LEARNING EMBEDDINGS FOR FASHION RECOMMENDATION

BY

SHREYA RAJPAL

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Professor David A. Forsyth
ABSTRACT

In this work, we present a novel methodology to recommend items that are compatible with a given item of clothing. Compatibility is a hard notion to capture because of its diversity and subjectivity. We propose an embedding based approach to solve this problem, and perform recommendation based on product-closeness to the given clothing item. We perform this by first decomposing the notion of product-closeness into two inter-related notions of product similarity and product compatibility. Then, we incorporate product type into our embedding mechanism, and learn different embedding networks for different product types. We evaluate our proposed strategy extensively, and demonstrate that it performs better than the baseline, and is an effective method for performing few-shot transfer to compatibility prediction tasks.
To my parents, for their love and support.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. David Forsyth for his inspiring guidance throughout my Masters program. I also want to take this opportunity to offer my gratitude to all my friends and family, for seeing me through thick and thin.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1  INTRODUCTION AND MOTIVATION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2  RELATED WORK</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Learning Embedded Representations</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Type-Specific Embeddings</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Fashion Recommendation</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Transfer Learning</td>
<td>5</td>
</tr>
<tr>
<td>CHAPTER 3  METHODOLOGY</td>
<td>6</td>
</tr>
<tr>
<td>3.1 Dataset</td>
<td>6</td>
</tr>
<tr>
<td>3.2 Decoupling Similarity and Compatibility</td>
<td>6</td>
</tr>
<tr>
<td>3.3 Loss Function</td>
<td>8</td>
</tr>
<tr>
<td>3.4 Training Process and Network Architecture</td>
<td>8</td>
</tr>
<tr>
<td>3.5 Optimization Method</td>
<td>11</td>
</tr>
<tr>
<td>CHAPTER 4  EXPERIMENTS</td>
<td>13</td>
</tr>
<tr>
<td>4.1 Baseline</td>
<td>13</td>
</tr>
<tr>
<td>4.2 Visualization of embedding</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Quantitative performance evaluation</td>
<td>15</td>
</tr>
<tr>
<td>4.4 Product Recommendation</td>
<td>20</td>
</tr>
<tr>
<td>CHAPTER 5  CONCLUSION AND DISCUSSION</td>
<td>27</td>
</tr>
<tr>
<td>CHAPTER 6  REFERENCES</td>
<td>28</td>
</tr>
<tr>
<td>APPENDIX A SUPPLEMENTARY MATERIAL</td>
<td>30</td>
</tr>
</tbody>
</table>
LIST OF TABLES

3.1 **Architecture Specification.** This table contains information about the architectures used for the Compatibility and Similarity networks. The embeddings generated from the Similarity Network are then used as input for the Compatibility Network. The Compatibility Network is trained on compatibility links, whereas the Similarity Network is trained on similarity links, and both links are non-overlapping. In contrast, the baseline is trained on the union of compatibility and similarity.

4.1 **Experimental results for Compatibility Learning Task.** This table studies results on the compatibility prediction task, therefore only the embeddings from the type-specific compatibility spaces are used for our method. We can observe that our method significantly outperforms the baseline on all metrics, signalling that our method can learn a high-quality notion of compatibility between products. Another interesting observation is that training data imbalance (as discussed in Section 3.1) causes no significant difference on the performance of the learned embeddings.

4.2 **Experimental results for the Similarity Learning Task.** This table studies results on the similarity prediction task, therefore the embeddings used for our method are the output of the similarity network described in Table 3.1. The embeddings for the baseline are the same as before. We can observe that both methods perform roughly similar – our method outperforms the baseline on AUC score, while performing worse than the baseline on Average distance ratio. This is consistent with the hypothesis that the baseline learns an embedding that is equivalent to a similarity embedding, without much notion of compatibility, in spite of being provided with additional training data and learning capability.

10
19
21
LIST OF FIGURES

3.1 Generalization performance of optimization methods. As can be seen from plots, there is no discernible difference between the optimization methods for our problem. 12

4.1 \textbf{t-SNE plot of Similarity embedding.} As we can see from this plot, the similarity embeddings embeds similar looking items of the same category nearby, and there is little-to-none overlap between items of different categories. 14

4.2 \textbf{t-SNE plot of Compatibility embedding for the bottom-shoe type pair.} In this plot, while some similar looking items of the same category are embedded nearby, there is significant overlap between items of different categories. 16

4.3 \textbf{t-SNE plot of the Baseline embedding.} In spite of providing the baseline with compatibility links, the baseline looks very close to the similarity embeddings, in that it embeds similar looking items of the same category nearby, with little-to-none overlap between items of different categories. 17

4.4 \textbf{Failure case for our method.} As we can see, even though our predicted set of compatible items does not contain the ground truth compatible shoe, the predicted shoes are still close enough to the ground truth item so as to be actually be compatible with the query item in real life. 22

4.5 \textbf{Successful case for our method.} We can see that in this case, not only does our method correctly retrieve the ground truth compatible item for the given query, but it also highlights the diversity in the set of retrieved predictions. Our method is able to predict 2 different types of footwear to go with the same pair of shorts – sneakers and sandals. 24

4.6 \textbf{Successful case for baseline.} We can observe that while the baseline correctly retrieves the ground truth compatible item in the set of predicted compatible items, there is no diversity in the set of retrieved items. 25

4.7 \textbf{Failure case for the baseline method.} As we can see, the set of predicted compatible items is nowhere close to the ground truth compatible item. Since we’re considering compatibility and similarity links independently, if the compatibility links for an item are not well maintained, then the baseline can not compensate by using alternative sources of information. 26

A.1 \textbf{t-SNE plot of the Compatibility embedding for the top-bottom type pair.} While some similar looking items of the same category are embedded nearby, there is significant overlap between items of different categories. Moreover, we can also observe some mis-labelling of categories, that cause shoes to appear in this embedding. 30

A.2 \textbf{t-SNE plot of the Compatibility embedding for the top-shoe type pair.} While some similar looking items of the same category are embedded nearby, there is significant overlap between tops and shoes. 31
A.3 **Failure case for our method on bottom-top prediction.** As we can see, even though our predicted set of compatible items does not contain the ground truth compatible bottoms, the predicted bottoms are still compatible with the query item in real life.

A.4 **Success case for our method on top-shoe prediction.** As we can see, our method very accurately predicts not just the ground truth item, but also a diverse set of compatible items.
CHAPTER 1: INTRODUCTION AND MOTIVATION

In recent years, computer vision has had many interesting applications in the domain of fashion. The image-dependant nature of fashion, combined with the fine-grained differentiation between different fashion styles, provides a rich playground for various computer vision tasks. In this work, we look at the task of clothing recommendation using image embeddings. More specifically, our objective is to recommend a set of clothes (called the recommended products) that are all compatible with a given item of clothing (called the query product).

This is an important problem, both for online retailers, as well as fashion consumers. It allows online retailers to better recommend clothes to customers, at the same time helping customers better style their own outfits.

In this work, we propose an embedding-based approach to solve this task that recommends the nearest neighbors (in the embedded space) of a query item. However, unlike previous work, we use a more sophisticated notion of inter-product distances that allows us to achieve better performance. We decompose the idea of inter-product closeness into two related notions of product similarity and product compatibility.

We formally define similarity and compatibility in the following way: two items are similar if they can be easily swapped in an outfit (such as 2 black pants that are alike). On the other hand, two items are compatible if they together make a cohesive outfit – such as a white shirt combined with a pair of black pants. Even though our objective is to recommend compatible items, we claim that first learning to perform a similarity recommendation task leads to improved generalization on the compatibility recommendation task. This is primarily because using our dataset has considerably fewer compatibility links between products compared to similarity links between products. Therefore, pre-training on similarity links provides a better performance on few-shot transfer to compatibility recommendation. The ideas of compatibility and similarity lend themselves very naturally to reasoning about clothes in terms of their types. Similarity can only be defined with respect to clothing items of the
same type. For example, 2 shirts can be similar to each other, while it’s difficult for a shirt and a pair of shoes to be similar. On the other hand, compatibility can only be defined with respect to items of different types. For example, two shirts can’t be compatible with each other. Moreover for an item, the item’s type strongly dictates what features will be important when evaluating its compatibility with items of other types. What features are important for making a pair of pants compatible with a pair of shoes, are very different from the features that are important for making the same pair of pants compatible with a shirt. Therefore, while learning compatibility embeddings for a pair of products, it is natural to learn a different compatibility embedding for a different type of product. The transfer of learning from the similarity space to the compatibility space, as well as the focus on type-specific compatibility embeddings, are the two main ideas explored in this work.

Our proposed architecture learns compatibility embeddings in a 2-step process. We first learn a type-agnostic similarity embedding, that allows us to embed similar-looking products nearby. We then use these similarity embeddings to learn a type-specific notion of compatibility. This leads to an implicit hierarchy of between the notion of similarity and compatibility, which is translated to hierarchy between the similarity and compatibility embedding spaces. In this hierarchy, similarity is trained first, and so is the primary embedding space, while compatibility is trained second, and therefore represents the secondary embedding space.

We evaluate our methodology against a strong baseline on multiple qualitative and quantitative metrics, and demonstrate that:

- Performing type-specific clothing recommendation allows us to better capture the notion of compatibility between clothes.

- Separating the notion of similarity and compatibility allows us share information even when compatibility information is scarce, leading to better performance on multiple metrics.

- Imposing a hierarchical relationship between similarity and compatibility allows us to
increase the diversity within the compatible products recommended by our method.

The rest of the document is organized as follows: Section 2 discusses related prior work, Section 3 describes our methodology in detail, Section 4 presents various qualitative and quantitative experiments that support our claims, and Section 5 discusses future extensions to this work.
CHAPTER 2: RELATED WORK

2.1 LEARNING EMBEDDED REPRESENTATIONS

Learning good embedded representations for the purpose of information retrieval and item recommendation are active areas of research. Common methods for embedding items that leverage deep neural networks include the utilization of a ‘Siamese Network’ [1] in conjunction with a margin-based loss or a triplet loss. Another approach is to solve some classification task (e.g. ImageNet classification [2]) and learn the embeddings as a side-effect. The former approach has shown a lot of success, as for instance in FaceNet [3]. However, all these methods use a singular notion of product closeness for the purposes of recommendation. In this work however, we approach product closeness in terms of two components – product similarity and product compatibility.

[4] does decompose product-wise relationships into similarity and compatibility for clothing, but their approach is agnostic to any type information in the data. On the other hand, we learn different compatibility embeddings based on the item type, and demonstrate that this leads to information sharing in the case of few shot transfer, along with improved performance.

2.2 TYPE-SPECIFIC EMBEDDINGS

Type-specific embeddings have been covered by previous works in [5, 6]. [7, 8] learn a common embedding space for all types. In contrast, our work learns different embedding spaces for different types.

2.3 FASHION RECOMMENDATION

Using computer vision for fashion recommendation has become increasingly popular in recent times [6, 9, 10, 11, 12, 4]. However, our work is novel in performing type-based
compatibility and similarity recommendation.

2.4 TRANSFER LEARNING

Recent approaches to few shot transfer include approaches such as [13, 14, 15]. However, these methods have no notion of performing type-based inference, and have never been applied to a task in the fashion recommendation setting.
CHAPTER 3: METHODOLOGY

In this chapter, we describe in detail our methodology of learning compatibility and similarity embeddings, along with the training procedure we used for generating them.

3.1 DATASET

We use the Amazon Products Dataset [16, 17] for training the embeddings. This dataset consists of product reviews, meta-data and images for 142.8 million products on Amazon across a wide range of categories, including books, electronics, automotive, etc. The metadata attribute contains co-purchasing links between products, which we use for extracting similarity and compatibility links, as we describe in Section 3.2.

For our purposes, we only use images and metadata attributes, and restrict our category to the ‘Clothing, Shoes and Jewelry’ category, which consists of 5.7 million products. This dataset consists of clothing items from many different types such as tops, bottoms, outerwear, dresses, etc. Because the number of type-spaces grows quadratically with the number of types, we restrict our experiments to 3 types – tops, bottoms and shoes, and filter all products that have no image associated with them.

We resize all images in this dataset to be of size $227 \times 227 \times 3$, and perform z-score normalization by subtracting mean $(0.485, 0.456, 0.406)$ from each channel, and then dividing each channel by standard deviation $(0.229, 0.224, 0.225)$ respectively. Hereinafter, we refer to this dataset as the Amazon clothing dataset. We partition the dataset into 80% training, 19% testing, and 1% validation data.

3.2 DECOUPLING SIMILARITY AND COMPATIBILITY

The Amazon clothing dataset consists of 4 types of co-purchasing links between products, namely: also bought, also viewed, bought together, and buy after viewing. We combine
all 4 types of links into one ‘related’ link, and then split ‘related’ links into similarity and compatibility links based on the product types. More formally, let $x^\tau_i$ represent a product $x$ with index $i$ and type $\tau$, where $\tau \in \{t \text{ (top)}, b \text{ (bottom)}, s \text{ (shoe)}\}$.\footnote{Henceforth, both the subscript and superscript of $x^\tau_i$ can be omitted, in case they are not required for the mathematics.}

Then, the set of products that share a ‘related’ link with $x^\tau_i$ is represented by the mapping $R(x^\tau_i)$. Furthermore, the set of similar products $(S(x^\tau_i))$ is

$$S(x^\tau_i) = \{x^v_j \forall x_j^v \in R(x^\tau_i), \text{s.t. } v = \tau\},$$

and the set of compatible products $C(x^\tau_i)$ is

$$C(x^\tau_i) = \{x^v_j \forall x_j^v \in R(x^\tau_i), \text{s.t. } v \neq \tau\}.\tag{3.2}$$

An interesting observation about the Amazon dataset is that both compatibility and similarity relationships are not constrained to be symmetric. If a top $x^t_i$ is compatible with bottom $x^b_j$, then that does not necessarily mean that $x^b_j$ is also compatible with $x^t_i$. For this dataset, the asymmetry exists because of how the compatibility and similarity relationships are populated – based on Amazon co-purchasing and browsing information. An effect of asymmetry is an imbalance in the training dataset, such as a lot more triplets existing for top-shoe compatibility than for shoe-top compatibility, as well as a lot of similarity links existing, with much fewer compatibility links.

While symmetry as well as a balanced training set would have been a desirable property, we demonstrate that our method works in spite of their absence.

### 3.2.1 Triplet Sampling

Sampling triplets is very important for good performance on learning embeddings. For the learning the similarity embeddings, for each anchor product $x^\tau_a$ we sample a positive
product $x_p^r$ randomly from $S(x_a)$, and a negative product $x_n^v$ randomly from the entire list of products. Note that both here both $x_p^r, x_p^r$ are of the same category, while the category of $x_n^v$ could be the same as or different from $x_a^r$.

In contrast, while sampling triplets for learning the compatibility embeddings, for each anchor product $x_a^r$, we sample a positive $x_p^v$ from $C(x_a^r)$, and sample the negative $x_n^v$ from all products that have the same category as $x_p^v$. This ensures that the distance comparison of $x_a^r, x_p^v$ and $x_a^r, x_n^v$ for back-propagating the loss is meaningful. For training, both similarity and compatibility samples have a support of 10.

3.3 LOSS FUNCTION

We use the triplet loss to learn both the compatibility and similarity embeddings. For a set of triplets of the form $(x_a, x_p, x_n)$ where $x_a$ is the anchor point, $x_p$ is a positive example that shares a link with $x_a$, and $x_n$ is a negative example that doesn’t share a link with $x_a$, the standard triplet loss [3] is formulated as:

$$
L = \sum_{(a,p,n)} [||f(x_a) - f(x_p)|| - ||f(x_a) - f(x_n)|| + \alpha],
$$

(3.3)

where $\alpha$ is a small constant.

For generating similarity embeddings, $x_a, x_p$ have the same type. For compatibility embeddings, $x_p, x_n$ have the same type, which is necessarily different from the type of $x_a$.

3.4 TRAINING PROCESS AND NETWORK ARCHITECTURE

The similarity embedding is trained first on similarity links. This is followed by using the output of the similarity network (on compatibility triplets) as input to train compatibility networks, wherein we minimize the distance between the outputs of the compatibility network in compatibility space.

More formally, let $f(x_i^r)$ represent a mapping between a product $x_i^r$, and it’s embedding
in the similarity space, represented by $\mathcal{F}$. Then, the objective while learning the similarity space is:

$$
\min_{f} \| f(x_i) - f(x_j) \|, \quad (3.4)
$$

where $x_j \in S(x_i)$.

Note that this space is common to all the clothing types, or in other words, $f$ remains constant no matter what the $\tau$ for the $x_i$ is. In contrast, the compatibility embedding space is unique for each type and each pair of types, such that for a pair of products $x^\tau_i$ and $x^\nu_j$ where $x^\nu_j \in C(x^\tau_i)$, the mappings $g^\tau_{\tau,\nu}, g^\nu_{\tau,\nu}$ map the embeddings of $x^\tau_i, x^\nu_j$ into a common compatibility space (represented by $\mathcal{G}_{\tau,\nu}$). The objective while learning all $g$ functions is:

$$
\min_{g} \| g^\tau_{\tau,\nu}(f(x^\tau_i)) - g^\nu_{\tau,\nu}(f(x^\nu_j)) \|, \quad (3.5)
$$

where $x^\nu_j \in C(x^\tau_i)$ and $\nu \neq \tau$.

Note that for $k$ types of clothing products, there are going to be $\frac{k!}{(k-2)!2!}$ number of compatibility spaces – one for each pair of clothing types. Moreover for each compatibility space, there are going to be 2 $g$’s (1 for each type) that map products into that $\mathcal{G}$, leading to a total of $\frac{k!}{(k-2)!}$ number of $g$’s.

We train the proposed architecture using PyTorch. The Similarity Network $f$ is initialized with a pre-trained AlexNet [2], with the last layer replaced with a fully connected layer with 256 units. The output of the network is normalized to lie on a unit sphere – this step is ends up being crucial for fast convergence. For completeness, the configuration of the architecture is provided in Table 3.1.

The compatibility network ($g^\tau_{\tau,\nu}$) uses the the output of the similarity network ($f$) for its input. The input is then passed through 2 fully connected layers with a ReLU activation, as shown in Table 3.1. There will be a different compatibility network for each $\mathcal{G}_{\tau,\nu}$ that, as well as for each type $g^\tau_{\tau,\nu}$. Therefore, there will exist 6 Compatibility Networks, namely:
### Similarity Network

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Input</th>
<th>Output</th>
<th>Kernel</th>
<th>Stride</th>
<th>Activ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional</td>
<td>3</td>
<td>64</td>
<td>11</td>
<td>4</td>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Convolutional</td>
<td>64</td>
<td>192</td>
<td>5</td>
<td>1</td>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>192</td>
<td>192</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Convolutional</td>
<td>192</td>
<td>384</td>
<td>3</td>
<td>1</td>
<td>ReLU</td>
</tr>
<tr>
<td>Convolutional</td>
<td>384</td>
<td>256</td>
<td>3</td>
<td>1</td>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>256</td>
<td>256</td>
<td>3</td>
<td>1</td>
<td>ReLU</td>
</tr>
<tr>
<td>Dropout (50%)</td>
<td>256</td>
<td>256</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>256 × 6 × 6</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>ReLU</td>
</tr>
<tr>
<td>Dropout (50%)</td>
<td>4096</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>4096</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>ReLU</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>4096</td>
<td>256</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Compatibility Network

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Input</th>
<th>Output</th>
<th>Kernel</th>
<th>Stride</th>
<th>Activ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Connected</td>
<td>256</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>ReLU</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>64</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>ReLU</td>
</tr>
</tbody>
</table>

Table 3.1: **Architecture Specification.** This table contains information about the architectures used for the Compatibility and Similarity networks. The embeddings generated from the Similarity Network are then used as input for the Compatibility Network. The Compatibility Network is trained on compatibility links, whereas the Similarity Network is trained on similarity links, and both links are non-overlapping. In contrast, the baseline is trained on the union of compatibility and similarity.

- \( g_{t,b} \) (projects *tops* into \( G_{t,b} \)),
- \( g_{b,t,b} \) (projects *bottoms* into a \( G_{t,b} \)),
- \( g_{s,b} \) (projects *shoes* into a \( G_{s,b} \)),
- \( g_{b,s,b} \) (projects *bottoms* into a \( G_{s,b} \)),
- \( g_{s,t,s} \) (projects *shoes* into a \( G_{t,s} \)),
- \( g_{t,t,s} \) (projects *tops* into a \( G_{t,s} \)).

Each of these 6 networks is initialized in the same way, as shown in Table 3.1. However, each network is trained on different samples from the dataset. For example, in the process
of learned the $G_{t,b}$ space, *tops* are passed through $g_{t,b}^t$ while *bottoms* are passed through $g_{t,b}^b$. The Euclidean distance between the outputs of $g_{t,b}^t$ and $g_{t,b}^b$ is back-propagated through each network, and the weights of the networks are updated accordingly.

### 3.5 OPTIMIZATION METHOD

Recent work has discussed at length about the generalization capability of different optimization methods [18, 19]. For training our embeddings, we explore 4 different optimization methods – Adam, SGD, Adgrad and RMSProp – to select the method with the best generalization performance. The results of this experiment are shown in Figure 3.1. We tried each optimization method on the task of predicting the embeddings in the similarity space, and repeated each experiment 10 times for the best value of the learning rate for each method. As can be observed from Figure 3.1, in the case of our problem, there is no difference between the different methods. Therefore, for the rest of the experiments, we train using SGD.
Figure 3.1: **Generalization performance of optimization methods.** As can be seen from plots, there is no discernible difference between the optimization methods for our problem.
CHAPTER 4: EXPERIMENTS

4.1 BASELINE

We use a strong neural network based baseline for comparison. The baseline architecture is connected exactly as our method (described in Table 3.1), except that there is no demarcation between similarity and compatibility networks – the entire architecture is treated as a single network. Moreover, the union of similarity and compatibility links is used to train the entire network.

The baseline network also optimizes a Triplet loss using SGD. Therefore, the baseline network has the exact same training data and network layers as our method, allowing us to distill the effects of decomposing product closeness into compatibility and similarity links, as well as the effects of using a network for learning hierarchical embeddings.

Overall, we observe that our method significantly outperforms the baseline on both qualitative and quantitative metrics, as we demonstrate below.

4.2 VISUALIZATION OF EMBEDDING

In this section, we visualize both the compatibility and similarity embedding spaces, and compare it with the baseline. We use t-SNE plots to visualize the learned embedding spaces.

First, we observe the visualization of the similarity space, as shown in Figure 4.1. As seen in the plot, the embeddings embed similar looking items of the same category nearby with minimal overlap between items of different categories. Moreover, as can be seen in the case of shoes and tee-shirts, many shoes are easily interchangeable, which was our original definition of similarity. Therefore, the similarity embedding space is well-behaved.

We now observe a visualization of the compatibility space for the bottom-shoe type pair. Note that there will be 2 other compatibility spaces – the bottom-top space, and the top-shoe space. Moreover, the bottom-shoe space is itself created from using 2 different compatibility
Figure 4.1: **t-SNE plot of Similarity embedding.** As we can see from this plot, the similarity embeddings embeds similar looking items of the same category nearby, and there is little-to-none overlap between items of different categories.

As can be seen from the plot, while some similar looking items of the same category are embedded nearby, many items of different categories overlap at many places in the embedding. Moreover, the overlap items are also compatible – such as the overlap shown
between skinny jeans and high heels, which are compatible items. This illustrates that the compatibility network is exhibiting the desired behaviour.

We now look at the behaviour of the baseline embedding, as shown in Figure A.2. Even though the baseline is trained on the same dataset as the compatibility network, it fails to capture any notion of compatibility and closely resembles the similarity embeddings. Similar looking items of the same category are embedded nearby, with minimal overlap between items of different categories. Since the baseline is to compare against the compatibility embedding space, this behavior is less desirable. Therefore, in terms of visualization of the embedding space behavior, our method outperforms the baseline.

4.3 QUANTITATIVE PERFORMANCE EVALUATION

In the previous section, we observed how the embedding space learned by our method is better suited for both compatibility and similarity prediction tasks compared to the baseline. In this section, we quantitatively compare the performance of our method and the baseline on 3 metrics, for both tasks. The 3 metrics are defined below:

- **Average Distance Ratio (Lower is better):** This is the ratio of average distance between randomly sampled compatible and non-compatible pairs. A low ratio implies that positive pairs are much closer than negative pairs, which is indicative of a well-behaved embedding space.

- **AUC Score (Higher is better):** This is the area under a receiving operator characteristic curve. A high AUC score signals high separability of positive versus negative pairs, which results in a higher expected performance of a prediction or recommendation task on the learned embedding spaces.

- **Recall@50 (Higher is better):** This metric is how many of the 50 nearest neighbours of a query product are actually compatible with the product. In many cases, a product $x$
Figure 4.2: t-SNE plot of Compatibility embedding for the bottom-shoe type pair. In this plot, while some similar looking items of the same category are embedded nearby, there is significant overlap between items of different categories.
Figure 4.3: \textbf{t-SNE plot of the Baseline embedding}. In spite of providing the baseline with compatibility links, the baseline looks very close to the similarity embeddings, in that it embeds similar looking items of the same category nearby, with little-to-none overlap between items of different categories.

may have less than 50 compatible products. Therefore, we define R@50 to be computed as:
\[ R@50 = \frac{\text{true compatible items}}{\min(50, |C(x)|)} \] (4.1)

4.3.1 Compatibility Comparison

First, we observe the performance of our method on the compatibility learning task in Table 4.1. The embeddings used for performing this evaluation are generated from the typespecific compatibility network \((g_{\tau,\upsilon})\) for each product, as described in Section 3. As can be observed, our method significantly outperforms the baseline on all 3 metrics for each pair type.

More specifically, the low average distance ratio signifies that our networks tend to embed compatible items much closer, while increasing the distance between non-compatible pairs. The high AUC score implies that the network will perform well on tasks involving ranking products in order of their compatibility. Our method also significantly beats the baseline on the Recall@50 metric. Although the absolute values of Recall@50 are lower, we demonstrate that in practice, our method recommends high-quality items for query products. Furthermore, the training data imbalance and asymmetry (as discussed in Section 3.1) cause no significant difference on the performance of the learned embeddings.

In contrast, the baseline has trouble distinguishing between compatible and non-compatible items, in spite of using the same information for learning the embedding as our method. This is mainly because the baseline is not robust to the imbalance of data between the large number of similarity links, and the relatively fewer compatibility links. Our method uses implicit information from the hierarchical structure imposed on compatibility and similarity to offset this data imbalance, leading to better transfer. Since the baseline doesn’t utilize similarity and compatibility links cleverly, it is not robust to the data imbalance. For example, our method performs well in a situation with 2 similar looking bottoms \(x_1^b\) and \(x_2^b\), when \(C(x_2^b) = \phi\). Since both the bottoms would be embedded nearby in the similarity space, the
compatibility embedding learned using $C(x^t_1)$ would also perform well for $x^t_2$, allowing us to circumvent the missing $C(x^b_2)$ information. The baseline would not be able to perform well in such a case.

The second desirable behaviour that results in our method outperforming the baseline is again caused by the much higher numbers of similarity links compared to compatibility links. Not only does the baseline not utilize the underlying information between compatibility and similarity, but it also ends up learning an embedding that can only capture the notion of similarity. We observed this behavior in Figures 4.1 and A.2, where the t-SNE plot of the baseline looked very similar to the t-SNE plot of the similarity embedding for our method (i.e. had homogeneous clusters of products from the same type, with very little overlapping), in spite of being giving the same training data as our method.

Overall, our learning task can effectively and accurately learn the underlying compatibility notion between for each pair of items.

<table>
<thead>
<tr>
<th>Comparison Type</th>
<th>Metric</th>
<th>Our Method</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>top-bottom</td>
<td>Avg. Distance Ratio</td>
<td>0.667</td>
<td>0.742</td>
</tr>
<tr>
<td></td>
<td>R@50</td>
<td>0.065</td>
<td>0.062</td>
</tr>
<tr>
<td></td>
<td>AUC Score</td>
<td>0.830</td>
<td>0.791</td>
</tr>
<tr>
<td>top-shoe</td>
<td>Avg. Distance Ratio</td>
<td>0.633</td>
<td>0.910</td>
</tr>
<tr>
<td></td>
<td>R@50</td>
<td>0.264</td>
<td>0.169</td>
</tr>
<tr>
<td></td>
<td>AUC Score</td>
<td>0.818</td>
<td>0.718</td>
</tr>
<tr>
<td>bottom-shoe</td>
<td>Avg. Distance Ratio</td>
<td>0.733</td>
<td>0.925</td>
</tr>
<tr>
<td></td>
<td>R@50</td>
<td>0.255</td>
<td>0.155</td>
</tr>
<tr>
<td></td>
<td>AUC Score</td>
<td>0.756</td>
<td>0.697</td>
</tr>
<tr>
<td>combined</td>
<td>Avg. Distance Ratio</td>
<td>0.668</td>
<td>0.829</td>
</tr>
<tr>
<td></td>
<td>R@50</td>
<td>0.144</td>
<td>0.103</td>
</tr>
<tr>
<td></td>
<td>AUC Score</td>
<td>0.813</td>
<td>0.678</td>
</tr>
</tbody>
</table>

Table 4.1: **Experimental results for Compatibility Learning Task.** This table studies results on the compatibility prediction task, therefore only the embeddings from the type-specific compatibility spaces are used for our method. We can observe that our method significantly outperforms the baseline on all metrics, signalling that our method can learn a high-quality notion of compatibility between products. Another interesting observation is that training data imbalance (as discussed in Section 3.1) causes no significant difference on the performance of the learned embeddings.
4.3.2 Similarity Comparison

In this section, we compare the performance of our method on the similarity prediction task with the baseline. For this task, the embeddings used for our method are the embeddings generated by the similarity network $f$, before they are processed through the compatibility network. Therefore, these embeddings are 256-dimensional as opposed to the 32-dimensional embeddings produced by both the compatibility network as well as the baseline. We normalize the 256-dimensional output of the similarity network as well as the 32-dimensional output of the compatibility network to lie on a unit sphere, because of which we are able to compare their performances. Since the baseline doesn’t distinguish between similarity and compatibility links, the output of the baseline can be used as is for comparing on the similarity evaluation task as well.

The baseline network has 2 additional fully connected layers (as shown in Table 3.1), which provide the network with additional representational capacity. In spite of this advantage, our method and the baseline perform roughly similar – our method outperforms the baseline on AUC score and R@50, while performing worse than the baseline on Average distance ratio. This is consistent with our earlier observations in Figures 4.1 and A.2, wherein the embeddings learned by the baseline were dominated by similarity links, and therefore very closely resembled the embedding learned by the Similarity network. This was in spite of the extra compatibility links provided to the baseline network, along with the baseline’s extra representational capacity. These results can be observed in Table 4.2.

Overall, our method is a significantly better choice since it allows us to capture the notion of both compatibility and similarity, whereas the baseline can only capture similarity.

4.4 PRODUCT RECOMMENDATION

In this section, we use the representation learned by our method and the baseline to compare them on a product recommendation task. The analysis performed in this section
### Experimental results for the Similarity Learning Task

This table studies results on the similarity prediction task, therefore the embeddings used for our method are the output of the similarity network described in Table 3.1. The embeddings for the baseline are the same as before. We can observe that both methods perform roughly similar – our method outperforms the baseline on AUC score, while performing worse than the baseline on Average distance ratio. This is consistent with the hypothesis that the baseline learns an embedding that is equivalent to a similarity embedding, without much notion of compatibility, in spite of being provided with additional training data and learning capability.

The task is structured as follows: we would like to style a query product $x_\tau$ of type $\tau$ with a compatible product of type $\nu$ from a given set of items. For the baseline, this task is performed by generating an embedding of $x_\tau$ using the baseline network, and then performing a $k$ nearest neighbours search over all items of type $\nu$ to find potentially compatible items. In contrast, the most compatible items using our method are retrieved by the following: pass $x_\tau$ through the similarity network, and then through the relevant compatibility network (which in this case would be $g_{\tau,\nu}$). Then, in the embedding space $G_{\tau,\nu}$, search for the $k$ nearest neighbours of type $\nu$ that would have been processed through the network $g_{\tau,\nu}$. These $k$ nearest neighbours are then considered as potentially compatible items.

We start our analysis by first looking at a failure case for our method in which the Recall@50 is 0, as shown in Figure 4.4. For this case, we are given a pair of shorts (type: *bottom*), and are required to predict a set of compatible shoes from the *bottom-shoe* space. The ground truth compatible shoe is shown in Figure 4.4b. Even though our predicted set of compatible items does not contain the ground truth compatible shoe, practically the ground truth shoe looks almost identical to the predicted shoes. This signals that it is likely
that our Recall@50 metric is not indicative of the actual utility and usability of the learned compatibility embeddings.

Figure 4.4: **Failure case for our method.** As we can see, even though our predicted set of compatible items does not contain the ground truth compatible shoe, the predicted shoes are still close enough to the ground truth item so as to be actually be compatible with the query item in real life.
This phenomenon occurs because of the way by which the dataset is generated – using co-purchasing information. This information is largely guided by many concerns other than just compatibility, such as quality, brand, price, etc., which are impossible to capture using just images. Therefore the dataset does not contain important compatibility relationships that should exist. Our method is able to circumvent this issue of missing compatibility links by using a hierarchical relationship between the similarity and compatibility links, which is why it can still make high-quality predictions.

We now move on to analyzing a successful case for our method. Once again, the task is to predict a set of compatible shoes for a given pair of jeans. The results for this task can be seen in Figure 4.5. In this case, our method correctly retrieves the ground truth compatible pair of sandals at the 4th position in the predicted set of compatible shoes. However, this example also highlights how our method can retrieve a diverse set of footwear that is all compatible with the query item – sneakers and sandals. This diversity is possible because of the many-to-one mapping that is implicitly implemented in the compatibility network. This allows for many dissimilar items that might be far apart in $\mathcal{S}$ to all map to the same neighborhood in $\mathcal{G}_{r,v}$.

It is interesting to compare a successful case for our method with a successful case for the baseline, which is shown in Figure 4.6. Once again, the task is to predict the most compatible pair of shoes for a given pair of jeans. Even though the baseline correctly retrieves the ground truth compatible item in the set of predicted compatible items, the set of retrieved items itself is very homogeneous. All of the retrieved shoes are high heeled boots, leading to hardly any diversity in the set of retrieved items. This is a direct outcome of the lack of a hierarchical embedding approach.

Finally, we observe a failure case for the baseline, as shown in Figure 4.7. Once again, the task is to predict a set of compatible shoes for a given pair of shorts. From the figure, we can see that the set of predicted compatible items is very low quality when compared to the ground truth compatible item. This is because of the lack of information-sharing
Figure 4.5: **Successful case for our method.** We can see that in this case, not only does our method correctly retrieve the ground truth compatible item for the given query, but it also highlights the diversity in the set of retrieved predictions. Our method is able to predict 2 different types of footwear to go with the same pair of shorts – sneakers and sandals.

between similarity and compatibility links present in the training data. In cases when the compatibility links for an item are not well maintained, the baseline can not offset the lack
of information because it considers compatibility and similarity links independently. Some additional queries are shown in Section A.

![Query item](image1.png)  ![Ground truth compatible item](image2.png)

(a) Query item.  (b) Ground truth compatible item.

![Recommended items](image3.png)

(c) The 10 most compatible items recommended by our method.

Figure 4.6: **Successful case for baseline.** We can observe that while the baseline correctly retrieves the ground truth compatible item in the set of predicted compatible items, there is no diversity in the set of retrieved items.
Figure 4.7: **Failure case for the baseline method.** As we can see, the set of predicted compatible items is nowhere close to the ground truth compatible item. Since we’re considering compatibility and similarity links independently, if the compatibility links for an item are not well maintained, then the baseline cannot compensate by using alternative sources of information.
CHAPTER 5: CONCLUSION AND DISCUSSION

In this work, we discussed a novel approach for clothing recommendation that uses separate notions of similarity and compatibility, along with differentiating products on the basis of type. We see that making these fine-grained differentiation allowed us to learn a more powerful embedding that was able to capture the complex notion of compatibility between products.

We demonstrated in experiments that using types and performing hierarchical embedding allowed us to outperform baselines on various metrics. However, we demonstrated experiments on only 3 types of clothes. In reality, fashion has a rich taxonomy of types – ranging from outerwear to jewellery to dresses. In our approach, the number of type-specific compatibility spaces increases quadratically with the number of types. In future work, it is important to develop strategies that can control an explosion of the number of compatibility embedding spaces, without sacrificing the increase in performance obtained by using types.

Another interesting avenue for exploration is controlling for ultra-‘friendly’ product types. Many items of clothing that are considered wardrobe staples are often compatible with a large number of items, which results in a reduction of the distances between otherwise non-compatible products in the embedding space. It is important to develop methods that can learn embeddings that are robust to variability in product ‘friendliness’.
CHAPTER 6: REFERENCES


Figure A.1: t-SNE plot of the Compatibility embedding for the top-bottom type pair. While some similar looking items of the same category are embedded nearby, there is significant overlap between items of different categories. Moreover, we can also observe some mis-labelling of categories, that cause shoes to appear in this embedding.
Figure A.2: t-SNE plot of the Compatibility embedding for the top-shoe type pair. While some similar looking items of the same category are embedded nearby, there is significant overlap between tops and shoes.
(a) Query item.  
(b) Ground truth compatible item.

(c) The 10 most compatible items recommended by our method.

Figure A.3: **Failure case for our method on bottom-top prediction.** As we can see, even though our predicted set of compatible items does not contain the ground truth compatible bottoms, the predicted bottoms are still compatible with the query item in real life.
Figure A.4: **Success case for our method on top-shoe prediction.** As we can see, our method very accurately predicts not just the ground truth item, but also a diverse set of compatible items.