Improved Recommendations Via Linked Latent Spaces

Marco Franco, Anirudh Kamath {FRANCO.MA, KAMATH.AN}@NORTHEASTERN.EDU

Abstract/Introduction

Recommendation algorithms exclusively utilize either image data or hard-coded semantic features to provide relevant solutions via pairwise similarity scores between elements [Wu+21][Jen+17]. Using the "Fashion Product Images" dataset containing corresponding images and semantic features, we construct a composition of models that produce low dimensional representations and converts between the two components. Randomly sampled visualizations of the model demonstrate relevant recommendations that are distinct from those proposed using either component individually. Lastly, by encoding semantic features directly into the latent space, we demonstrate controlled and meaningful latent space interpolation.

1. Dataset

The data used was a 10,000-size subset of 44,000 images and associated metadata from the "Fashion Product Images" Kaggle dataset posted by Param Aggarwal, which was composed by scraping e-commerce websites (examples in Fig. 1). Data ranged from shirts to shoes to accessories. Image data is denoted F_{images}, while corresponding fashion/clothing metadata is denoted F_{graph}.

Figure 1: Sample from F_{images}, the image component of the dataset

F_{graph} contains 8 categorical variables for each image, specifically Gender, Master Category (apparel, accessories, etc.), Subcategory (shoes, top wear, etc.), Article Type (T-shirt, casual shoes, etc), Base Color, Season, Year of Production, and Usage (casual, sport, etc.). The distribution was uneven within each category, i.e. season is heavily skewed to favor summer.

The general project was an exercise in constructing and translating between two discrete latent spaces (image and node). Broadly, this project is characterized as a recommendation system, since the composition of the aforementioned latent spaces is used to provide recommendations of fashion items similar to a given fashion image based on semantic data not found in the image itself.

2. Data Representation and Processing

2.1 Image Processing

The original images from the "Fashion Product Images", were downsampled using TensorFlow's Bilinear Interpolation from $(2400 \times 1600 \times 3)$, to $(256 \times 256 \times 3)$.

2.2 Graph Metadata Representation

We first define the metadata, F_{graph} as a graph, such that $F_{graph} = (V, E)$ where $V = \{\text{unique image identifying index}\}$ and $E = \{\text{shared categorical attribute}\}$. Therefore, two vertices $v_1, v_2 \in V$ will share an edge $e_a \in E \iff$ both v_1 and v_2 have attribute a. For example image 1 and image 2 are both apparel (Master Category), therefore these images share an edge of type "apparel".

Naturally, F_{graph} is considered a Knowledge Graph [Zha+19], since the categorical variables form relationships between the individual "Fashion Product Images". However, computational and memory deficits arose due to the $O(mn^2)$ runtime of making each node connected with every other node it shared an attribute with for m categorical attributes of F_{graph}.

We thus defined F'_{graph} such that $V' = V \bigcup E$ and the edges E' exist between instances and attributes. Following the previous example, image 1, image 2 and type "apparel" are all vertices, and image 1 and image 2 are each individually connected to the apparel vertex. Therefore, the attribute nodes serve purely as intermediate path nodes between two instance nodes, making our graph a little less precise, but our runtime linear at $O(mn)$. This is something we can improve upon given more funding and time.

3. Model Architecture

3.1 Image Model

We first created create image embeddings via Variational/Convolutional Autoencoders, which encode an image as an n dimensional vector, and then decodes said vector into the original image. However, due to either failed convergence after hyperparameter searches (varying convolutional layers and embedding size: Fig 2) in the case of the VAE, or long training times in the case of the CAE, we scratched both. Our final model was constructed by appending a single 2D convolutional layer, a 2D Max Pooling layer, and a Global Pooling layer to the TensorFlow ResNet50 model. ResNet is a commonly used convolutional neural network, and by using a heavily pretrained general use-case CNN, we were able to generate very accurate embeddings without any additional training. The final output of the model is an image encoded as a 128 dimensional vector. Importantly, this model is only capable of producing a latent encoding for an image; there is no decoding component.

3.2 Graph Metadata Model

To generate the node embedding for the metadata component, F'_{graph} we utilized the node2vec algorithm from the Python Node2Vec library, written by the same authors as the paper. We denote this network as G_{n2v}. Specifically, we constructed the node2vec al-

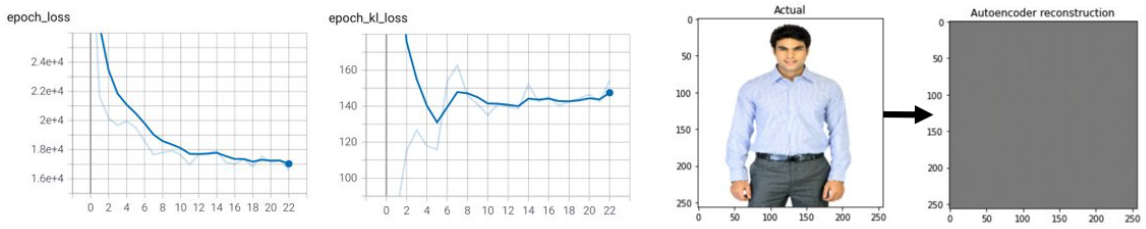

Figure 2: Performance statistics for VAE with lowest epoch loss. Both the Total Epoch Loss (LEFT) and the KL-loss (MIDDLE) vs epochs graphs for the VAE converge to local minima. Reconstruction of dataset image produced by the VAE model that had converged for multiple epochs (RIGHT).

gorithm to produce node encoding of dimension 32, calculated through 200 walks of length 30 for each node in the dataset.

3.3 Linking Model

The Linking/Conversion model, denoted as L_{I2G}, was constructed by training two dense fully connected layers, the first having a nonlinear SELU activation function to convert the 128 dimensional image embedding to a 96-length embedding, then one standard dense layer to compress this 96-length embedding to a predicted 32-dimensional node embedding. This network was trained to minimize the mean squared error between the node embedding and the converted image embedding using the rmsprop optimizer.

3.4 Composition of Models

We define our Recommendation system, denoted as *Rec*, by composing the three aforementioned models to produce the node embedding corresponding to a given image, as seen in Figure 3. Given an image, *Rec* first utilizes I_{ResNet} to produce a 128 dimensional embedding of the image. This embedding is then converted, via L_{I2G} into the node latent space to the corresponding 32 dimensional node embedding. We then calculate the pair-wise cosine similarity between this converted embedding, with the embeddings constructed by the G_{n2v}. We then sort these ranked similarities, and by using the indices, we return the corresponding images.

4. Results

4.1 Latent Space Visualization

In order to properly motivate the architecture and feasibility of the conversion/linking model L_{I2G}, we first employed 2-dimensional T-SNE visualization. The graphs in Figure 4, illustrate the result of T-SNE visualization, calculated over 300 iterations, with 2 components, and a perplexity score of 40. The graphs were constructed such that the horizontal and vertical axis represent the two principle components formed for each dataset respectively.

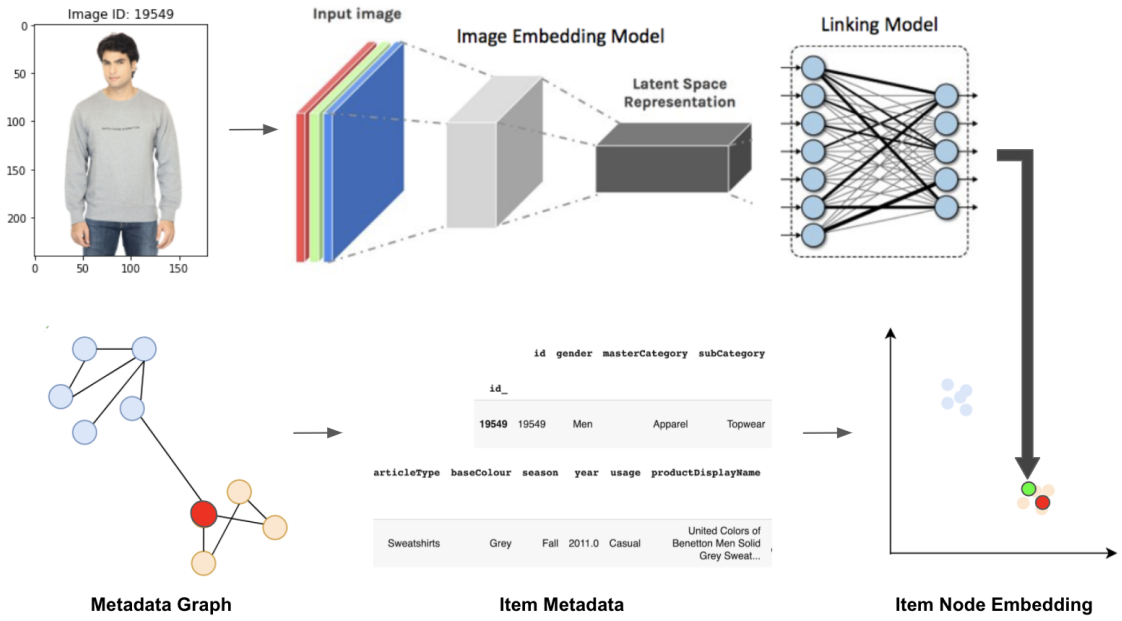

Figure 3: The *Rec* model first constructs an image embedding for a given image. The image embedding is then converted into the graph latent space via the linking model (green dot) with the aim of being as close as possible to the ground truth (red dot).

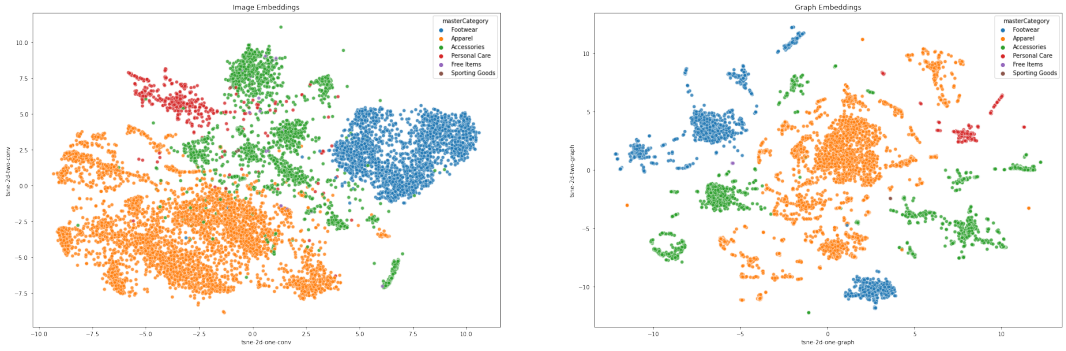

Figure 4: The T-SNE Visualization of the I_{ResNet} (LEFT) and G_{n2v} (RIGHT) embedding spaces. For this visualization, we only graph nodes according to their *Master Category* attribute.

We note the structural similarity between the T-SNE graphs of the two embeddings. In both graphs, the footwear attribute (illustrated in blue) of the Master Category variable is not only clustered, but also disjoint from any other cluster. Moreover, the apparel attribute and accessories attributes (orange and green respectively) share the same relation across the two T-SNE graphs, specifically that the accessories are distributed around the apparel cluster.

While the structural similarities are interesting, we posit this stems from the inherent similarity of the data contained in both the images and the metadata. We believe that both components of the dataset share the same set of retrievable variables. While these results are not shocking, it is interesting that two drastically different down-sampling/embedding models were able to produce structurally similar latent spaces.

4.2 Latent Space Interpolation

Figure 5: Constructing a new graph space vector by first converting an unseen image to an image embedding, then translating the image embedding into the graph space, subtracting the "topwear" node embedding, and adding "blue" and "shoes". The RHS indicates the 6 catalog items with the highest similarity scores (cosine) to the calculated vector.

By including the the attributes as vertices in our graph, our node embedding algorithms will now map attributes to the latent space alongside actual image identifiers. To investigate the role of these embedded attributes in the latent space, we perform interpolation in the graph data latent space using the attribute nodes as a basis. Specifically, we calculate the latent representation of each attribute, such that attributes like "blue" are now a 32-dimensional vector, and perform vector subtraction and addition using these attribute encodings.

As illustrated in Figure 5, we were able to demonstrate that the encoding of the color vectors

into the graph latent space is "meaningful" and preserves structure in the sense that the model infers attributes such as "athletic" and "men's" from an image completely outside of the dataset (image of author Anirudh Kamath) and preserve these attributes throughout the interpolation in the shoes (i.e. the shoes aren't women's or formal shoes).

4.3 Recommendations

Due to the nature of recommendation algorithms, there is not a single clear summary metric we can apply to evaluate the success of the model. Rather, we provide two sets of images to support both the strength and uniqueness of recommendations. First, with respect to the accuracy/strength of recommendations, Appendix: Recommended Images provides multiple randomly sampled examples of the result of the *Rec* algorithm, where each figure illustrates the 6 pre-existing elements with the highest cosine similarity score to the embedded and translated image. To highlight the uniqueness of the model predictions, we randomly sampled elements from the dataset and then present the 6 elements with the highest cosine similarity from either the image space, graph space or produced by the *Rec* model (left to right respectively).

Interestingly we note for several of the examples, there are recommendations that appear visually different from the original image, but share a multitude of features. For example, the recommendation of a pink bag for the lady with the pink dress is less visually similar, but shares many common attributes, indicating that both components of the dataset are being utilized. Moreover, the comparison of suggestions indicates that not only is the *Rec* model producing accurate recommendations, but the model is also proposing unique items that are not included in the recommendations of the individual components.

5. Conclusion

The composition of the individual models into the *Rec* model performed well as illustrated by the randomly sampled visualizations in the Appendix. While the exact strength of the model is difficult to approximate, the shared structure between the distinct latent spaces, the high cosine similarity between the translated and ground truth vectors, and the interpretability of the attribute embedding vector arithmetic indicates a successful approximation of the underlying structure of the Fashion Product Images dataset. Future improvements could include replacing the ResNet50 down-sampling model with fancier autoencoders (VAEs, β-VAEs, etc.) and broadening the subset of the dataset used to train (we currently use 25% of the available data). Future areas of exploration include analyzing and quantifying the extent of the interpretability of the latent space interpolation we have demonstrated.

References

[Jen+17] Kartik Chandra Jena et al. "Principles, techniques and evaluation of recommendation systems". In: *2017 International Conference on Inventive Systems and Control (ICISC)* (2017). DOI: 10.1109/icisc.2017.8068649.

[Zha+19] Shuai Zhang et al. “Deep Learning Based Recommender System”. In: *ACM Computing Surveys* 52.1 (2019), pp. 1–38. DOI: 10.1145/3285029.

[Wu+21] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/tnnls.2020.2978386.

Appendix of Results

Figure 6: We take the image on the left, compute an image embedding, translate the image embedding to the graph space, and find the items in the dataset with node embeddings most similar to the translated vector

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
id_										
19549	19549	Men	Apparel	Topwear	Sweatshirts	Grey	Fall	2011.0	Casual	United Colors of Benetton Men Solid Grey Sweat...

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
id_										
41599	41599	Men	Apparel	Topwear	Shirts	Blue	Summer	2012.0	Casual	Basics Men Blue Shirt
2011	2011	Men	Apparel	Topwear	Tshirts	Blue	Fall	2010.0	Casual	ADIDAS Men Classic Blue Striped T-shirt
27555	27555	Men	Apparel	Topwear	Shirts	Black	Summer	2012.0	Casual	Scullers Men Black Shirt
19696	19696	Men	Apparel	Topwear	Shirts	Grey	Fall	2011.0	Casual	United Colors of Benetton Men Solid Grey Shirt
37726	37726	Men	Apparel	Topwear	Shirts	Blue	Summer	2012.0	Casual	John Players Men Soft Check Shirt
10805	10805	Men	Apparel	Topwear	Shirts	Black	Fall	2011.0	Casual	Wrangler Men Rider Solid Black Shirts

Figure 7: Item attributes from the chosen item in Figure 6. You can see that attributes are preserved through the translation, such as "Men", "Apparel", "Topwear", "Casual", and "Shirt"

Figure 8: Similar to Figure 6, but this time after translating, we add the node vectors for "Shoes" and "Green" and subtract the node vector for "Topwear". This was an advantage of directly embedding our attributes as nodes.

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
id_										
19549	19549	Men	Apparel	Topwear	Sweatshirts	Grey	Fall	2011.0	Casual	United Colors of Benetton Men Solid Grey Sweat...

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	productDisplayName
id_										
20891	20891	Men	Footwear	Shoes	Casual Shoes	Green	Fall	2011.0	Casual	Basics Men Green Casual Shoes
24182	24182	Men	Footwear	Shoes	Casual Shoes	Green	Summer	2012.0	Casual	Converse Men Green Casual Shoes
21472	21472	Women	Footwear	Shoes	Casual Shoes	Green	Winter	2012.0	Casual	Roxy Women Cute Lime Casual Shoes
13191	13191	Men	Footwear	Shoes	Casual Shoes	Green	Fall	2011.0	Casual	Puma Men Eco ortholite Green Casual Shoes
4126	4126	Men	Footwear	Shoes	Casual Shoes	Black	Summer	2011.0	Casual	Fila Men's Fuel HI Black Shoe
15316	15316	Men	Footwear	Shoes	Casual Shoes	Green	Fall	2011.0	Casual	ADIDAS Men Vigor Green Casual Shoes

Figure 9: Item attributes from the chosen item in Figure 8. You can see that the vector arithmetic made a difference, suggesting with attributes including "Green" and "Shoe" and not any topwear, while also preserving attributes like "Men" and "Casual"

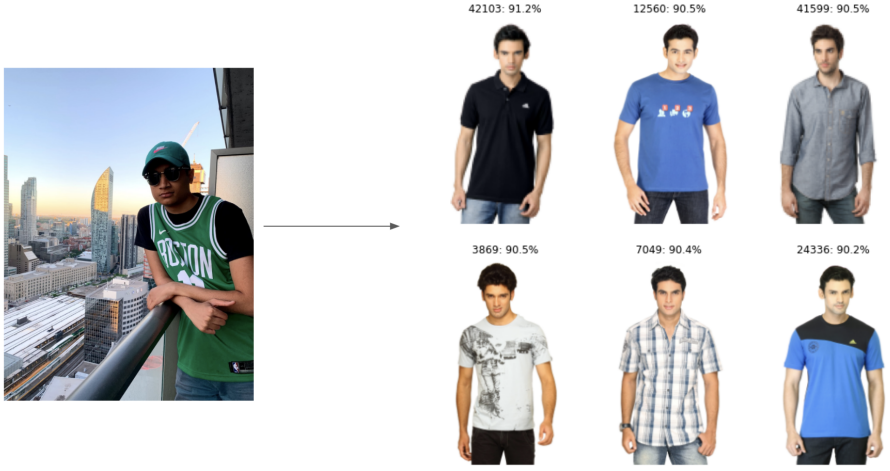

Figure 10: Taking an image completely out of the dataset and seeing how the model performs. This is the end goal of the project - be able to infer attributes from an image without any additional data. You can see it performs rather well, suggesting casual men's topwear.

Figure 11: When we perform vector arithmetic on the translated vector from Figure 10, we see that attributes such as "men" and "athletic" are preserved, and only the requested attributes like "topwear", "shoes", and "blue" are modified

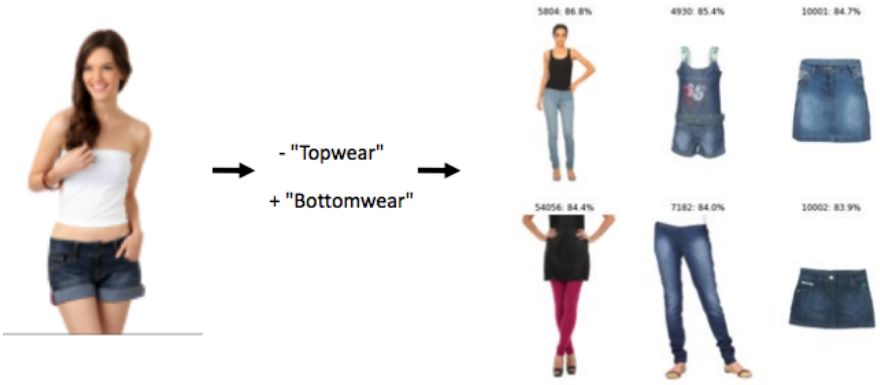

Figure 12: So far all the examples have been men's, but it works on women's clothing too, as well as on other forms not including just topwear and shoes.

Figure 13: Another more complex extension of the previous figure.