An Interactive Framework for Image Retrieval

Nikhil Mehta
Department of Computer Science
Purdue University
West Lafayette, IN 47906
mehta52@purdue.edu

Fan Yang
Amazon Alexa AI
Cambridge, USA
fyaamz@amazon.com

Stephen Rawls
Amazon Alexa AI
New York, USA
sterawls@amazon.com

Loris Bazzani
Amazon AI
Cambridge, USA
bazzanil@amazon.de

Chengwei Su
Amazon Alexa AI
Cambridge, USA
chengwes@amazon.com

Emre Barut
Amazon Alexa AI
Cambridge, USA
ebarut@amazon.com

Abstract

While online shopping, customers often see a product that they have a preference for, but do not purchase it due to not liking a few aspects of the product (e.g., sleeve type or stripe colors on a shirt), and thus have to continue their search. Instead, if the customer were to select a preferred product and issue a modification query, and the system could find a similar product with the desired modification, then the customer would be more likely to find their ideal product with less exploration. In this work, we propose a graph-based interpretable framework to tackle this problem in a multi-modal image retrieval setup. We then show how to incorporate interactions in the form of user feedback to solve this task more effectively.

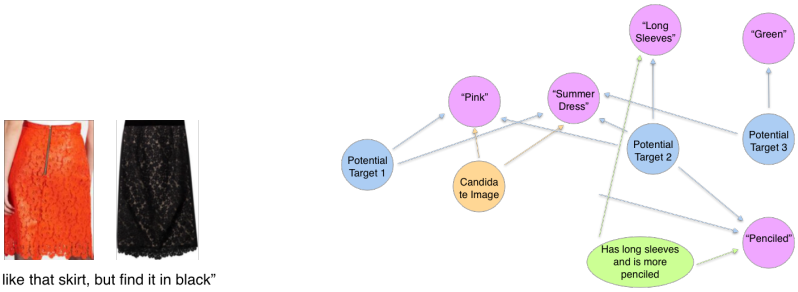
1 Introduction

In today's online shopping world, users often search for products to purchase via a search engine, which are then displayed to them as a sequence of product images. Users can then either select one of the images to purchase, or if they do not like any of the options, issue a new search query in hopes of finding something more relevant to their liking. In the machine learning community, this problem is often known as product-image retrieval. Often times, in product-image retrieval, the user has an idea of the product they are looking for, and wants to convey this information to the retrieval engine. However, this can be challenging, as retrieval engines cannot always find desired products based on the users' few-word query. In such cases, if the user cannot find their ideal products in their initial search, they have to continuously issue different search queries or scroll through recommendations and similar items in order to find what they want - which often does not end successfully.

In this work, we tackle this multi-modal image retrieval problem (user using text to search for a product image) with a different multi-step *interactive* approach, following prior work Vo et al. (2019) and Anwaar et al. (2021). In these settings, after the user's original query returns an initial set of images, the user can successively narrow down their search by selecting from the returned images, and issuing modification queries. The system then finds similar images to the user-selected ones that follow the desired user modifications. Fig 1(a) shows an example, where after an initial query to find skirts, the user likes one of the returned skirts, but wants it in black, and the system is able to find it.

Like prior work Vo et al. (2019); Anwaar et al. (2021), we focus on tackling a single step of this interactive process, framing it as follows: Given a candidate image and a text describing the desired modification, find a target image that expresses it. Throughout the rest of this paper, we refer to this problem as multi-modal product-image retrieval.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) (b)

Figure 1: (a) Example of multi-modal image retrieval. After issuing an initial query, the user is shown the skirt on the left, but wants it black. Thus, they issue a new query with text shown and the image of the skirt. The system finds a similar product with the desired modification, successfully returning the image on the right. (b) Information Graph capturing interactions between images (candidate: orange background, target: blue background), relative captions (green background), and attributes (pink background)

This task is typically modeled using end-to-end deep learning approaches, which achieve strong performances but have several drawbacks. First, they are challenging to interpret (Molnar et al., 2020), which is not desirable in a customer-facing product, as it is difficult to understand the reason behind predictions. Further, lack of interpretability makes incorporating user feedback and interaction difficult. Incorporating interactions can be beneficial, as it allows users to provide feedback to the system, which it can learn from to not only do better in general, but also incorporate user preferences. For example, a user could tell the system some information about a mistake, such as it picked the wrong color, and the system can learn from it do better in the future. In another type of user interaction, the user could interact to explain their preferences (i.e. their style), and the system could then make predictions in line with those in the future. For example, the user may have a preference for long sleeve dresses, and could let the system know that, so it predicts more long sleeve dresses for this user in the future. We feel that users interacting with the model in ways such as these is a crucial component of a multi-modal product-image retrieval system, and design our approach with this in mind.

In addition to lack of interpretability, neural approaches do not take advantage of the highly connected structure of this multi-modal product-image retrieval task. Product images are often tagged with various attributes, many of which are shared across products, linking them together. Assuming this connected structure, the task of finding a target image that is similar to the candidate image but with the required textual modification can be simplified to searching for images with the desired modification amongst images with the same attribute tag as the candidate. Thus, the connected structure can be beneficial.

For this reason, in this work, we propose to tackle this problem as a graph modeling task, using Graph Neural Networks (GNNs). We implement our system on two challenging multi-modal product-image retrieval datasets, Fashion200K (Han et al., 2017) and FashionIQ (Wu et al., 2021), achieving performance improvements over baselines. Further, we take advantage of the interpretable graph structure to make this task more *interactive*. We propose two forms of user interaction, each further improving performance, showing the benefits of our framework in a realistic deployed scenario.

2 Model

Here, we first describe our interpretable graph-based framework, and then present how to train the models. Then, we show how we can solve this task interactively by incorporating user feedback.

2.1 Graph

Our graph aims to take advantage of the inter-connected structure of products and their tagged attributes, which can help solve the task better. Our solution (see Fig 1(b)) consists of the following nodes: (1) I, product images, (2) T, relative captions describing how the candidate image can be modified to get the target image, and (3) A, attributes associated with each image. Each graph node is initially represented by a feature vector, which is modified as the graph is trained (for further

Model	R@1	R@10	R@50
TIRG Vo et al. (2019)	14.1	42.5	63.8
TIRG with Complete Text Query	14.2	41.9	63.3
TIRG with BERT and Complete Text Query	19.9	51.7	71.8
ComposeAE Anwaar et al. (2021)	22.8	55.3	73.4
Our Graph with Predicted Attributes	16.97	40.0	59.91
Our Graph with Caption Based Attributes	82.1	82.5	87.5

Table 1: Fashion200K results. Our graph with caption based attributes (gotten by splitting the caption into two attributes: the first word and the rest of the caption) achieves performance improvements over baselines.

Model	R@10	R@50	R@100
TIRG with Complete Text Query	3.34	9.18	9.45
TIRG with BERT and Complete Text Query	11.5	28.8	28.9
ComposeAE Anwaar et al. (2021)	11.8	29.4	29.9
Our Graph	6.74	19.09	30.2

Table 2: FashionIQ results. We see improvements over baselines on R@100.

details on the graph, see App. A.1). The graph is formed by first adding image (I - candidate: orange background, target: blue background), relative caption (T - green background), and attribute (A - pink background) nodes. Relative captions connect to the candidates they describe modification on. Further, images and relative captions are connected to their attributes (for relative captions it's determined via exact word match - complete attributes that are found in the captions are connected).

Our graph model use the connectivity between attributes and images to learn properties and relationships between various nodes to solve the downstream task. As attributes describe product meta-data and are common across multiple products, they can be used by the graph to learn useful image clusters. Then, the task of finding the target image given the candidate image and relative caption is made simpler, as the model already knows what images are similar to the candidate based on the learned clusters. The model then determines what attribute the caption describes, and finds the images with that attribute near the candidate image space. App A.2 provides an example.

We train a Relational Graph Convolutional Network (R-GCN) to learn a graph embedding function. We optimize two loss functions simultaneously, each with equal weight. In the first, we learn the graph by training link prediction. Given a candidate image and relative caption, the positive example is the true target image, while the negative examples are other random target images. Our second training objective is the compositional loss from Anwaar et al. (2021) - details in their work - who obtain strong performance on this task. Our setup is similar to theirs, except our initial embeddings come from the learned graph, whereas theirs comes from RESNET-18 and BERT.

2.2 Interactivity

A benefit of our graph-structure is its interpretability, as we can determine which nodes contributed to every prediction by looking at edge connections / embedding space similarity. We can take advantage of the interpretability to enable a more *interactive* setup, creating a better end user experience. In this section, we explore two forms of interactions. For both, the user provides the model information via natural language, which is incorporated by making new connections in the graph.

2.2.1 Interactions

More metadata: In the first form of interaction, which we call *more metadata*, the user (ex: database worker) provides the model with some more information about a given product. They interact by choosing an image, and telling the system an attribute about it. For example, they could say "that shirt has long sleeves". In order to incorporate this interaction, we add the user provided attribute to the graph if it does not already exist as a new node, and then connect it to the appropriate image. To determine the attributes in the user description, we use an entity extraction model Akbik et al. (2018).

I like That!: The second interaction is even more customer focused, as the model learns customer preferences. Here, after seeing the model's prediction, the user can say if they liked any of the returned target(s). The model will then learn from the feedback, and predict the liked images in a

Model	R@1	R@10	R@50
AA1: Graph with Predicted Attributes	16.97	40.0	59.91
AA2: Graph w/ Additional 5 Attributes Tagged per image	15.81	75.77	83.63
AA3: Graph w/ Additional 15 Attributes Tagged per image	22.75	75.63	84.06
AA4: Graph with 1/2 Images With 15 Additional Attributes	8.43	59.71	83.95
AA5: Graph with Additional All Caption Based Attributes	82.1	82.5	87.5

Table 3: Fashion200K More Metadata. We achieve the strongest performance when all images are tagged with all aattributes (AA5), but performance is still strong when using only 5/15 (AA2, AA3), tagging only half the images (AA4), or predicting attributes from an external model (AA1).

Model	R@1	R@10	R@50	# Edges	Conn. Acc.
BB1: Graph with Predicted Attributes	16.97	40.00	59.91	-	-
BB2: I Like That! 1000 Examples	16.55	62.26	77.14	58,400	65.73%

Table 4: Fashion200K I Like Like That Interaction. We can see that the "I Like That" interaction improves performance (BB2 vs BB1), particularly in R@10 and R@50. The model also has high connection accuracy, whis is calculated based on how often it is successful at incorporating the interaction (every time it predicts an image the user liked, it should also predict another one the user liked in its top 10).

similar setup in the next iterations (i.e. these liked images and ones similar to it would be more likely to be predicted, as the user likes them). This solution is incorporated via forming an edge between all images that the user selected, connecting each image to the relative caption, and finally connecting each image to the candidate image. This signifies to the model that these images are similar, and are preferred for the current caption and candidate image. Moreover, whenever the model predicts one of these images in the future, there will also be a higher chance of predicting the other ones, as the new connections put the products closer in the embedding space. Similarly, if the model receives a relative caption similar to the one that the interaction was collected for, that caption will be embedded close by, and thus the probability of predicting the images that the user liked would be higher.

Learning User Interactions: We explain the protocol we use to learn interactions in App A.3. A key benefit is that to incorporate user interactions, there is *no new training that needs to be done*. Thus, the interactions can be incorporated seamlessly, enabling a true interactive scenario.

3 Experiments

3.1 Datasets + Graph Results

We evaluate our graph framework on 2 multi-modal image retrieval datasets in Tab 1 and Tab 2. Fashion200K Han et al. (2017), Vo et al. (2019)) and Fashion-IQ (Wu et al., 2021) (Dataset details: A.4. As products in Fashion200K have no attributes, we either predict them, or get them by splitting the caption (which could explain our high performance there as may enable the graph to exploit the dataset). Attribute details in A.5. All models are evaluated at different k values for recall: whether or not the true target image is in the top k predictions by the model. Specifically on Fashion200K we see improvements with our model that uses attributes gotten by splitting the caption. Further, on the challenging Fashion-IQ, we see improvements on R@100. App A.6 shows an ablation study of the graph.

3.2 Interactivity Results

Next, we present results of our interactivity protocols. Our experiments represent a true interactive scenario, where the model does not have to be trained in order to capture the user interaction.

More MetaData: For the "more metadata" interaction, we evaluate on Fashion200K. As we are unable to collect human data for the interaction, we simulate it by parsing each caption into two attributes - the first word, and the rest of the caption, (details: A.5). To use simulated attributes that are general and don't exploit the dataset, we rank all attributes by how often they appear and use only the top k. In Table 3, we show how performance improves compared to the graph model with predicted attributes, when we add the top 5, the top 15, and all attributes. We show the top k attributes (5, 15) that we used in App Sec A.7, and it is clear that these are ones that humans would

have knowledge about. And yet, we see significant improvements (particularly in R@10 - 35.63 - and R@50). This shows how even a little bit of extra metadata via interactions can be very beneficial.

I Like That: We also simulate the "I like that" interaction on Fashion200K by randomly selecting images the user may have liked. To do this, for a given candidate interaction scenario, we search in the embedding space (using FAISS Johnson et al. (2017)) for the 10 most similar predicted images to the target. After the interactions, we also evaluate our model to determine how many interactions the model followed: We consider our model successful at following the interaction if anytime the model predicts on of the images that the user liked, it predicts another one in the set of 10. The results in Tab 4 show that interactions improve overall performance and the model also is able to incorporate them with a high 65.73% accuracy. This shows how the graph framework can allow incorporating user preferences, provided via user interactions.

4 Conclusion

In this work, we propose an approach to tackle the multi-modal image retrieval problem, by using Graphs. Our graphical model is able to take advantage of the highly connected structure of products and their attributes/metadata, to achieve a new state-of-the-art performance on the challenging Fashion-IQ dataset. Further, we show how our solution's interpretability can be exploited to handle two types of user interactions, each improving the performance of the model without any further training, enabling a true interactive scenario.

For future work, our graph can be extended with new node types (and relation types) to feature other metadata that is not product specific, such as manufacturer information or date released. This would further increase the connectivity of the graph, and likely lead to performance improvements. In addition, new interaction types can also be incorporated (as new edges or relations in the graph), along with having humans interact instead of only simulating it.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Muhammad Umer Anwaar, Egor Labintcev, and Martin Kleinsteuber. 2021. Compositional learning of image-text query for image retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1140–1149.

Sheng Guo, Weilin Huang, Xiao Zhang, Prasanna Srikhanta, Yin Cui, Yuan Li, Matthew R.Scott, Hartwig Adam, and Serge Belongie. 2019. The imaterialist fashion attribute dataset. *arXiv preprint arXiv:1906.05750*.

Xintong Han, Zuxuan Wu, Phoenix X Huang, Xiao Zhang, Menglong Zhu, Yuan Li, Yang Zhao, and Larry S Davis. 2017. Automatic spatially-aware fashion concept discovery. In *Proceedings of the IEEE international conference on computer vision*, pages 1463–1471.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. 2020. Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 417–431. Springer.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Nam Vo, Lu Jiang, Chen Sun, Kevin Murphy, Li-Jia Li, Li Fei-Fei, and James Hays. 2019. Composing text and image for image retrieval-an empirical odyssey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6439–6448.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Hui Wu, Yupeng Gao, Xiaoxiao Guo, Ziad Al-Halah, Steven Rennie, Kristen Grauman, and Rogerio Feris. 2021. Fashion iq: A new dataset towards retrieving images by natural language feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11307–11317.

A Appendix

A.1 Graph Details

First, we describe the feature representations for each node in our graph. **Images** are represented by their RESNET-18 He et al. (2016) representation. RESNET embeddings He et al. (2016) extract properties of the images that can be helpful for the model to understand them. **Relative Captions** are represented by the SBERT RoBERTa embeddings of the text. RoBERTa Liu et al. (2019) provides strong language representations, while SBERT Reimers and Gurevych (2019) builds on it to derive semantically meaningful sentence embeddings Reimers and Gurevych (2019). Finally, **Attributes**, as they are text words, are also represented by SBERT RoBERTa. Our R-GCN is a 3 layer model with 128 dimensional hidden units. We use a learning rate of 0.001 with the Adam Optimizer. Models are implemented in PyTorch using the DGL Wang et al. (2019) library.

A.2 Simple Use Case of Graph Framework

To further explain the benefit of our graph framework, we now discuss an overly-simplified case, where the exact attributes the target image is tagged with are present in the caption, and there are some attributes that both the candidate and target image are tagged with. In this overly-simplified case, solving this multi-modal image retrieval task involves three steps in our graph. First, the graph identifies all the images that have the exact attributes as the relative caption. We can call this set X. Then, it finds all images that have the same attributes as the candidate image, call it set Y. Finally, the correct target image is the $X \cap Y$, or the images that are similar to the candidate (have the same attributes), while having the desired modification (attribute) of the relative caption. Of course, this overly-simplified example assumes that all images can be found by checking attributes, and that the relative caption mentions attributes in the graph, which is usually not the case. This is why we must learn an embedding of the graph, and train it to make predictions, which we discuss in the main-paper. However, even in a more complex scenario, the properties of our graph framework that enabled the overly-simplified case to be simple, still apply. Instead of searching based on exact attribute match, the graph neural network model searches in the embedding space that was learned based on attribute and textual connectivity.

A.3 Learning Interactions

We now discuss the protocol we use to initially learn the interactions. After this process is complete, the model learns how to take advantage of the interactions. Thus, whenever any future interactions are done, they can be incorporated into the model directly, without additional training. This allows the model to be continuously updated (and improved) based on interactions, even post-deployment.

For all interactions, we add new edges to the graph to encourage nodes to be closer together. These new edges represent a new relationship type in the R-GCN, which must be learned, which is done by performing some interactions (we do not perform the same interaction types that we evaluate on), and then re-training the model. Once this process completes, when the user is interacting at test time,

Model	R@1	R@10	R@50
A1 : 20%. Conf. (4)	0.3	25.47	51.05
A2 : 10% Conf. (7)	1.2	30.52	58.15
A3 : 2% Conf. (13)	16.97	40.00	59.91

Table 5: Ablation study on number of attributes

there is *no new training that needs to be done*, since the model has already learned the edge type and understands it when making the final prediction.

For More MetaData, we trained with 10 Attributes outside of what we evaluated on, and for I Like That we used 200 examples.

A.4 Dataset Details

Fashion200K Han et al. (2017) is a dataset that initially consisted of images and their captions collected from Google search, and was extended by Vo et al. extended for the multi-modal image retrieval task. They choose candidate images randomly, and then target images were ones that have the same caption as the candidate image, but with a one word difference. Then, the caption of the target image becomes the relative caption. On this dataset, we compare to Anwaar et al. (2021) and Vo et al. (2019).

Fashion-IQ Wu et al. (2021) is a dataset constructed specifically for the multi-modal image retrieval task, where the relative captions were provided by human annotators. Each candidate image / target image pair has two relative captions, which following prior work Anwaar et al. (2021) we combine into one, joining it with the words "and it". Fashion-IQ also has three splits, but like Anwaar et al. (2021) we combine it into one split, adding the name of the dataset split as the first word of the caption. Like with Fashion200K, we compare to Anwaar et al. (2021) and Vo et al. (2019).

A.5 Fashion200K Attributes

As products in Fashion200K have no attributes, which is critical for our graph-based framework, we either predict them, or get them by splitting the caption.

Predicting Attributes: To predict attributes, we use a pre-trained attribute prediction model that was trained on the IMAT 2018 Guo et al. (2019) dataset. We feed in the image, and it predicts the attributes. Although this setup is noisy, in a brief manual study we noticed that many images were annotated with meaningful attributes, and we thus use this is an approximation. This is an area for improvement for future work.

Attributes from Captions: The second approach we used was to extract attributes from the relative captions. For this, we broke up the caption into two parts - first word and the rest - and considered each of them as an attribute. For example, if the caption is: "red long striped t-shirt", the two attributes are "red" and "long striped t-shirt". While this approach worked very well, it may exploit some properties of the dataset (as explained above, target images to construct the dataset were chosen based on having a one word difference in their caption to the candidate) to achieve unfair high performance, so we don't use this approach in our baseline model. For both attribute extraction methods, we connected the attribute nodes to caption nodes.

A.6 Ablation Study

In this sub-section, we perform an ablation study of our graph, which can be seen in Tab 5. As mentioned earlier, in our graph images and captions are connected through attributes. Thus, in this ablation study we look at how performance on Fashion200K changes as the number of attributes increases. We increase the number of attributes by lowering the confidence threshold of the attribute prediction model. As we can see, the more attributes our model has, the higher the performance is, showing that more useful meta-data about the products in the graph would lead to higher overall performance, as the graph is better connected. We tried lowering the conference threshold beyond what is reported in the results, but the performance dropped, showing that having accurate attributes is also important.

A.7 Attributes Used

In this table, we show the 5 and 15 attributes we used in Sec 2.2.1 to achieve significant performance improvements on Fashion200K. All of these attributes are ones that could be easily provided by human interactors.

Here are the ones for the Top 5:

```
black, multicolor, blue, white,
gray
```

And here are the ones for the Top 15:

```
black, multicolor, blue, white,
gray, red, pink, green, natural,
knee length skirt, short dress,
beige, brown, purple,
knee-length dress
```