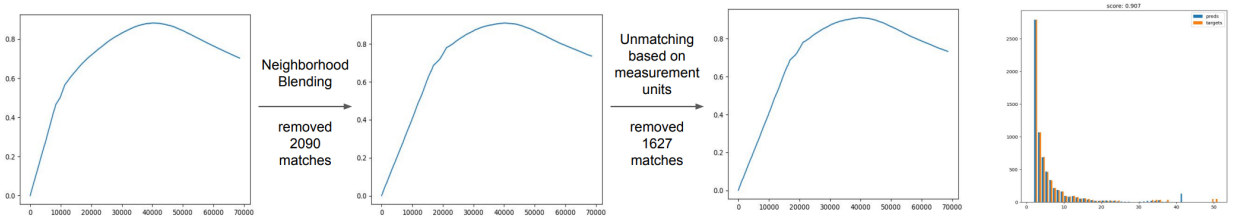


0.902 at 5.236573219299316
pairs or 0.891 threshold



F1 Score: 0.900

Find the Top 40 Nearest Neighbours for Each Row



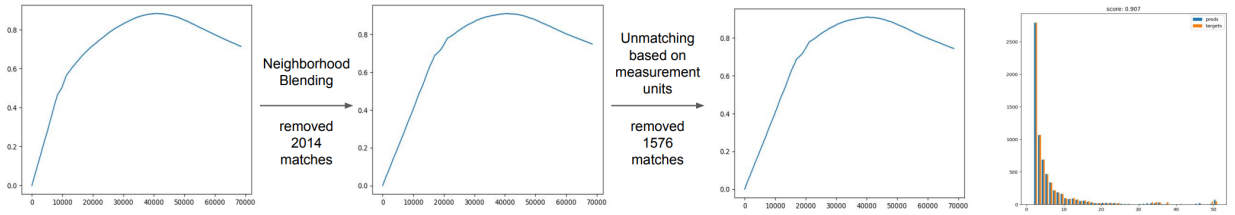
0.882 at 5.872591972351074 pairs or 0.766 threshold

0.909 at 5.9185638427734375 pairs or 0.891 threshold

0.909 at 5.836544036865234 pairs or 0.891 threshold

F1 Score: 0.907

Find the Top 100 Nearest Neighbours for Each Row



0.883 at 5.959281921386719 pairs or 0.765 threshold

0.909 at 5.971979141235352 pairs or 0.891 threshold

0.910 at 5.889959335327148 pairs or 0.891 threshold

F1 Score: 0.907

FINAL MODEL