Text2Outfit: Controllable Outfit Generation with Multimodal Language Models

Yuanhao Zhai[1]* Yen-Liang Lin[2] Minxu Peng[2] Larry S. Davis[2]
Ashwin Chandramouli[2] Junsong Yuan[1] David Doermann[1]
[1]State University of New York at Buffalo [2]Amazon

Abstract

Existing outfit recommendation frameworks focus on outfit compatibility prediction and complementary item retrieval. We present a text-driven outfit generation framework, Text2Outfit, which generates outfits controlled by text prompts. Our framework supports two forms of outfit recommendation: 1) Text-to-outfit generation, where the prompt includes the specification for each outfit item (e.g., product features), and the model retrieves items that match the prompt and are stylistically compatible. 2) Seed-to-outfit generation, where the prompt includes the specification for a seed item, and the model both predicts which product types the outfit should include (referred to as composition generation) and retrieves the remaining items to build an outfit. We develop a large language model (LLM) framework that learns the cross-modal mapping between text and image set, and predicts a set of embeddings and compositions to retrieve outfit items. We devise an attention masking mechanism in LLM to handle the alignment between text descriptions and image tokens from different categories. We conduct experiments on the Polyvore dataset and evaluate the quality of the generated outfits from two perspectives: 1) feature matching for outfit items, and 2) outfit visual compatibility. The results demonstrate that our approach significantly outperforms the baseline methods in text to outfit generation.

1. Introduction

Fashion outfit recommendation system constructs outfits from items that go well together (e.g., an outfit that includes a top, a bottom, shoes, bags, etc). It is an important recommendation problem for online retailers, as customers frequently shop for items that can be worn together. Existing outfit recommendation methods mainly focus on compatibility prediction (determining whether a set of fashion items in an outfit go well together) [7, 9, 21, 29, 30] or outfit complementary item retrieval (completing a partial outfit by finding a compatible item) [20, 25]. The outfit items are predicted by a pre-trained model and are not controlled by the text prompt. For example, generating outfits with desired attributes, e.g., wide-leg sweatpants.

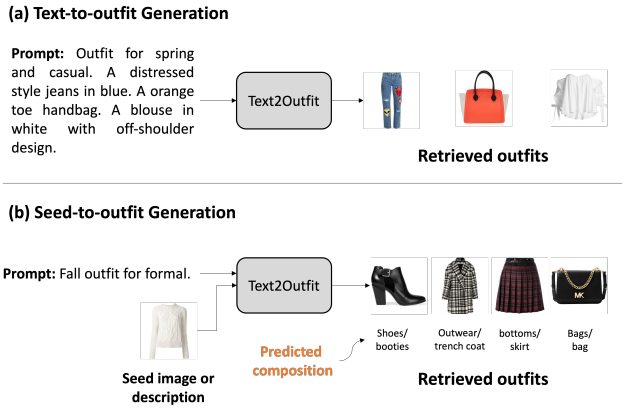

Figure 1. A text to outfit generation framework that generates outfits driven by the text prompt. Our framework supports two forms of outfit recommendation. For text-to-outfit generation, the prompt includes outfit-level and item-level specifications. The model retrieves outfit items that match the texts in the prompt and are stylistically compatible. For seed-to-outfit generation, the prompt includes the specification for a seed item (a seed image or text description). The model predicts the composition (outfit categories) and retrieves the items to build outfits.

Recent work ([14, 15]) for text to image retrieval leverages a pre-trained LLM model to learn image embeddings. They perform better retrieval than non-LLM based approaches (e.g., CLIP [24]) especially for longer text descriptions. However, these approaches focus on retrieving single images so they are not designed for retrieving image sets like outfits.

We present an outfit generation framework (Text2Outfit) that generates outfits driven by a text prompt (cf. Figure 1). Our framework learns the cross-modal mapping (text to image set) and predict the embeddings to retrieve outfits

*This work was done during an internship at Amazon.

that match the product features specified in the text prompt and are visually compatible. We instantiate our framework in two outfit recommendation scenarios using text prompts. In the first scenario (text-to-outfit generation), users specify the text prompt that describes both outfit-level features (e.g., an outfit for spring and causal) and individual items (e.g., a blouse in white with an off-shoulder design). Our model retrieves the complementary outfit items that match the prompt to build outfits. In the second scenario (seed-to-outfit generation), users specify the requirements for a few items (e.g., a seed item), using either images or texts, e.g., a white knitted sweater, and the model both completes the composition and retrieves complementary items to complete the outfit. We consider the case of just one seed item, but the framework can be extended to include multiple seed items. To build complete outfits, we predict the outfit composition (outfit product categories) for the remaining items and retrieve items in the predicted categories to build outfits.

We use learnable image tokens to generate embeddings for image retrieval similar to ([14, 15]). In contrast to prior work, we use a set of image tokens to retrieve outfits (image set) beyond a single image. Specifically, we incorporate a set of image tokens for different product categories (e.g., top, bottom, shoes). For each image token, we use the MLP layers on the top of the hidden state vector from the LLM to generate the embeddings. We train the framework in an end-to-end manner (fine-tune both LLM model and MLP layers) using LoRA [10] with a loss function that considers both product feature matching and outfit compatibility.

The prompt includes the texts describing multiple items, which introduces the challenge of aligning the outfit texts and image tokens. It is important to avoid the case - where the image token from one category wrongly attends to the text descriptions from other categories. We present a masking mechanism in the self-attention layers to facilitate the needed attention learning (cf. Section 3.1 and Figure 3) .

For seed-to-outfit generation, since we only have one seed item as input, we must select the composition (outfit categories) for the remaining items. For example, for specific occasions/seasons, some categories should not be included. So, for summer outfits, the coat category should not be included in the final outfit. We train the LLM model to predict outfit compositions (outfit category combinations) and retrieve the items in the predicted categories.

We evaluate our framework on the public Polyvore Outfit dataset [29] and measure the quality of generated outfits from two perspectives: 1) feature matching for outfit items and 2) outfit compatibility. Since the Polyvore dataset does not have text product attributes, we use Claude-3 [3] to extract the product features for each outfit item (we verify the precision of Claude-3 with human annotations in Section 4.1). The experiment section shows that our approach outperforms the baseline approaches in the product feature matching and outfit compatibility.

To summarize, the main technical contributions of our work are:

- We propose a text-driven outfit generation framework that generates outfits based on text prompts or images, facilitating outfit recommendation in two forms.
- We develop a large language model framework to predict the embeddings (and compositions) to retrieve the outfit items where they match the input prompt and are visually compatible.
- We devise an attention masking mechanism to handle the alignment between the text descriptions in the prompt and the image tokens from different outfit categories.

2. Related Work

2.1. Outfit Recommendation

Outfit Compatibility Prediction: Outfit compatibility prediction evaluates how well a set of fashion items are compatible with each other in an outfit. Early works [21, 30] focused on pairwise item-level compatibility by leveraging item co-view or co-purchase information. However, compatibility is determined only by the pairwise items, not the entire outfits. Later works such as [7, 9, 27, 29] addressed the compatibility problem at the outfit level. They use different types of architectures to model the relationship between outfit items (e.g., bidirectional LSTM [9], subspace embeddings [27, 29] and graph convolution networks (GCN) [7]). These methods can be used for the fill-in-the-blank (FITB) problem - given a subset of items in an outfit and a set of candidate items, select the most compatible candidate. [4] extends FITB to build an entire outfit from a seed item - it starts from a seed item and iteratively finds the complementary items to build an outfit. However, the composition (outfit category combination) is pre-determined from the existing outfits. It is unclear how to compute the compositions for the new seed items.

These approaches are mainly designed for compatibility prediction and FITB. In contrast, our framework generates outfits based on the input text prompt and predicts the outfit composition given a seed item.

Outfit Complementary Item Retrieval: Complementary item retrieval is to find a compatible item(s) to complete a partial outfit from a large database. Lin *et al*. [20] designed a category-based attention mechanism that enables scalable indexing and searching for complementary items. Sarka *et al*. [25] generated the embeddings using a transformer-based architecture that encodes the compatibility for the partial outfit. These approaches address the FITB problem in a scalable manner (support large-scale indexing and complementary item retrieval), which differs from our problem settings.

Outfit Retrieval: CLIP4Outfit [11] trained transformer encoders to learn cross-modal similarity (the similarity be-

tween text descriptions and outfit images) and used outfit-level embeddings to retrieve the most relevant outfit given the text descriptions. However, this approach considers the whole outfit as a candidate for retrieval. To be able to match a variety of text descriptions, the approach requires constructing a large-scale outfit dataset that covers different types of outfits, which is not feasible in practice. Moreover, their framework does not support outfit generation based on a given seed item. In contrast, our model is an outfit-building approach, retrieving individual items and assembling them into complete outfits.

2.2. Text-to-Image Retrieval

Most related work focuses on text-to-single image retrieval. CLIP [12] learned joint representations of text and images through large-scale contrastive learning. These representations can be fine-tuned on the domain-specific datasets to achieve higher performance, such as Fashion-CLIP [6] for the fashion domain. Later works (e.g., Flamingo [1], BLIP [16] and BLIP-2 [17]) learned a unified vision-language representation space and translated visual features into LLM to generate captions for images. The unified vision-language representation (e.g., Q-Former [17]) can be used for text-to-image retrieval. Some recent work leverages the LLM model to predict embeddings for retrieval [14, 15], which performs better than CLIP embeddings for longer captions. FROMAGe [15] takes the interleaved image and text as input and produce image embeddings and text as output. GILL [14] extends FROMAGe to support image retrieval and image synthesis. Similar to [14, 15], our framework also uses learnable image tokens for retrieval. In contrast to prior work on single image retrieval, our approach focuses on retrieving multiple items to build the outfits (image set retrieval).

2.3. Text-to-Image/Outfit Synthesis

Most work focuses on generating a single image given the text prompt [5, 8, 23, 26]. Some recent work [13, 18] generates a full-body human image wearing garments that reflect the provided textual descriptions. However, the synthesized images may not fully preserve the attribute details specified in the prompt as mentioned by [18]. In addition, to obtain shoppable content, these approaches require using a retrieval system to retrieve similar images from the product catalog (the generated images may not be able to find similar catalog images). In contrast, our framework is designed for outfit generation and retrieves outfit items that preserve the attributes specified in the prompt.

3. Method

Figure 2 illustrates the system overview of our framework. Our framework supports both forms: 1) text-to-outfit generation and 2) seed-to-outfit generation. For text-to-outfit generation, the task is to find the outfit items that match the text descriptions described in the input prompt. We use a set of image tokens and learn the image embeddings for different categories. We fine-tune the LLM to learn the embeddings using the outfit retrieval and compatibility losses, which encourage embeddings to retrieve the items that match the product features and are visually compatible. We devise a masking attention approach to handle the alignment between the text descriptions in the prompt and the image tokens. Details are described in Section 3.1.

For seed-to-outfit generation, the task is to find outfit items to complete an outfit given a seed item and a prompt that describes the global outfit features. We train the LLM model to predict the outfit compositions (outfit category combination) and the embeddings to search the relevant outfit items to generate outfits. Details are described in Section 3.2

3.1. Text-to-Outfit Generation

We are given an input text prompt: $T = \{T_1, T_2, ..., T_k\}$, where each text description T_i describes the outfit-level specification (e.g., outfit for spring and casual) or the item-level specification (e.g., a blouse in white with off-shoulder design). The order of the text descriptions for each category is arbitrary. The goal is to find the outfit items that match the text descriptions in the prompt (cf. Figure 2 (a)).

To produce embeddings, we add a set of image tokens to LLM: $[\mathrm{IMG}_{\mathrm{cat}_1}, ..., \mathrm{IMG}_{\mathrm{cat}_n}]$; each image token corresponds to a high-level category (we use 11 image tokens for Polyvore dataset; the category list is described in Section 4.3). The hidden state vectors of these image tokens from LLM are denoted as $\mathrm{h}_{\mathrm{cat}_1}, ..., \mathrm{h}_{\mathrm{cat}_n}$. We learn a set of MLP layers: $\mathrm{W}_{\mathrm{ret}} = \left\{\mathrm{W}^{\mathrm{ret}}_{\mathrm{cat}_i}, ..., \mathrm{W}^{\mathrm{ret}}_{\mathrm{cat}_n}\right\}$ to transform the hidden state vectors to the embeddings: $\mathrm{e} = \{\mathrm{e}_{\mathrm{cat}_1}, ..., \mathrm{e}_{\mathrm{cat}_n}\}$, where embedding $\mathrm{e}_{\mathrm{cat}_i}$ is used to retrieve the outfit images for the category i.

An outfit item is represented by an image I and its text descriptions T. We use a visual encoder E_{img} to extract the visual embeddings from the image and use a text encoder E_{txt} to extract text embeddings from the text descriptions. The multi-modal feature of an outfit item is denoted by: $\mathrm{f} = \mathrm{W}_{\mathrm{item}} \cdot (E_{\mathrm{img}}(\mathrm{I}) \parallel E_{\mathrm{txt}}(\mathrm{T}))$, where $\parallel$ is the concatenation and $\mathrm{W}_{\mathrm{item}}$ is a learnable MLP model that is fine-tuned with the outfit retrieval loss.

We train our framework end-to-end using the following losses: outfit retrieval loss, attention mask loss, and outfit compatibility loss. The outfit retrieval loss retrieves items that match the product features mentioned in the prompt. The attention mask loss handles the alignment between the outfit text descriptions and the image tokens. The outfit compatibility loss improves the visual compatibility of the retrieved outfit items.

Outfit retrieval loss. A training batch includes a set of posi-

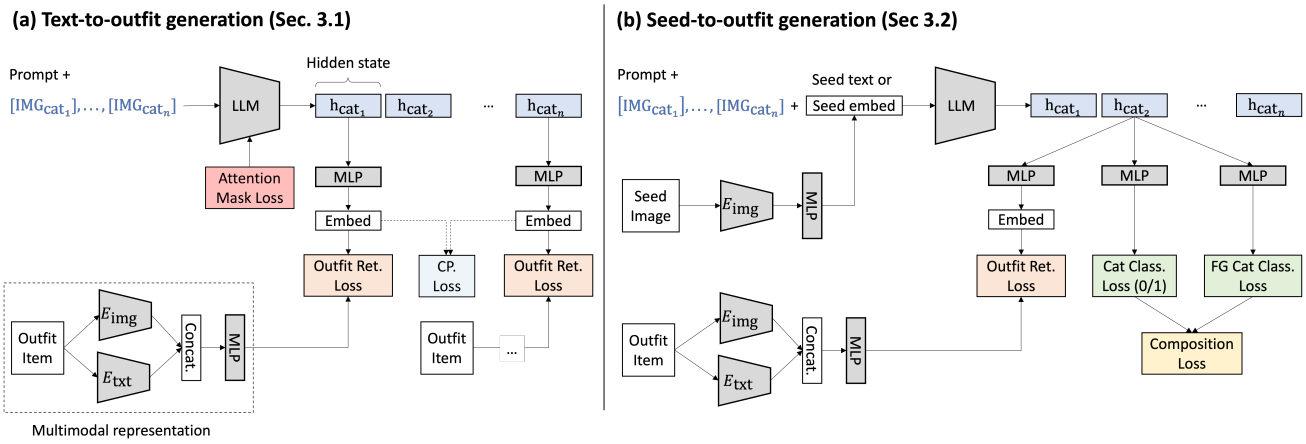

Figure 2. System overview of the text to outfit generation framework. For text-to-outfit generation (a), the LLM framework predicts a set of embeddings for different outfit categories, which are used to retrieve the outfit items. We devise an attention masking approach to handle the alignment between the text descriptions in the prompt and the image tokens from different categories (details are in Section 3.1). For seed-to-outfit generation (b), the LLM framework takes a seed item as input, predicts the outfit compositions (outfit categories), and retrieves items in the predicted categories to build the outfits (details are in Section 3.2).

tive outfits. We randomly sample a set of negative outfits for each positive outfit, where each item in the negative outfits has the same high-level category as the positive outfit. Since there are no ground truth negative outfits, random negatives are typically used in outfit training (e.g., [20, 25, 29]). We extract the occasion, season, color and product features of the outfit items using Claude-3 (details are in Section 4.1). To simulate the prompt for training, we randomly select the product features of each outfit item and use Claude-3 to generate the free-form prompt. The goal is to learn the embeddings so that the similarity between the predicted embeddings and the positive outfits is greater than the negative ones.

Specifically, a batch includes a set of triples: $\Upsilon = \{T, p, N\}$, where T is the prompt, p is the positive outfit and N is a set of negative outfits. The outfit retrieval loss is defined as:

$$\mathcal{L}_{\text{ret}}(p,\ N;\ \text{e}) = \sum_{i=1}^{n} \mathcal{L}_{\text{cat}_i}(\text{f}^p_{\text{cat}_i}, \text{f}^N_{\text{cat}_i};\ \text{e}_{\text{cat}_i})\xi(\text{cat}_i) \quad (1)$$

We combine the loss for each category to obtain the outfit retrieval loss. ξ is the indicator function denoting if the category i is presented in the outfit. The loss for each category is defined as:

$$\mathcal{L}_{\text{cat}_i}(\text{f}^p_{\text{cat}_i}, \text{f}^N_{\text{cat}_i};\ \text{e}_{\text{cat}_i}) = -\frac{\exp(\text{sim}(\text{f}^p_{\text{cat}_i}, \text{e}_{\text{cat}_i})/\tau)}{\sum_{n=1}^{N}\exp(\text{sim}(\text{f}^n_{\text{cat}_i}, \text{e}_{\text{cat}_i})/\tau)}, \quad (2)$$

We adopt the InfoNCE loss ([28]). $\text{f}^p_{\text{cat}_i}$ is the feature for positive outfit item and $\text{f}^n_{\text{cat}_i}$ is the feature for negative outfit item. e_{cat_i} is the embeddings. τ is the temperature. We use the cosine function for the sim function.

Attention Mask loss. The prompt includes the text descriptions for multiple categories. We observed the alignment problem where the image token from a category may attend to the text descriptions from other categories. For example, the top image token attends to the shoe text descriptions. An example prompt: "An outfit for spring and formal event. Black high heels. A black dress with a rose print. A red bracelet with gemstones." The text describes the features for fine-grained categories like high-heel, dress, etc, but the image tokens correspond to high-level categories (e.g., tops, bottoms, shoes). These pose the challenge for the LLM model to find the relations between the outfit text descriptions and multiple image tokens, which affects the product feature matching performance (cf. Table 3)).

To address this issue, we design a masking approach in the self-attention layers of the LLM decoder. Figure 3 illustrates the mask attention for the top image token. We incorporate the mask loss to guide attention learning, where the mask loss is applied to all image tokens. We compute the binary cross-entropy loss (BCE) for each image token between the ground truth mask and the predicted attention values. The mask loss is defined as:

$$\mathcal{L}_{\text{mask}} = \sum_{i=1}^{n}\sum_{j=1}^{k} \text{BCE}_i(\alpha_j, m_j^{\text{GT}}), \quad (3)$$

where n is the number of image tokens. k is the number of tokens from the text descriptions. α_j is the predicted attention value and m_j^{GT} is the ground truth mask.

Compatibility loss (CP. loss). We include a compatibility loss to facilitate the embedding learning that improves the compatibility of the generated outfit items. We trained an

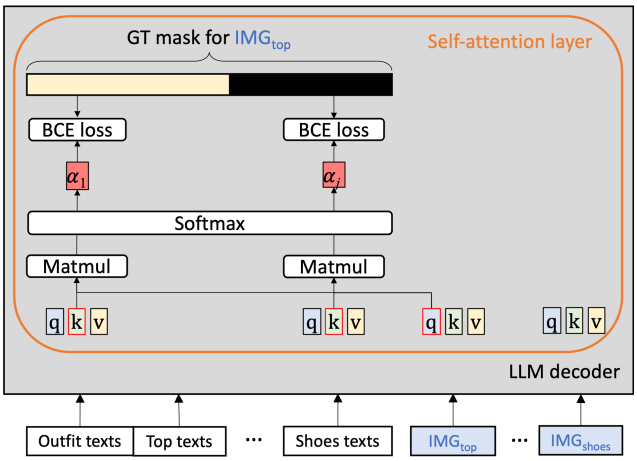

Figure 3. The figure illustrates the masked attention for the top image token, where the mask facilitates the top image token to attend to the right text descriptions (outfit text and top text sections, highlighted in yellow) in the prompt. Details are in Section 3.1.

outfit compatibility model (details are in Section 4.2), which predicts a compatibility score given a set of outfit items. In training, we freeze the model weights of the compatibility model (denoted as $\mathcal{C}$) and fine-tune the embeddings (e) such that the compatibility score is maximized. The compatibility loss is defined as:

$$\mathcal{L}_{\text{cp}}(e) \;=\; -\mathcal{C}\left(\mathrm{W}_{\text{cp}} \cdot \mathrm{e}\right) \tag{4}$$

Training and Inference. We train the framework end-to-end using LoRA [2] by minimizing the following loss function:

$$\begin{aligned} &\min_{\theta, \mathrm{W}_{\text{ret}}, \mathrm{W}_{\text{item}}, \mathrm{W}_{\text{cp}}} \mathcal{L}\left(\theta, \mathrm{W}_{\text{ret}}, \mathrm{W}_{\text{item}}, \mathrm{W}_{\text{cp}}\right) \\ &\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{\text{ret}} + \lambda_2 \cdot \mathcal{L}_{\text{mask}} + \lambda_3 \cdot \mathcal{L}_{\text{cp}}, \end{aligned} \tag{5}$$

where θ is LLM model weights. W_{ret}, W_{item}, and W_{cp} are the MLP models for outfit retrieval, outfit item feature extraction, and compatibility, respectively. $\mathcal{L}_{\text{ret}}$ is the outfit retrieval loss, $\mathcal{L}_{\text{mask}}$ is the mask loss and $\mathcal{L}_{\text{cp}}$ is the compatibility loss. λ_1, λ_2, and λ_3 are empirically chosen to make the magnitudes of the losses similar in scale. We report the results of using different losses in Table 3.

In the inference, we feed the text prompt to the LLM model to predict the embeddings and use them to retrieve the outfit items.

3.2. Seed-to-Outfit Generation

The task is to generate a complete outfit given an input seed item and a text prompt describing the global outfit features. Unlike the text-to-outfit generation task, where users specify the texts for each outfit item, here, users only specify the information for the seed item. Therefore, we need to predict the composition of the remaining items. One can retrieve the outfit items from all the categories to build an outfit. However, some categories do not go well with certain types of occasions, seasons, or the seed types. For example, coats should not be included for summer outfits, or an all-body dress should not be selected if the seed is a top.

Composition loss. To address this problem, we train the LLM model to predict the outfit compositions (a set of outfit categories), where the compositions are learned from outfit datasets (cf. Figure 2 (b)). Specifically, the input includes a prompt and a seed item (users provide an image or a text description for the seed item. For the image, we use a visual encoder to extract the embeddings and feed them to the LLM). To predict the compositions, we add two MLP models (one for predicting the high-level category and another for predicting the fine-grained category) on the top of the hidden state vectors of the high-level category to predict the composition. The composition loss is defined as:

$$\mathcal{L}_{\text{comp}} = \sum_{i=1}^{n} \mathrm{CE}_i(y^{\text{hc}}_{\text{pred}}, y^{\text{hc}}_{\text{gt}}) + \mathrm{CE}_i(y^{\text{fg}}_{\text{pred}}, y^{\text{fg}}_{\text{gt}}), \tag{6}$$

where the hc denotes the high-level category and fg denotes the fine-grained category. CE is the cross-entropy loss. We train the framework with LoRA end-to-end with the following loss:

$$\mathcal{L} = \lambda_4 \cdot \mathcal{L}_{\text{ret}} + \lambda_5 \cdot \mathcal{L}_{\text{cp}} \;+\; \lambda_6 \cdot \mathcal{L}_{\text{comp}} \tag{7}$$

The loss function includes the retrieval, compatibility, and composition prediction losses. We train the model to predict the composition of the seed item belonging to the top category. The performance of other categories for the seed item is in Appendix Table 2. In inference, we search for outfit items in the predicted high-level and fine-grained categories to compose an outfit.

4. Experiments

4.1. Dataset

We evaluate the model performance on the Polyvore dataset [29]. The dataset includes two different sets: a disjoint set and a non-disjoint set. The non-disjoint set contains 53,306 training outfits and 10,000 testing outfits, while the disjoint set contains 16,995 training outfits and 15,145 testing outfits. We train our framework on the non-disjoint training set as it contains a larger number of outfits and report the performance on both non-disjoint and dis-joint testing sets (results are in the Appendix Section 1).

The Polyvore dataset does not have annotated product attributes. Motivated by recent works that leverage LLM for annotations (e.g., [19, 31]), we leverage Claude-3 [3] to extract outfit-level attributes (occasion and season) and item-level attributes (color and product features). We verify

the agreement between the predicted attributes from Claude-3 and humans on a subset of testing samples (500 outfits and 2629 items). The results show the Claude-3 annotated attributes achieve 94.7% overall precision (97.3% for color, 95.5% for product feature, 92.2% for occasion, and 93.6% for season), verifying the feasibility of using Claude-3 for attribute annotations.

4.2. Evaluation

We evaluate the quality of the retrieved outfits in two ways: 1) feature matching for outfit items and 2) outfit compatibility. To evaluate feature matching, we compute the number of items in the top-10 retrieval results that match the product features specified in the prompt. We evaluate the feature matching for color, occasion, season, and product feature attributes for the outfit items, which are critical attributes for outfits. To handle non-exact matches, where the product features are different but their meanings are similar, we use ModernBERT embedding [22] to evaluate the affinity of two features (we select the thresholds of different attributes using a validation set). We report the results using precision@top-10 (denoted as P@10).

To evaluate outfit compatibility, we train an outfit compatibility model using a SOTA architecture (OutfitTransformer [25]), where the model takes a set of outfit items as input, and predicts a compatibility score. We replace the ResNet features in the original architecture with the Fashion-CLIP embeddings [6] and use the item image and the text description as input. We achieve better outfit compatibility prediction than the original architecture. We achieve 94.93% (93% for OutfitTransformer) and 93.88% AUC (88% for OutfitTransformer) for the non-disjoint and disjoint sets respectively. We report the compatibility (CP) score for the top 10 retrieved outfits.

There is no ground-truth outfit composition given a seed and prompt. To obtain a quantitative quality score, we use Claude-3 (we also evaluate the performance with Pixtral Large from Mistral AI, see Appendix Section 2) to evaluate the quality of the predicted compositions. Recent work (e.g., [33]) shows that strong LLMs achieve high agreement with humans. We evaluate the quality in two ways: 1) We use Claude-3 to predict a quality score for each predicted composition and report an average quality score (denoted as Comp. score). 2) Given two compositions generated by the baseline and our approach, we ask Claude-3 to pick the better composition and report the winning rate of our approach (denoted as Comp. win rate).

4.3. Baselines and Model Settings

There is no existing framework for text-to-outfit retrieval tasks ([11] is designed to retrieve entire outfits, which is different from our settings as mentioned in Section 2). We convert a set of baseline approaches to the outfit retrieval tasks and report the performance. In Table 1, we categorize these baselines into the text-to-single-item retrieval and text-to-outfit retrieval approaches. Existing methods for text-to-image retrieval [6, 14, 24] mainly focus on single-image retrieval. To make these methods applicable to outfit retrieval problem, we make the following changes: 1) We convert the outfit-level specification (occasion/season) into item-level specification. 2) We use the text descriptions of each category to retrieve the outfit items (assuming that the text descriptions to category correspondence are given).

For text-to-outfit retrieval, we converted the GILL [14] framework to support multiple-category retrieval. For text-to-outfit retrieval methods, the text descriptions to category correspondence information are not given, which is directly predicted by the models (except the fixed mask approach in Table 1, which uses the text-to-category correspondence to generate the fixed mask for attention). The details of the baseline approaches are described below.

- **CLIP** [24]: We convert the outfit-level specification into item-level specifications and perform retrieval for each category. We use ViT-L-14 version.
- **Fashion-CLIP 2.0** [6]: We use the same settings as CLIP and use Fashion-CLIP to retrieve items for each category. We use version 2.0, which achieves better performance than version 1.0.
- **BLIP-2** [17]: We use the pre-trained image encoder and Q-Former from BLIP-2. Like CLIP, we perform image retrieval for each category using the corresponding text description. We use the ViT-g version.
- **GILL [14] + Multiple categories**: We add the learnable image tokens for each high-level category for multiple image retrieval. We fine-tune the MLP layers to learn embeddings while keeping the LLM model fixed following the original training approach. We use the same prompt format as our model. Note that we compare our method with an improved version of GiLL, in which we replace the original CLIP vision encoder with Fashion-CLIP

Model Settings: For our framework, we use the pre-trained OPT 6.7B model [32] (same as GILL) as the LLM backbone. We fine-tune the LLM with LoRA (we set the rank to 64 for Q, K, and V projection matrices). We use Fashion-CLIP 2.0 [6] as the visual encoder E_{img} and the text encoder E_{text}. Fashion-CLIP 2.0 achieves better performance than CLIP embeddings. In model training, the visual and text encoder weights are fixed. We use the bi-directional masks for image tokens. We use 11 image tokens for the Polyvore dataset, corresponding to the following high-level categories: tops, bottoms, all-body, outerwear, shoes, bags, jewelry, accessories, sunglasses, hats, and scarves.

4.4. Text-to-Outfit Performance Evaluation

Table 1 summarizes the text-to-outfit retrieval results. We use the testing set of the Polyvore dataset to generate the

Method	Modality	Color (P@10)	Product feature (P@10)	Season (P@10)	Occasion (P@10)	Avg (P@10)	CP. score
Text-to-single-item retrieval							
CLIP [24]	Text only	96.06	70.91	90.22	76.62	83.46	63.49
Fashion-CLIP [6]	Text only	**96.82**	80.42	89.68	81.27	87.05	64.07
BLIP-2 [17]	Text only	93.29	66.94	85.46	84.51	82.55	65.09
CLIP [24]	Text to image	83.59	61.32	85.49	67.01	74.35	49.77
Fashion-CLIP [6]	Text to image	87.59	72.67	85.91	68.88	78.76	53.03
BLIP-2 [17]	Text to image	86.39	66.69	85.62	67.75	76.61	51.05
Text-to-outfit retrieval							
GILL [14] + multiple category	Text to image	48.19	48.15	86.87	69.69	63.23	74.26
Text2Outfit + learnable mask (ours)	Text to image	78.42	76.54	86.60	70.45	78.00	66.28
Text2Outfit + learnable mask (ours)	Text to image/text	86.05	<u>84.81</u>	**95.69**	**93.66**	<u>90.05</u>	**84.05**
Text2Outfit + fixed mask (ours)	Text to image/text	<u>96.61</u>	**86.27**	<u>94.87</u>	<u>92.48</u>	**92.56**	<u>82.51</u>

Table 1. Text-to-outfit retrieval performance comparison. P@10 denotes precision at top-10 and CP. score denotes compatibility score using [25] for top-10 retrieved outfits. Note that for single-item retrieval approaches, the text descriptions and its corresponding category are given while the outfit retrieval approaches are not given such information. The bold text denotes the best performance and the underlines denotes the second best performance. Details are in Section 4.2.

Method	Composition	Season (P@10)	Occasion (P@10)	Comp. score (c3)	Comp win rate (c3)	CP. score	CP. score (c3)
Seed text + prompt (baseline)	All categories	97.32	96.20	2.59	6%	80.82	2.78
Seed text + prompt (ours)	Predicted	97.56	96.01	**4.57**	**94%**	**86.18**	**3.85**
Seed image + prompt (baseline)	All categories	95.00	87.50	2.85	8%	72.88	2.66
Seed image + prompt (ours)	Predicted	96.43	91.80	**4.32**	**92%**	**83.36**	**3.83**

Table 2. Seed-to-outfit retrieval performance comparison. The scores with (c3) are evaluated by Claude-3. Details are in Section 4.5.

prompt. Given a test outfit including multiple outfit items, we randomly select outfit attributes (occasion and season) and item attributes (color and product features) and use Claude-3 to generate the free-form prompt. We use the generated prompt to evaluate the outfit retrieval performance.

Our approaches achieve 90.05% and 92.56% average P@10 using the learnable mask and fixed mask respectively, showing better feature-matching precision than the baseline approaches. The GILL + multi-category approach has an alignment issue (the alignment between the outfit texts and the images tokens) while our approach uses the learnable mask to alleviate the alignment issue. We fine-tune the model end-to-end with LoRA instead of using the frozen LLM model as GILL, improving the performance (see Table 3 for detailed analysis). Our approach with a learnable mask also performs better than the single-item retrieval approaches, where they are given the text description to category correspondence information, while our approach does not.

In addition, our approach shows better outfit compatibility than the baseline approaches. We achieve the compatibility score 84.05 using the learnable mask setting (we compute the average compatibility scores for the ground truth positive (86.95) and negative (22.99) outfits on the Polyvore testing set using OutfitTransformer. The generated outfits from our method achieve similar compatibility as the positive outfits). Single-item retrieval approaches perform worse on outfit compatibility because they find items that match the text descriptions but do not model the item relationships in an outfit. Our framework is fine-tuned on the outfit data, and incorporates the outfit retrieval and compatibility losses, which improve overall compatibility. Figure 4 shows example results of our framework.

4.5. Seed-to-Outfit Performance Evaluation

Table 2 summarizes the results for seed-to-outfit retrieval. We used the Polyvore test set to construct the input (seed and prompt) for evaluation. Given a test outfit including multiple outfit items, we randomly select the outfit attributes (occasion and season) and use Claude-3 to generate the free-form prompt.

We compare our approach with a baseline approach that uses all 11 high-level categories for the composition. We feed the compositions (e.g., category 1/fine-grained category 1, category 2/fine-grained category 2, ...) of the baseline and our approach to Claude-3 for evaluation (for the baseline approach, the fine-grained category is obtained from the top-1 retrieved outfit). Our approach shows better composition scores (the score is between 1 (poor) to 5 (excellent)) and winning rates than the baseline (e.g., 94% and 92% winning rate of using seed text and seed image, respectively). We verify the agreement between Claude-3 and humans on a subset of samples (500 outfits). The Spearman's rank correlation coefficient is 0.73, which indicates they are highly correlated. We report the visual compatibility using both OutfitTransformer [25] (CP. score) and Claude-3 (CP. score (c3)). Our approach shows better visual compatibility than the baseline approach.

4.6. Ablation Study

Multimodal feature: We compare the performance of single-modality (image) vs. multimodality (image + text)

	Modality	LoRA	Mask Loss	CP. loss	Color (P@10)	Product feature (P@10)	Season (P@10)	Occasion (P@10)	Avg (P@10)	CP. score
1	Image	-	-	-	48.31	53.03	85.10	68.62	63.76	53.43
2	Image + text	-	-	-	44.60	47.18	88.94	81.07	65.45	78.62
3	Image + text	✓	-	-	74.74	75.01	95.60	93.90	84.82	86.95
4	Image + text	✓	Global CE	-	79.28	82.23	96.02	94.43	87.99	85.63
5	Image + text	✓	Local BCE	-	85.58	84.87	95.89	93.47	89.95	83.43
6	Image + text	✓	Local BCE	✓	86.05	84.81	95.69	93.66	90.05	84.05

Table 3. Ablation study of different components in our framework. Details are described in Section 4.6.

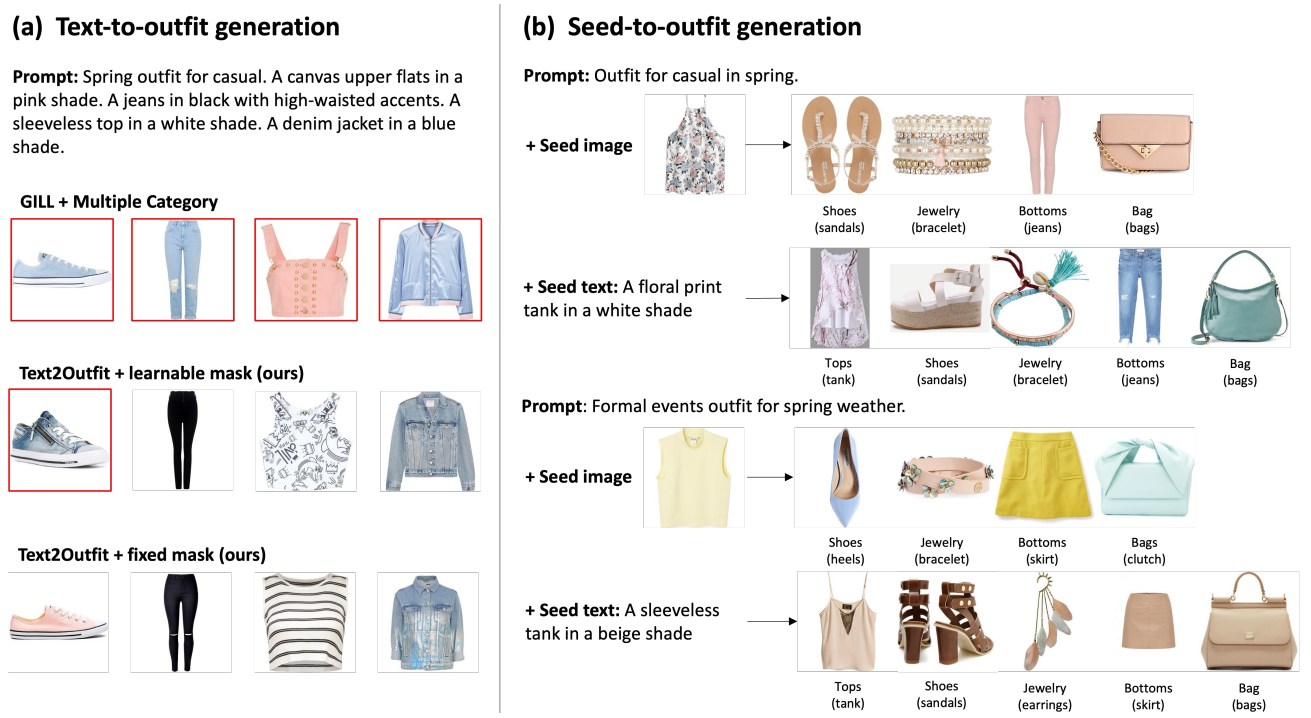

Figure 4. Qualitative results. We compare the results for text-to-outfit generation with the extended GILL [14] approach that supports multiple image retrieval and our framework (learnable mask and fixed mask). The non-matched items are marked with boxes. We show the predicted compositions and retrieval results for seed-to-outfit generation with a seed image or text. More visual results are in the Appendix Section 4.

(row-1 and row-2 in Table 3). Using multimodality improves the average P@10 from 63.76% to 65.45% for feature matching.

LoRA: We compare the performance of fine-tuned LLM (with LoRA) vs. frozen LLM (row-2 and row-3 in Table 3). Using LoRA significantly improves the average P@10 from 65.45% to 84.82% and improves the compatibility score from 78.62% to 86.95%.

Mask loss: We investigate different loss functions for the attention masking: Global CE and Local BCE (row-4 and row-5 in Table 3). For Global CE, we consider the attention values from all the text tokens to each image token as a distribution and optimize a global cross-entropy loss. We optimize each attention value for Local BCE with a binary cross-entropy loss. Local BCE performs better, as it optimizes the attention directly with the ground truth mask, while global CE optimizes the overall distribution of the attention.

Compatibility loss: We analyze the impact of including a compatibility loss in the training loss (cf. Equation 5). It further improves compatibility (row-6 in Table 3).

5. Conclusions

We present a framework for text-based outfit generation. We explore two different forms of outfit generation: 1) text-to-outfit generation and 2) seed-to-outfit generation tasks, which enable the controllable outfit generation. In addition, we introduce a masking mechanism which better handles the alignment between outfit texts and the image tokens, and achieves comparable performance when using the ground-truth masks. The experimental results demonstrate that our model outperforms the baseline approaches for text to outfit retrieval tasks.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *NeurIPS*, 35: 23716–23736, 2022. 3

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024. 5

[3] Anthropic. Introducing the next generation of claude. https://www.anthropic.com/news/claude-3-family, 2024. 2, 5

[4] Federico Becattini, Federico Maria Teotini, and Alberto Del Bimbo. Transformer-based graph neural networks for outfit generation, 2023. 2

[5] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023. 3

[6] Patrick John Chia, Giuseppe Attanasio, Federico Bianchi, Silvia Terragni, Ana Rita Magalhães, Diogo Goncalves, Ciro Greco, and Jacopo Tagliabue. Contrastive language and vision learning of general fashion concepts. *Scientific Reports*, 12 (1):18958, 2022. 3, 6, 7

[7] Guillem Cucurull, Perouz Taslakian, and David Vazquez. Context-aware visual compatibility prediction. In *CVPR*, pages 12617–12626, 2019. 1, 2

[8] Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, Haoran Wei, Xiangwen Kong, Xiangyu Zhang, Kaisheng Ma, and Li Yi. DreamLLM: Synergistic multimodal comprehension and creation. In *ICLR*, 2024. 3

[9] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *ACM MM*, pages 1078–1086, 2017. 1, 2

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 2

[11] Junkyu Jang, Eugene Hwang, and Sung-Hyuk Park. Lost your style? navigating with semantic-level approach for text-to-outfit retrieval. In *WACV*, pages 8066–8075, 2024. 2, 6

[12] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, pages 4904–4916. PMLR, 2021. 3

[13] Yuming Jiang, Shuai Yang, Haonan Qiu, Wayne Wu, Chen Change Loy, and Ziwei Liu. Text2human: Text-driven controllable human image generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022. 3

[14] Jing Yu Koh, Daniel Fried, and Russ R Salakhutdinov. Generating images with multimodal language models. *NeurIPS*, 36, 2023. 1, 2, 3, 6, 7, 8

[15] Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. Grounding language models to images for multimodal inputs and outputs. In *ICML*, pages 17283–17300. PMLR, 2023. 1, 2, 3

[16] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pages 12888–12900. PMLR, 2022. 3

[17] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742. PMLR, 2023. 3, 6, 7

[18] Shikai Li, Jianglin Fu, Kaiyuan Liu, Wentao Wang, Kwan-Yee Lin, and Wayne Wu. Cosmicman: A text-to-image foundation model for humans. In *CVPR*, pages 6955–6965, 2024. 3

[19] Wei Li, Hehe Fan, Yongkang Wong, Yi Yang, and Mohan Kankanhalli. Improving context understanding in multimodal large language models via multimodal composition learning. In *Forty-first International Conference on Machine Learning*, 2024. 5

[20] Yen-Liang Lin, Son Tran, and Larry S Davis. Fashion outfit complementary item retrieval. In *CVPR*, pages 3311–3319, 2020. 1, 2, 4

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015. 1, 2

[22] Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2024. 6

[23] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 3

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 1, 6, 7

[25] Rohan Sarkar, Navaneeth Bodla, Mariya I Vasileva, Yen-Liang Lin, Anurag Beniwal, Alan Lu, and Gerard Medioni. Outfittransformer: Learning outfit representations for fashion recommendation. In *WACV*, pages 3601–3609, 2023. 1, 2, 4, 6, 7

[26] Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative multimodal models are in-context learners. In *CVPR*, pages 14398–14409, 2024. 3

[27] Reuben Tan, Mariya I Vasileva, Kate Saenko, and Bryan A Plummer. Learning similarity conditions without explicit supervision. In *ICCV*, pages 10373–10382, 2019. 2

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. 4

[29] Mariya I Vasileva, Bryan A Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. Learning type-aware embeddings for fashion compatibility. In *ECCV*, pages 390–405, 2018. 1, 2, 4, 5

[30] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*, pages 4642–4650, 2015. 1, 2

[31] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help, 2021. 5

[32] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 6

[33] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 6