Augmenting KG Hierarchies Using Neural Transformers

Sanat Sharma[0009−0003−2479−3041], Mayank Poddar, Jayant Kumar[0009−0004−3858−5961], Kosta Blank[0009−0008−6404−9803], and Tracy King[0000−0002−7956−505X]

Adobe Inc.

Abstract. This work leverages neural transformers to generate hierarchies in an existing knowledge graph. For small (<10,000 node) domain-specific KGs, we find that a combination of few-shot prompting with one-shot generation works well, while larger KG may require cyclical generation. Hierarchy coverage increased by 98% for intents and 95% for colors.

Keywords: knowledge graphs · hierarchy generation · few-shot prompting.

1 Introduction

Knowledge graphs (KG) are widely used in industry to understand user behavior and provide contextual recommendations (figure 1) and search results. At Adobe, we utilize a KG to understand users' creative intent and recommend Adobe assets based on the intent. While the original KG had over 4000 intent nodes, the original taxonomy was mostly flat, lacking substantial hierarchies that could amplify the semantic significance between nodes and drive additional intent-based recommendation use cases.

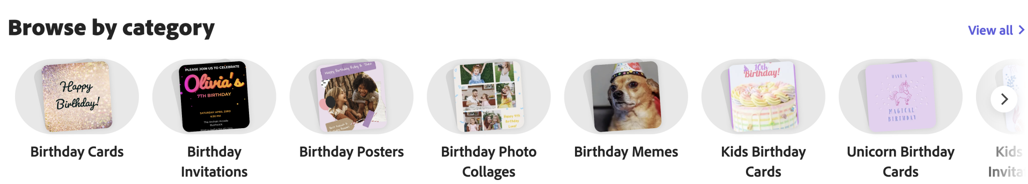

Fig. 1. Adobe Express SEO page for *birthday* with related pages powered by the intent-based KG.

In this work, we present a novel approach to automatically generate intricate graph hierarchies in KGs by leveraging neural transformers. We enhance the structure of our graph by generating hierarchies for both intent (what an Adobe user wants to accomplish, e.g. create a child's birthday card or a banner for their cafe's website) and color node types, resulting in a significant increase in hierarchy coverage: 98% for intent and 95% for color. Hierarchies have key benefits to our users. **Organizational Structure**: Hierarchical relationships makes it easier to navigate and comprehend the KG. Hierarchies help maintain order and provide a clear understanding of how different concepts are related to each other. **Semantic Relationships**: Rich intent hierarchies allow us to capture the semantic relationships between concepts. They also help us unlock key features, such as powering browse and SEO relationships. **Scalability and Flexibility**: Top level categories allow for easier addition of new intents without disrupting the overall structure as the KG grows.

2 Related Knowledge Graph Work

Knowledge graphs are widely used in industry in a variety of roles, from providing social media recommendations [12, 11] to providing entity linking and semantic information between concepts [2, 9]. With recent improvements in attention-based networks, specifically large transformers [5, 10], there has been academic focus towards grounding language models with KGs [6], thereby providing semantic reasoning and generation based on the KG information. Recent works also investigate automated generation and completion of KGs using large transformer models [4, 1]. They utilize language models like ChatGPT to add new nodes to subsets of the graph. While most works focus on adding additional nodes to KGs, our work focuses on augmenting the semantic relationships of existing nodes in the graph using large transformer models. We generate rich hierarchies and associations inside the graph, something that is novel to the field.

3 Approach

First we create top level (L_1) categories for a specific class of nodes (e.g. intent or color). L_1 categories can be selected by domain experts or by a language model. They need to be broad and expansive, as our aim is to transition from a general intent to a more specific one, progressing through multiple levels. We created the L_1 intent nodes by examining Adobe Express frequent queries and their intents, Adobe Stock content categories, and the Google open-source product type taxonomy [3]. This resulted in 26 L_1 intent categories (e.g. Business and Industry, Travel, Shopping, Beauty and Wellness, Health). These overlap with standard taxonomies but comprise a subset relevant to Adobe Stock and Express users.

After establishing the L_1 categories, a classifier module assigns all KG nodes to one or more L_1 categories. Then a generator module enhances the existing hierarchy (if any) with the newly added nodes. Finally, a scalable pipeline auto-ingests the new hierarchies into the KG and queries the KG at inference time. To generate our hierarchies, we utilize two modules (Figure 2):

1. **Classifier Module**: The classifier module takes all nodes to be added to hierarchy and classifies them into one or more of the L_1 candidate classes. We found large language models to be better at few-shot classification than their smaller variants.
2. **Generator Module**: The generator module runs in a loop for each L_1 category. The generator takes the existing hierarchy for that L_1 category (just the L_1 node if no hierarchy exists) and adds all the candidate nodes to generate the updated category hierarchy. We found one-shot hierarchy generation using large language models to be the best approach (§3.2).

3.1 Few-Shot Prompting

In order to use the classification module, we do few-shot learning in which we provide the language model (GPT4 with a 32K context [5]) with a few classification examples and a strict prompt. A sample prompt we used is *"You are a taxonomist, classify the given node to one or more of the provided categories. If you think the category should be its own thing, return Other. Please return a dictionary every time."* With the prompt, we provide a few sample nodes, the categories and an output prediction. Based on a few rounds of samples, we then provide the true candidates for classification to the model. Similar to other approaches in the industry [7, 8], we see a significant boost of 12% in accuracy by doing few-shot learning compared to zero-shot classification.

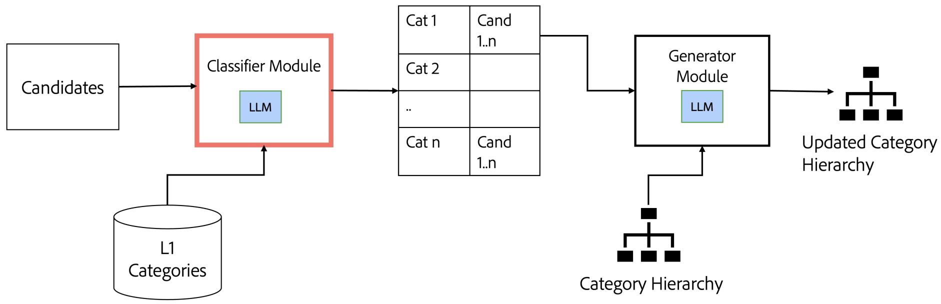

Fig. 2. Hierarchy Generation Approach

3.2 Generation Module

Once the candidates are categorized, we experimented with two techniques in the generation module to create the updated hierarchy. **Cyclical Generation**: Generate each level of the category in a loop. This means that L_2 level nodes are added first, then L_3 and so on. This is needed when the existing hierarchy is large and cannot fit in the model context. **One-Shot Generation**: All candidate nodes are added to the hierarchy in a single pass, without any cyclical generation. We found that this approach produced better results with smaller (<10,000 node) taxonomies.

Cyclical Generation In the cyclical generation approach, we follow a cyclical pattern to generate each level of the hierarchy for an L_1 category and its children.

1. From the candidate set of nodes at level L_i, classify nodes that belong to that level. Utilize the existing nodes at that level (i.e. any nodes already present in the hierarchy) to help the language model via a few-shot approach.
2. The nodes not categorized to be part of level L_i will become part of lower levels ($L_{i+1..n}$).
3. Pass the remaining nodes as well as the existing hierarchy to the generator module to create the new updated hierarchy for level L_i. The generator module will attempt to categorize and place each of the remaining nodes under one of the L_i nodes.
4. For each of the L_i nodes and their hierarchy, repeat the process in a recursive manner to fine-tune the hierarchies. The process stops when either a specified depth (L_i) is reached or all nodes have been added to the hierarchy.

While the cyclical approach is useful, especially for larger graphs, we saw several drawbacks with it when generating our intent hierarchies. **Error Propagation**: LLMs can hallucinate or generate incorrect structured content. Having multiple steps in the generation process can lead to error propagation through the chain. This is the biggest issue with a recursive approach. **The Other Conundrum**: LLMs are bad at placing nodes into the "Other" category. This means that most nodes were assigned into a level's category (L_i) rather than being placed into the Other category (to be a part of the L3 and lower levels). **Order Importance**: Whether we pass nodes in a batch or one at a time for classification, the order of nodes plays a huge difference in the categorizing. For example, if "birthday party" was categorized as an L_i first and then the node "birthday" is shown to the LLM, it often incorrectly categorizes "birthday" as a child of "birthday party" due to their similarities. Additional checks and another overall pass is required to fix the categorizations. One-shot generation (see below) alleviates these issues.

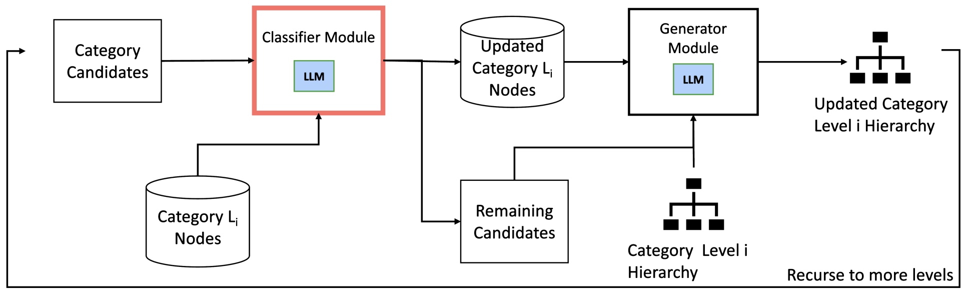

Fig. 3. Cyclical Generation requires classification of nodes and addition at each level

One-Shot Generation In the one-shot generation process, we provide all the nodes to be added as well as the full existing hierarchy to the LLM and allow it to generate the L_2, L_3 and lower categories with just a few examples provided. This approach worked well when there was an existing, partial hierarchy in place to guide the language model's intuition. If a large number of candidates need to be added to the hierarchies, batched generations followed by an overall pass where the language model has the chance to correct any errors is utilized. For our domain-specific use cases, we found full, one-shot generation to be more viable since our taxonomy is relatively small ($<$ 5000 intent nodes and $<$ 500 color nodes).

Updating the Graph For new intents, the above approach is used to integrate them into the KG. When creating intents for a new domain (e.g. Adobe app tools), a subgraph is generated for the new domain and then merged into the existing KG. One algorithm improvement, suggested by an anonymous reviewer, is to examine each subgraph in the generated graph and query the LLM as a third step if it looks good. This uses the LLM for evaluation and updating, instead of generating.

4 Hierarchical KG Evaluation and Conclusions

Statistics on the hierarchical KG generated using one-shot generation and few-shot prompting are summarized below ('Lower' indicates nodes in L_5 or below categories).

KG	Nodes	In Hierarchy Before	In Hierarchy Now	% Change	L_1	L_2	L_3	L_4	Lower
Intents	4639	891	4630	419%	25	383	1826	1937	920
Colors	328	12	328	2100%	12	224	92	1	0

We evaluated the KG hierarchies through a human-in-the-loop approach. We provided a graphical interface to identify nodes that are incorrectly positioned and to offer suggestions for enhancement. We engaged 16 Adobe-internal domain experts to review the hierarchies within each L_1 category for both intent and color nodes. The hierarchies were found to be relevant >95% of the time. Lower levels were spot-checked for accuracy. Identified errors were then manually corrected.

The ultimate evaluation will be in leveraging the KG hierarchies in search and recommendation features. The non-hierarchical graph already provides related search style links between Express SEO pages (figure 1) and powers null and low recovery in Adobe Express by mapping queries and templates to intents. These use cases will be enhanced by using the hierarchy to provide additional links, to type the links, and to provide back-off through the hierarchy. The Express SEO color pages represent the first user-facing application of the hierarchical graph.

Adobe Company Portrait

Adobe Inc. enables customers to change the world through digital experiences and creativity. The Adobe search and discovery team supports search and recommendations across customer text, image, video, and other document types as well as over Adobe Stock assets and Adobe help and tutorials.

Main Author Bio

Sanat Sharma is a senior machine learning engineer at Adobe Inc. He earned his Master's degree from University of Texas, Austin in 2020, with a focus on NLP. Sanat's work focuses on search improvements and contextual recommendations, and his work has been published at conferences such as SIGIR and CVPR.

References

1. Carta, S., Giuliani, A., Piano, L., Podda, A.S., Pompianu, L., Tiddia, S.G.: Iterative zero-shot LLM prompting for knowledge graph construction (2023), https://arxiv.org/abs/2307.01128
2. Dong, X.L., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA, August 24-27, 2014. pp. 601–610 (2014), http://www.cs.cmu.edu/ nlao/publication/2014.kdd.pdf
3. Google: Google product type taxonomy, https://www.google.com/basepages/producttype/taxonomy.en-US.txt, 2021-09-21 version
4. Meyer, L.P., Stadler, C., Frey, J., Radtke, N., Junghanns, K., Meissner, R., Dziwis, G., Bulert, K., Martin, M.: LLM-assisted knowledge graph engineering: Experiments with ChatGPT (2023), https://arxiv.org/abs/2307.06917
5. OpenAI: GPT-4 technical report (2023), https://arxiv.org/abs/2303.08774
6. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: A roadmap (2023), https://arxiv.org/abs/2306.08302
7. Parnami, A., Lee, M.: Learning from few examples: A summary of approaches to few-shot learning (2022)
8. Song, Y., Wang, T., Mondal, S.K., Sahoo, J.P.: A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities (2022), https://arxiv.org/abs/2205.06743
9. Stokman, F., Vries, P.: Structuring knowledge in a graph. In: van der Veer, G.C., Mulder, G. (eds.) Human-Computer Interaction, pp. 186–206. Springer (1988). https://doi.org/10.1007/978-3-642-73402-1_12
10. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open foundation and fine-tuned chat models (2023), https://arxiv.org/abs/2307.09288
11. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the Facebook social graph. CoRR **abs/1111.4503** (2011), http://arxiv.org/abs/1111.4503

12. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. CoRR **abs/1806.01973** (2018), http://arxiv.org/abs/1806.01973