Scalable and Explainable Outfit Generation

Alexander Lorbert, David Neiman, Arik Poznanski, Eduard Oks, Larry Davis
Amazon
{lorbert, neimand, arikp, oksed, lrrydav}@amazon.com

Abstract

We present an end-to-end system for learning outfit recommendations. The core problem we address is how a customer can receive clothing/accessory recommendations based on a current outfit and what type of item the customer wishes to add to the outfit. Using a repository of coherent and stylish outfits, we leverage self-attention to learn a mapping from the current outfit and the customer-requested category to a visual descriptor output. This output is then fed into nearest-neighbor-based visual search, which, during training, is learned via triplet loss and mini-batch retrievals. At inference time, we use a beam search with a desired outfit composition to generate outfits at scale. Moreover, the attention networks provide a diagnostic look into the recommendation process, serving as a fashion-based sanity check.

1. Introduction

The advances in both machine learning and deep learning have extended well beyond the academic and scientific communities, and have found their way into the fashion industry in recent years. The AI challenges faced by those in the fashion industry are quite complex due to the subjective, individual, seasonal, cultural, and dynamic nature of fashion and style. In light of the many challenges relating to fashion, we focus on a particular problem of outfit recommendation.

The customer-centric problem we address is: "Given my current outfit, what <*category*> do you recommend?" For example, a customer may want to be recommended a pair of shoes to go with the blouse and pants she is already wearing. By providing recommendations in this format, customers can plan or "grow" an outfit from one or more initial clothing items. In this way, items can be added to (or removed from) an outfit, allowing customers to go from an outfit's initial draft to its final draft.

We aim to automate this process of building an outfit from a single seed item that is both scalable and explainble. Typically, outfitting solutions involve a composition template and/or use fill-in-the-blank (FITB) approaches for refining and evaluating. Recurrent units are also employed, which may lead to different recommendations for the same partial outfit due to the ordering of the items. We argue that outfit compositions should be dynamic and that using a FITB approach of selecting a single item from a small set of, say, 5 items is not scalable. Furthermore, outfits are inherently sets and providing recommendations for a partial outfit should not depend on item ordering. Additionally, we would prefer an algorithm that can provide some kind of signal that may provide insight as to how the recommendations are being generated.

We address outfit generation in a two step process. First, we train a neural network that learns a visual descriptor and produces an outfit vector for nearest neighbor visual search (using inner products). Each input into this network is a partial outfit and requested, next category. We use attention [6] for combining this variably-sized input in an approach similar to a masked language model [1], yielding an order-invariant function. Next, during inference, for a given seed item we propose an outfit composition and then employ a beam search to conduct the outfit building. Our use of inner-products for nearest neighbor search allows us to scale and recommend items from large image repositories. During training, we mimic the retrieval process via minibatch retrievals with a triplet loss [3, 5]. By internally leveraging attention, we achieve the desirable properties of order-invariance and explainability.

2. Related Work

In [2], Han *et al*. leveraged data from Polyvore to address outfits. The authors used Polyvore's ordering to train a bidirectional LSTM for fashion compatibility. They also incorporate the text associated with each item by adding a contrastive loss that compares the visual with the textual. Lin *et al*. [4] take a similar approach to our work, but only provide single recommendations and not entire outfits. They also make use of attention, but only at the paired category level thereby discarding, for example, valuable visual information (*e.g*., some colors/patterns may effectuate more attention).

3. Outfit Recommendation

Let $\mathcal{R} = \{(I_j, c_j)\}_{j=1}^{N}$ denote our repository of N image-category pairs. The j-th item in the repository is associated with image I_j and label $c_j \in \{1, 2, \ldots, K\}$, where K is the number of different clothing/accessory categories. With training in mind, we shall denote an outfit as an *ordered* set of unique indices, where each element is an index into our repository. Two outfits are equivalent if one is a permutation of the other. Thus, an outfit of r items has $r!$ different, equivalent representations.

Let f denote a visual feature extractor. The function f receives an image and outputs a feature vector of length M. Here, f can be fixed, learned, or fine tuned. In practice, we use a convolutional neural network. Our visual search mechanism takes in a vector of length M and finds the repository images closest to it by computing the similarity/distance between the input vector and $\{f(I_j)\}_{j=1}^{N}$. If the search is restricted to a particular category, c, then we consider the subset $\{f(I_j)\}_{c_j=c}$.

For training purposes, we are given a set of outfits, denoted $\mathcal{Z}$. Outfit $z \in \mathcal{Z}$ is a vector of integers: it has $|z|$ items, where the j-th element, z^j, is an index in to the repository $\mathcal{R}$. Here, z has a particular order, but in training we will scramble the order repeatedly to ensure the model is order-robust.

We can formulate our problem as follows: for outfit z, given items $(I_{z^1}, c_{z^1}), \ldots, (I_{z^t}, c_{z^t})$, produce a visual feature vector for category $c_{z^{t+1}}$, denoted x^{t+1}, so that $f(I_{z^{t+1}}) \approx x^{t+1}$. If training is successful, inputting x^{t+1} into the visual search will result in a top retrieval spot for $I_{z^{t+1}}$. For a given [scrambled] outfit we consider the $|z|-1$ recommendations ($t = 1, 2, \ldots, |z| - 1$, *i.e.*, $1 \to 2$, $1\&2 \to 3$, $1\&2\&3 \to 4$, *etc.*).

We consider the architecture shown in Figure 1 for a partial outfit of z_i. Each supplied image passes through the function f to generate an unnormalized visual descriptor. Each supplied category is mapped to an embedding and concatenated with the associated visual descriptor, which is then passed through the function g to give $u_i^j = g(f(I_{z_i^j}); \text{embed}(c_{z_i^j}))$. For the requested next category, the image is masked and a learned "missing" image descriptor replaces the associated value of f, yielding a value for u that is strictly dependent on its category. Nevertheless, we will abuse notation and simply use u_i^j, which is a latent representation of the category and *available* visual information.

The latent u vectors are now passed through a self attention block, where they are mapped to a context-aware version of themselves, denoted $\bar{u}$. For recommendation, we concern ourselves with the output vector corresponding to the visually-masked item. Thus, using a single attention head, for the j-th item we obtain $\bar{u}_i^j = \sum_{r=1}^{t+1} \alpha_r v_r$ for $\alpha_{1:t+1} = \text{softmax}\left(\langle q_1, k_1\rangle/\sqrt{d_k}, \ldots, \langle q_{t+1}, k_{t+1}\rangle/\sqrt{d_k}\right)$, where q_r, k_r and v_r are the respective query, key and value associated with u_i^r, d_k is the key dimension, and the softmax ensures that the α weights sum to 1. The context-aware $\bar{u}_i^{t+1}$ is then passed through the function h to give the output, x_i^{t+1}. Reordering items 1 through t does not change the value of the output, making this a set operation.

Our training objective is a triplet loss that mimics retrieval at the batch level.We encourage the output visual descriptor, x_i^{t+1}, to be closer to the actual visual descriptor, $f(I_{z_i^{t+1}})$, when compared with all other visual descriptors belonging to the same category. This truncated retrieval is done at the batch level, so if we have a batch of B outfits, $\mathcal{B} = \{z_1, \ldots, z_B\}$, then we seek to minimize the triplet loss

$$\sum_{i=1}^{B} \sum_{j=2}^{|z_i|} \sum_{\tilde{i} \neq i} \sum_{\tilde{j}=1}^{|z_i|} \Bigg(\mathbb{1}[c_{z_i^j} = c_{z_{\tilde{i}}^{\tilde{j}}}] \times \left[D\left(x_i^j, f(I_{z_i^j})\right) + \gamma - D\left(x_i^j, f(I_{z_{\tilde{i}}^{\tilde{j}}})\right) \right]_+ \Bigg) \quad , \quad (1)$$

where D is a distance function, γ is a margin parameter, and $[\cdot]_+$ is the positive part function. The first two summations traverse all image descriptors in all of the outfits from the second item onward. The subsequent two summations extract all image descriptors from the other outfits in the batch, and the indicator function ($\mathbb{1}$) ensures we only compare items of the same category. For speed and scalability, we employ a Euclidean distance function of the ℓ_2-normalized vectors. This allows us to use an inner product (cosine similarity) for fast nearest neighbor search.

Missing category prediction In addition to next-item recommendations, we also need to know what kind of items we could potentially add to the outfit. This is especially relevant when the partial outfit is a single item. It can also serve as guidance in an interactive setting, when a customer assembles an outfit incrementally, adding one item at a time.

As mentioned above, for [scrambled] outfit z_i we generate the $|z_i|-1$ recommendation outputs. For each recommendation we consider the first t items and focus on item $t + 1$. Thus, for t items present in the current, partial outfit, there are $|z_i|-t$ absent items. For missing category prediction we use a bank of K binary classifiers–one for each category–to predict the remaining $|z_i|-t$ categories. Summing K binary cross-entropy loss functions, we form a multi-task objective by combining it with the triplet loss of (1).

We again employ the latent u vectors for missing category prediction. These vectors serve as input into a separate attention module, which outputs a single vector. Next, we feed this vector into a multilayer perceptron with output dimension equal to K for the missing category classifications. These values will be used as preliminary step for outfit generation discussed below.

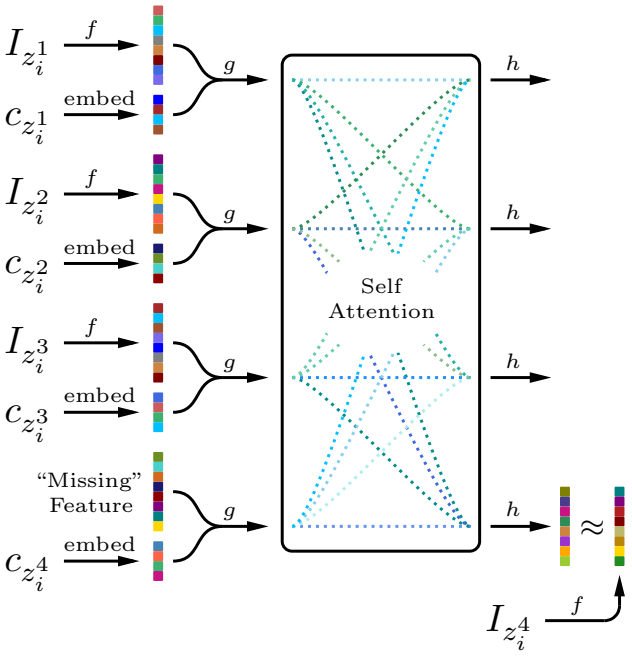

Figure 1. Core network architecure. For a partial outfit of length 4, we mask out the visual component of the fourth item and learn to estimate it via the first 3 items and the fourth category.

4. Outfit Generation

Generating an outfit for a given seed item is accomplished in a two-step process. First, we propose an outfit composition, *i.e.*, a set of categories. A single item can lead to several potential compositions and, in practice, we can choose one or more of these. In the next step, we build the outfit incrementally via a beam search according to the composition order. The composition order is significant - ancillary items such as jewelry and accessories will typically appear later on in the composition while central items such as tops, bottms and shoes are featured in the beginning. Notwithstanding, our approach does not require ordering a composition in a particular way.

Composition generation For a given seed item number of sensible compositions is relatively limited. We adopt a simple approach of relying on the compositions present in the training data. First, we collect the frequent compositions containing the seed item category, recording their curation order. This provides a set of binary vectors, where each binary vector is a vector of length K with item j equal to 1 if category j is present in the outfit.

We now use the output of the missing category classifiers after inputting the seed item, which is also a vector of length K. Calculating the inner product of this vector with the above binary vectors produces a score with which to rank the compositions. This provides a set of candidate compositions for our beam search.

Beam search Provided with a seed item and composition we execute a beam search to assemble the outfit. Repeated calls to the model requesting the next category provides lists of items with their cosine similarities. Summing these similarities provides an overall score for the outfit. For a moderately-sized beam width (20-200), we are able to generate hundreds of thousands of outfits per day on a single host. To provide a diverse set of outfits for the same seed, the beam search can be modified to generate multiple outfits in parallel with the constraint that no item is duplicated across outfits (except for the seed item).

5. Results

We used 240K women's outfits and 265K men's outfits for training (curated in-house). Typical outfit lengths ranged from 4 to 8 items, with an average of 5.8. For each network, f was a fixed, Imagenet-pretrained ResNet50 with an additional trainable fully connected layer with output dimension 256. The learned category embedded had dimension 8, and g produced latent vectors of 256 as well. We used 4 attention heads for both recommendation and missing category prediction. Our batch size was 256, and we trained for 200 epochs.

Sample Outfits We present some examples of generated outfits using our approach in Figure 3. Our trained models–one for women, one for men–generated outfits using the repositories featured in Table 1. Some items in the repositories were also present in the training data, as listed in the table. We observed that training items appear in outfits twice as often as test items.

Explainability The attention block embedded in the network may provide insight into the recommendation procedure. Intuitively, the visual content and category of some items should matter more than others when building an outfit. We demonstrate this via 2 fashion "rules" using 2 test outfits.

The first "rule" concerns matching jewelry, *e.g.*, a gold necklace should be worn with a gold bracelet and earrings. In Figure 2 (left), we see that the fourth attention head indeed emphasizes the earrings and bracelet when recommending a necklace, and the top search results do indeed yield necklaces that match the present jewelry. The second "rule" involves matching a belt to the shoes one is wearing. In Figure 2 (right), we see that attention heads 3 and 4 emphasize the shoes for recommending a belt, and the top search results yield brown belts, matching the brown shoes.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

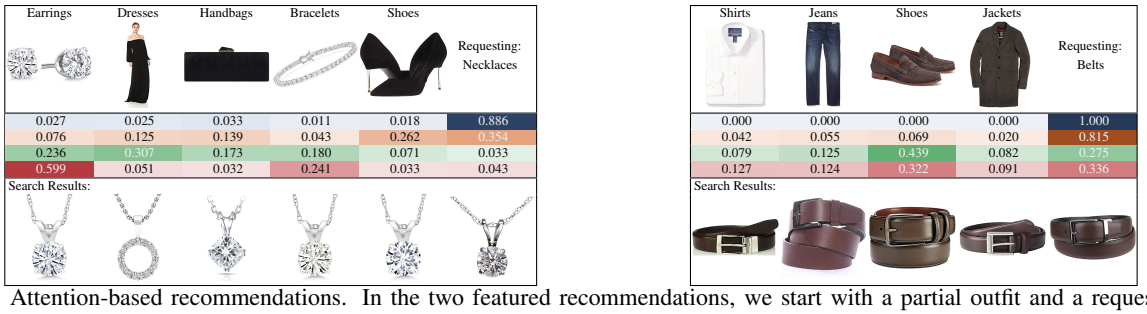

Figure 2. Attention-based recommendations. In the two featured recommendations, we start with a partial outfit and a requested next category. Beneath each item/category are the attention weights for the 4 attention heads (each row adds to 1). The top search results are shown in the bottom row.

	Category	Train	Test	Total	%-Train
Women	Accessories	12,634	10,341	22,975	55.0%
	Active	9,524	18,057	27,581	34.5%
	Coats, Jackets & Vests	6,307	19,393	25,700	24.5%
	Dresses	19,513	72,795	92,308	21.1%
	Fashion Hoodies	1,148	7,832	8,980	12.8%
	Handbags	21,669	34,519	56,188	38.6%
	Jeans	9,551	12,503	22,054	43.3%
	Jewelry	26,582	46,970	73,552	36.1%
	Jumpsuits	4,236	4,185	8,421	50.3%
	Leggings	720	4,397	5,117	14.1%
	Pants	6,344	15,025	21,369	29.7%
	Shoes	49,477	176,273	225,750	21.9%
	Shorts	2,017	5,292	7,309	27.6%
	Skirts	4,797	5,359	10,156	47.2%
	Suiting & Blazers	3,855	3,417	7,272	53.0%
	Sweaters	4,236	22,566	26,802	15.8%
	Swimsuits & Cover Ups	6,101	36,368	42,469	14.4%
	Tops, Tees & Blouses	23,820	77,319	101,139	23.6%
	Watches	7,485	8,456	15,941	47.0%
	Total	220,016	581,067	801,083	27.5%
Men	Accessories	22,956	39,904	62,860	36.5%
	Active	8,638	12,519	21,157	40.8%
	Fashion Hoodies	1,370	6,036	7,406	18.5%
	Jackets & Coats	9,194	12,557	21,751	42.3%
	Jeans	13,067	3,628	16,695	78.3%
	Jewelry	985	2,321	3,306	29.8%
	Pants	7,794	4,085	11,879	65.6%
	Shirts	30,892	57,107	87,999	35.1%
	Shoes	34,337	66,226	100,563	34.1%
	Shorts	4,583	4,195	8,778	52.2%
	Suits & Sport Coats	4,549	1,306	5,855	77.7%
	Sweaters	4,125	4,502	8,627	47.8%
	Swim	2,218	5,804	8,022	27.6%
	Watches	14,087	12,389	26,476	53.2%
	Total	158,795	232,579	391,374	40.6%

Table 1. Item repository. Outfit items are selected from 19 women's categories and 14 men's top-level categories. For the women's [men's] repository, 72.5% [59.4%] of the items did not appear in a training outfit.

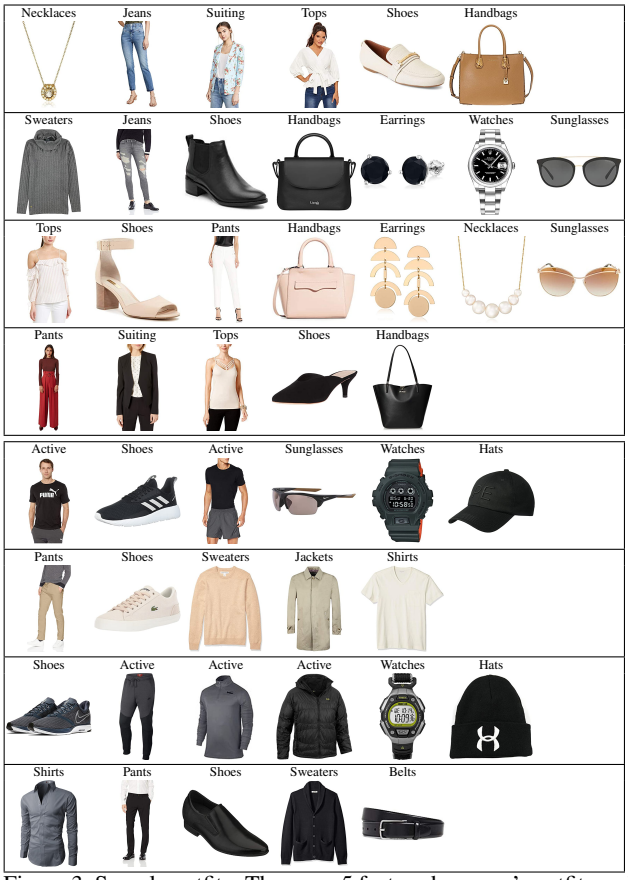

Figure 3. Sample outfits. There are 5 featured women's outfits and 5 featured mens's outfits. Each outfit was assembled according to the seed item (left-most) and ordered composition presented. The seed items were not part of the training set.

[2] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1078–1086, 2017.

[3] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[4] Yen-Liang Lin, Son Tran, and Larry S. Davis. Fashion outfit complementary item retrieval. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020.

[5] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012. IEEE, 2016.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.