Co-Purchase prediction based on Visual and Non-Visual on Shoes Subcategory of Amazon Item Features

James Zhao
UC San Diego
jjz005@ucsd.edu

Sravya Balasa
UC San Diego
sbalasa@ucsd.edu

Savannah Collyer
UC San Diego
scollyer@ucsd.edu

Yixuan Zhou
UC San Diego
yiz044@ucsd.edu

ABSTRACT

Within fashion recommendation, many similarities between certain items are reliant on certain features and representations present within a certain image. For example, there exists certain styles of clothes that would be worn in the same outfit (ex: scarves and boots), and would subsequently be more likely to be co-purchased. By co-purchased, we mean two items that were bought by the same user. In this project, we attempted to model this relationship between items from Amazon's Clothing, Jewelry, and Item catalog by using images, metadata, and user interactions with the items from Amazon's dataset. In this paper, we describe our predictive task, dataset, methodologies, results, and models that we utilized to model these fashion co-purchases, which, in reality, are used to create fashion recommendations on e-commerce websites.

INTRODUCTION

It is clear that the styles of objects, like clothes, which are visually perceived by humans, drastically influence the affection of humans to the item. The efforts that are devoted to modeling fashion trends are able to achieve some success with collaborative filtering and personalized recommendation, yet, the visual representation of the items are seldom being considered. However, the development of computer vision with state of the art approaches involving deep learning are able to perform many tasks, including classification, segmentation, and etc. The modern tools that we have from the development of computer vision are applicable to images such as fashion images from e-commerce websites. Therefore, in this paper, we are interested in exploring how the technologies from computer vision can help us facilitate the construction of a good recommendation mechanism in additional to regular non-visual techniques.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The recommendation task that we are interested in is given that some user bought an item *A*, whether they will buy another item *B*. We are using images in addition to non-visual data to accomplish this task. In this paper, we are going to compare the performance of models that make use of visual features with the models only utilizing non-visual features.

Outline

We briefly discuss some related literature to our paper in the next section. We then document the experiment that we conducted, in which we explain the dataset that we used, the models that we built, and the baselines that we compared against. We follow it by discussing the results of our baselines and models. Lastly, we discuss future work that can be done in this research direction.

RELATED LITERATURE

The most relevant work to what we used in this paper is the Image-based Recommender from this paper [4]. In fact, our model using image data is an adoption of the model stated in this paper. The models explored by McAuley modeled which clothes and accessories went well together. The model was desisgned to consider co-purchasing, substitution of items, and complementary items. Additionally, the paper evaluated outfits on various subsets of the entire Amazon product dataset (including Clothing, Games, Furniture, etc). Similarly, this gave us the idea to focus our work on a subset of the "Clothing, Shoes, and Jewelry" section of the dataset, by just using data for items that were shoes. Therefore, we decided to just focus on whether a person who bought an item *A* will be likely to buy item *B*, within the subset of the data that we chose (Shoes).

Additional work has also been done in modeling fashion trends temporally as well as visually using both personalized and non-personalized methods, using this Amazon dataset. [1] However, within our project, this was not explored due to the limited scope and time frame available. From this paper, we took the idea to create models that were unpersonalized. An example of this is predicting that *A* and *B* were co-purchased if *any* user *X* bought those two items together.

Other state-of-the-art models for item recommendation also incorporate many deep-learning based models, including neural

collaborative filtering as opposed to heuristic-based collaborative filtering using Similarity metrics. Other deep learning models also utilize deep neural networks, such as CNN's, RNN's, Graph Neural Networks, and AutoEncoders, in order to learn representations of useful features for scoring recommendations for both personalized and unpersonalized tasks [6].

Similar datasets to ours include a dataset for Google Local Reviews [5] which contains data about reviews of businesses from Google. The data includes geographic information from each business (metadata) and reviews, like our Amazon product dataset. Even further, for this similar dataset, a factorization machine was also used, similar to the one we utilized with textual data. Their use of factorization machines to use content-based features to enhance translation based models for sequential recommendation is similar to why we used factorization machines to incorporate fields from our items' metadata.

Another similar dataset to ours is this Reddit dataset [2] that contains submissions of Reddit posts along with metadata; ours is similarly split into two parts (reviews and metadata). Their goal was to understand how content and titles of posts influenced the popularity of posts in a community, similar to how we used the title text of items to determine if people would buy those items in our factorization machine. They achieved this goal through a statistical model that takes in the content of the submission, the submission title, the community where the submission is posted, and the time it is posted. Overall, for both datasets, the content doess influence the community that is absorbing it, like boots being used by people in colder regions and tech related posts browsed by people in engineering fields.

We also establish the non-visual baselines based on the latent-factor models / factorization machine methodologies that we learned in class. This will be further elaborated in "Models".

EXPERIMENT

Dataset

The dataset that we use is the Amazon product dataset collected in [4]. Overall, this dataset has over 142.8 million reviews. As mentioned above, we only used the "Clothing, Shoes, and Jewelry" subset of this dataset, which has 6 million reviews, and metadata for 1.5 million products. We even further used a subset of just "Shoes", which left us with 167K reviews and metadata for 58K items.

This dataset is split up into multiple parts. First we have metadata about all the items in our dataset. Second, we have review data, which is user interactions with items along with more information such as the rating given, time of rating, etc. Third, we have image features for items.

We further filtered our metadata to only include items we also have image features for.

Given two items from the metadata portion of dataset *A* and *B*, the two items can have:

1. Users who bought *A* bought *B* later.
2. Users who bought *A* bought *B* at the same time.
3. Users who viewed *A* bought *B* later.
4. Users who viewed *A* also viewed *B*.

In this paper, the relationship that we are going to focus on are the first two relationships.

This is the information we used from the metadata portion of the dataset:

- *asin*: product/item id
- *related*: a dictionary with keys for the relationships described above (ex: *also_bought*), and values with lists of item ids satisfying those relationships.
- *title*: name of the product
- *imUrl*: url of the image used by our visual model
- *categories*: different categories the product fell into (ex: [Women's Shoes, Boots])

We used the *also_bought* and *bought_together* keys in the *related* feature of the metadata to generate pairs of item ids that represented the two classes of co-purchased and non co-purchased items. For example if *XYZ* had *ABC* under its *also_bought* list, (*XYZ*, *ABC*) would be a co-purchased pair with an expected output of "1" representing co-purchased. Similarly, we randomly generated pairs of items that were not co-purchased so our final data fed to our model for training and testing was balanced. These had an expected output of "0" representing not co-purchased.

In addition, we calculated the word counts for the *title* and *categories* features for the non-visual model. We then used the word counts to one hot encode a subset of the most popular words from both features (86 for categories, 3000 for words from titles).

Rank	Word
1	womens
2	shoes
3	mens
4	black
5	sandals

(a) Top 5 most popular words in the *title* feature

Rank	Word
1	shoes
2	jewelry
3	clothing
4	women
5	sandals

(b) Top 5 most popular words in the *categories* feature

Table 1: Top 5 most popular words feature

Lastly, from review dataset, we only used the *overall* feature which represents the rating a product was given on a scale from 1-5. After analyzing the dataset, we noticed that a majority of items were given a 5.0 rating.

There are in total 57985 shoes in the dataset that we considered, and there are 121387 pairs of shoes that have a positive relationship. When generating our pairs, we make a 'bidirectionality assumption' that if the pair (X, Y) exists, we also add

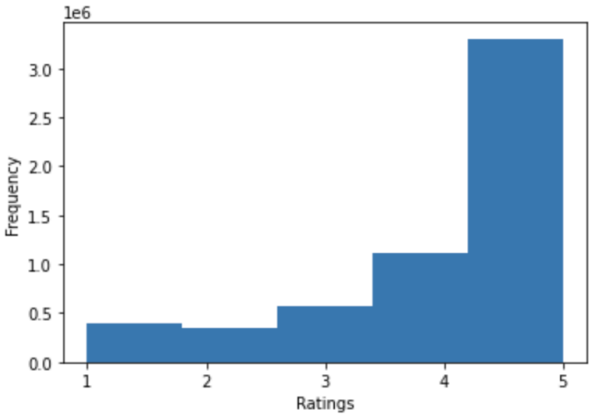

Figure 1: The distribution of product ratings

the pair (Y, X). We then augment the dataset with randomly selected negative pairs to generate the final dataset that we use. In total, the dataset contains $121387 \times 2 = 242774$ pairs of data, with half of it being the positive pairs and half of them being the negative pairs.

Here are some visualisation of models, we have the positive pairs examples being:

Figure 2: A pair of shoes that are positively related to each other.

Through this example we might be able to hypothesize that someone lives near the beach with a warm weather will buy flip flops and slippers together.

A negative pairs example is:

Figure 3: A pair of shoes sampled to be negatively related to each other.

In this example we may conjecture that someone lives in a place that is so cold that they need to wear boots will be too cold to wear sneakers outdoors.

To examine the visual embedding of each item, we performed principal component analysis on the whole shoe dataset and randomly selected 50 items to plot their dimension reduced images here.

As we can see from this figure, there exists some clusters that we can see in the dimension reduced graph. For example, the boots are clustered towards the left, flip flops are located at the bottom of the graph, and leather shoes are shifted to the top-right corner of the image. This motivates us to utilize the visual embedding of each item when designing the model.

Figure 4: The distribution of two dimensional representation of the embedding of 50 randomly selected items

Additional features that were added varied per model, so that will be discussed further in the "Model" section.

Predictive Task

For our predictive task, given two items i and j, as well as their associated features, predict the probability ($p_{i,j}$) that those items have been co-purchased. Our method of evaluation will be prediction accuracy (with $p_{i,j} >= 0.5$ corresponding to 'copurchased' and $p_{i,j} < 0.5$ corresponding to 'non-copurchased').

The intuition of the predictive task is if two items are co-purchased, then it should have some certain signals from the visual representation indicating that the two items are "similar", so that one user who bought the first item (indicting the user is attracted by the item visual feature) will also buy the section item.

Model

This paper discusses three models, one that used the visual/image data, one that used user interactions, and one that uses other textual metadata.

Non-Visual Model (Baseline)

For the baseline model, we used the review dataset to create user and item interaction sets (ex: userPerItem and itemPerUser). We then used these sets to compare the **Jaccard** similarity of the two items against a threshold. We originally elected to use Jaccard because we had user interaction data from the reviews portion of our dataset and examples co-purchased and non co-purchased pairs that we created that

we created from the metadata portion of our dataset. If the similarity was above the threshold, we predicted 1 to signify that the items were likely to be co-purchased and 0 if not co-purchased. We tested the performance of the model using the positive and negative pairings created from the metadata. We used the following Jaccard equation:

$$sim(\textbf{item1}, \textbf{item2}) = \frac{\mathbf{s}_1 \cap \mathbf{s}_2}{\mathbf{s}_1 \cup \mathbf{s}_2}$$

This model did not perform much better than random guessing. Further results and analysis are discussed under the "Non-Visual Model Performance" section.

Non-Visual Model

To beat our baseline model only using non-visual data, we used a **Factorization Machine** using the $fastFM$ Python library. Generally speaking, a factorization machine is a supervised learning algorithm that can be used for both classification and regression. The main defining factor is that it extends from a linear model because interactions between features in high dimensional sparse datasets/matrices (matrix where most of the elements are zero) are captured.

Our manipulation of our data for the factorization matrix ended up generating a sparse matrix due to utilizing one hot encoding, which was the perfect input to achieve accurate results. As mentioned earlier, we generated positive and negative pairings created from the metadata. Given a pair of items (A, B), a row in the matrix contained the following:

1. Average rating for A from the review data
2. Average rating for B from the review data
3. One-hot encoding of the *categories* field from A's metadata
4. One-hot encoding of the *categories* field from B's metadata
5. One-hot encoding of the $title$ field from A's metadata using the 3000 most popular words from all titles
6. One-hot encoding of the $title$ field from B's metadata using the 3000 most popular words from all titles
7. One-hot encoding of A's item id
8. One-hot encoding of B's item id

In the end, this gave an approximately matrix of size approximately (240K, 66K).

Some state of the art models that were studied in CSE158 that we utilized involved feature engineering using textual information from the product title and training a factorization machine, like the one we used, to learn interactions between the features. The intuition behind this is that some items may be co-purchased due to some semantic similarities between the types/features of shoes (which would usually be captured in the title, e.x. sandal versus slipper), and it would be useful to capture these types of interactions.

However, it may also be the case that the product title may not contain sufficient information to correctly identify whether or not two shoes were co-purchased, leading us to believe that perhaps visual features near the output-layer of a CNN may provide more complex information that product titles otherwise wouldn't be able to provide.

Visual Model

The first problem to note is that our predictive task is not as trivial as comparing the similarity of two images because the raw images are not translation invariant. However, we should expect that shifting a image by a some fixed amount should have minimal effects on the decision process of a person to buy it or not.

The second problem to note is that it is not necessarily the case that people want to buy similar items, like two pairs of boots, but might instead want to buy a boot and a sneaker. So, we might expect that even if two items are extremely similar, people will not buy the second one because they already own the first. Developing a model that understands these complexities is crucial to achieving a high accuracy.

To tackle the first problem stated with a model that used the visual data, we used a pretrained Alexnet [3] to generate an embedding of the images. Through the convolutional layers of the Alexnet, it can be argued the translation invariance is preserved. This property is also justified by the good performance that it has at image classification. The second problem is resolved by the design of our models, in which we devise trainable distance metric which is known as Mahalanobis Distance.

For our visual model, we utilized the image features that were supplied in the metadata of our Amazon dataset, which consists of a 4096-length float vector of image embedding from images that were passed through a pretrained Deep CNN. Our model makes the assumption that co-purchases are bidirectional in order to simply our prediction task. Afterwards, we learned a distance metric described as follows, known as a **Mahalanobis Distance**:

$$d(i,j) = (\mathbf{x}_i - \mathbf{x}_j)\mathbf{Y}\mathbf{Y}^T(\mathbf{x}_i - \mathbf{x}_j)^T$$

where $\mathbf{Y}$ is a (4096, $\mathbf{k}$) dimensional matrix, where each feature k is a different weighted combination of the 4096 image features. Thus, the model is able to 'learn' complicated representations of image features that are particularly salient for this classification task. The value of $\mathbf{k}$ is a hyper parameter in this model. For our experiments, we utilized K = 1 (a specific case of the Mahalnobis Distance, known as **Weighted Nearest Neighbor**), K = 10, and K = 50.

After obtaining a distance metric, the distance is passed through a modified version of the sigmoid activation layer. This layer utilizes a term $\mathbf{c}$ as a threshold, in which distances greater than c are considered 'non-copurchased', and distance less than c are considered 'copurchased'. This activation function is described as follows:

$$\sigma(\ d(i,j)\ ) = \frac{1}{1 + e^{-(\ c - d(i,j)\ )}}$$

The loss function that we optimized was Binary Cross Entropy Loss.

$$\mathbf{L}(i,j,y_{i,j}) = y_{i,j} * log(\ p_{i,j}\ ) + (\ 1 - y_{i,j}\ ) * log(\ 1 - p_{i,j}\ )$$

Our model was implemented using Pytorch, and trained using a Pytorch Extension called Pytorch Lightning. Our model was trained using Minibatch stochastic gradient descent, using the Adam optimizer with a learning rate of 0.001 and a batch size of 128. Our model was trained for 15 epochs in each case, as our model's validation performance begins to show signs of overfitting, and we were partially constrained by training time and GPU availability.

RESULTS

Non-Visual Baseline Model Performance

When only Jaccard similarity was utilized using the same set of co-purchased items discussed in "Dataset", our model outputs were more or less random, which may be because each user does not always review many unique shoes, or more generally items, that we specifically have metadata for. Specifically, the accuracy was 0.507558, which is close to random. Since our data was not suited to having a large quantity of user interactions, we found the Jaccard similarity metric to be relatively unstable and a non-predictive feature.

Non-Visual Model Performance

The resulting accuracy for the Factorization Machine on our test set (80/20 training/test split) was 94.7002 percent. The accuracy shows us that the Factorization Machine with the inputted matrix was able to classify whether two items were co-purchased or not with 94.7002 percent accuracy, which is significant. In comparison to our non-visual baseline, the accuracy was much higher for this model, due to how the format of our data was perfectly aligned with what the model expected. Heavily utilizing the metadata for each item rather than user interactions from the reviews lent itself to having many fields to one hot encode, which generated the features for the ideal sparse matrix, as discussed in the "Model" section above.

Visual Model Performance

The results of the Visual Model's performance are summarized in Table 2. We also recorded the loss and accuracy metric per training epoch in 5 and 6. Due to limits in resources and having to train models locally, we were not able to explore too many hyper parameters and were partially limited in how long we could train models for.

Interestingly, the Dimensionality of the Mahalanobis matrix that performed the best was K = 10. However, this was not observed in this paper [4], in which increasing the dimensionality of K improved the model's performance. However, this difference could have occured for a multitude of reasons, include different methods of optimizations (Adam optimizer vs L-BFGS), weight initialization, or perhaps our bi-directional assumption. Nevertheless, our model still performed extremely well for K=10.

Hyperparameter	Accuracy
K = 1	0.8576
K = 10	0.9369
K = 100	0.7602

Table 2: Results summarizing performance of the Visual Model. Our models were evaluated on accuracy using an 80-20 train/test split, with various dimensions for the Mahalanobis matrix.

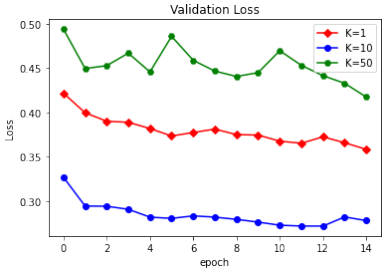

Figure 5: Validation loss plots for K=1, 10, and 50

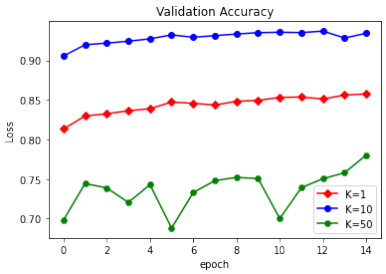

Figure 6: Validation accuracy plots for K=1, 10, and 50

In Figure 7, we plot the heatmap of $\mathbf{YY}^T$ matrix for calculating the Mahalanobis distance using the $\mathbf{Y}$ that we got when K=10. Since the matrix $\mathbf{YY}^T$ is 4096×4096, we performed a max-pooling so that the heatmap has better visualization result.

Since the distance

$$d(i,j) = (\mathbf{x}_i - \mathbf{x}_j)\mathbf{YY}^T(\mathbf{x}_i - \mathbf{x}_j)^T,$$

the matrix $\mathbf{YY}^T$ tells us how we would associate the embedding of $\mathbf{x}_i$ and $\mathbf{x}_j$ when calculating the distance. From the heatmap, we can clear that there is a positive association of each entry in the embedding corresponding entry to the other embedding, as the main diagonal contains large value. This can be explained by two items visual embeddings should be similar if they are similar. And if they are similar, the corresponding entry should have similar value. Hence, when evaluating their distance, different values at the same entry should have large and positive contribution to the distance of the two items, indicating they are not similar.

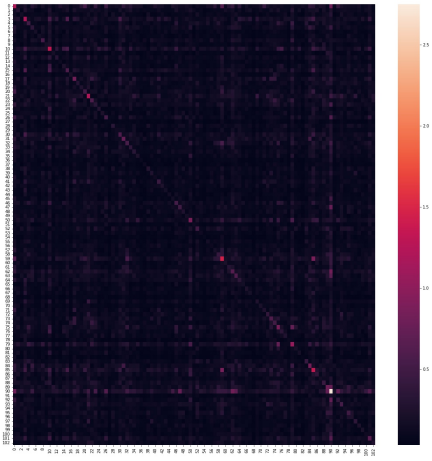

Figure 7: Heatmap of the Mahalanobis distance

We can see that there are other spots other than the main diagonal also have positive values, this might indicate how our model differentiate different styles, categories, and etc. from the embeddings.

To further demonstrate the accuracy of our visual model, we sub-sampled 10k positive pairs and 10k negative pairs and run through our models. We store the predicted probability and plot the predicted results as follow:

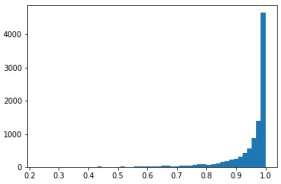

Figure 8: Positive pairs prediction

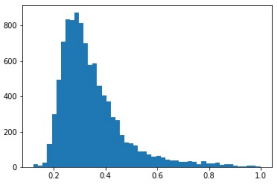

Figure 9: Negative pairs prediction

As we can see, our model is extremely well at assigning positive prediction label to positive pairs. Our model performs relatively good with negative pairs as well, as we can see most of the results are less than 0.5, which indicates it will assign negative prediction label to the negative pair.

It seems as though both the visual and non-visual representations performed relatively similarly. As a result, we believe that the dataset may simply be considered 'easy', in which semantic similarities (whether visual or textual) or simply categorical information (e.x. sandals, women's shoes) largely determine positive co-purchases between items. It also may be the case that the 'also_bought' pairs were generated on Amazon's end using semantic similarities between items as well, and our models may be 'rediscovering' these relationships.

REFERENCES

[1] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. (2016).

[2] Jure Leskovec Himabindu Lakkaraju, Julian McAuley. 2013. What's in a name? Understanding the Interplay between Titles, Content, and Communities in Social Media. (2013).

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`

[4] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. (2015).

[5] Rajiv Pasricha and Julian McAuley. 2018. Translation-based Factorization Machines for Sequential Recommendation. (2018).

[6] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1, Article 5 (feb 2019), 38 pages. `DOI: http://dx.doi.org/10.1145/3285029`