Multimodal Retrieval Augmented Generation over Knowledge Graph

Alkid Baci

Januar, 2025

Paderborn University

Department of Computer Science
Heinz Nixdorf Institute
Data Science Group (DICE)

Software Engineering

Multimodal Retrieval Augmented Generation over Knowledge Graph

Alkid Baci

1. Reviewer Prof. Dr. Axel-Cyrille Ngonga Ngomo
Department of Computer Science
Paderborn University

2. Reviewer Dr. rer. nat. Michael Röder
Department of Computer Science
Paderborn University

Supervisor Dr. rer. nat. Caglar Demir

Januar, 2025

Alkid Baci
Multimodal Retrieval Augmented Generation over Knowledge Graph
Software Engineering, Januar, 2025
Reviewers: Prof. Dr. Axel-Cyrille Ngonga Ngomo and Dr. rer. nat. Michael Röder
Supervisor: Dr. rer. nat. Caglar Demir

Paderborn University
Data Science Group (DICE)
Heinz Nixdorf Institute
Department of Computer Science
Warburger Str. 100
33098 and Paderborn

Abstract

The rapid growth of digital data has reached record levels in the last decade and continues to grow exponentially presenting challenges and opportunities for knowledge representation and application. Retrieval-augmented generation (RAG) is an approach for improving the capabilities of the generative models that has gained great focus in the last four years but the multimodality aspect is less prevalent in this approach which mainly focuses on the text modality. We present a multimodal RAG framework over knowledge graph (KG) for the image recommendation task with generated textual description. By using a KG in this framework, we exploit the benefits that comes with the structured data. The proposed approach is extended further into using class expression learning (CEL) during the augmentation phase. A core part of this work is using the data from a multimodal dataset to generate a KG which is further adjusted for use in the RAG model. We utilize a large language model (LLM) to enrich the KG with additional context and show that the proposed RAG models achieved improved performance during the retrieval phase when operating over the enriched KG. Furthermore, the findings suggest that integrating CEL into the RAG framework can be a promising approach for advancing the application of RAG over KGs.

Zusammenfassung

Das rasante Wachstum digitaler Daten hat in den letzten zehn Jahren ein Rekordniveau erreicht und nimmt weiterhin exponentiell zu, was sowohl Herausforderungen als auch Chancen für die Wissensdarstellung und -anwendung mit sich bringt. Retrieval-augmented generation (RAG) ist ein Ansatz zur Verbesserung der Fähigkeiten der generativen Modelle, der in den letzten vier Jahren große Aufmerksamkeit erlangt hat, aber der Multimodalitätsaspekt ist in diesem Ansatz, der sich hauptsächlich auf die Textmodalität konzentriert, weniger verbreitet. In dieser Arbeit wird ein multimodales RAG-Framework über einen Knowledge Graph (KG) für die Aufgabe der Bildempfehlung mit generierter textueller Beschreibung vorgestellt. Durch die Verwendung eines KG in diesem Rahmen nutzen wir die Vorteile, die mit den strukturierten Daten einhergehen. Der vorgeschlagene Ansatz wird durch die Verwendung von Class Expression Learning (CEL) während der Augmentierungsphase erweitert. Ein Kernstück dieser Arbeit ist die Verwendung von Daten aus einem multimodalen Datensatz, um eine KG zu generieren, die für die Verwendung im RAG-Modell weiter angepasst wird. Wir verwenden ein großes Sprachmodell (LLM), um die KG mit zusätzlichem Kontext anzureichern, und zeigen, dass die vorgeschlagenen RAG-Modelle während der Retrieval-Phase eine bessere Leistung erzielen, wenn sie mit der angereicherten KG arbeiten. Darüber hinaus deuten die Ergebnisse darauf hin, dass die Integration von CEL in den RAG-Rahmen ein vielversprechender Ansatz sein kann, um die Anwendung von RAG über KGs voranzutreiben.

Acknowledgement

This thesis was completed within the Data Science Group (DICE) at Paderborn University under the leadership of Prof. Dr. Axel-Cyrille Ngonga Ngomo. I am deeply grateful to him for providing me the opportunity to explore and present my ideas through scientific work such as this. I extend my heartfelt thanks to my supervisor, Dr. rer. nat. Caglar Demir, for his steadfast support, invaluable guidance, and boundless encouragement throughout this journey, both in this thesis and in my role as a student assistant. I feel privileged to have collaborated with brilliant minds at DICE over the past two years. In particular, I wish to thank Dr. rer. nat. Michael Röder for his generosity in reviewing this thesis.
I am immensely thankful to my family for being there for me through hard times and for bringing joy and motivation to my life. Their sacrifices and resilience have paved the way for me to reach this point and to keep moving forward. Words cannot express how much they mean to me.
Finally, I acknowledge the contributions of the following creators for their icons used in this work's figures: freepik, smashicons, circlon-tech, grafix-point, and lab-design-studio, available at Flaticon.

Contents

Introduction 1

1.1 Motivation and Problem Statement

Knowledge graphs (KGs) provide a structured representation of knowledge/data, capturing relationships between entities and concepts in a given domain and it has various applications including user recommendations, a topic of interest in this work [Hog+21b]. The structured data makes it possible to use machine learning algorithms that fall in the realm of *explainable artificial intelligence* (XAI), which means that the generated predictions can be explained and understood by humans. KGs can integrate data from multiple sources and formats and they provide a semantic framework for representing concepts in a human-understandable language [VO24]. Furthermore, they enable contextualization and personalization of information by capturing contextual relationships and user preferences [Abu+23].

KGs can be used for the retrieval process in *Retrival-Augmented Generation* (RAG) frameworks. RAG is a solution that has become an important part of overcoming challenges such as hallucinations or information irrelevance for *Large Language Models* (LLMs) with rapid development during the last 4 years [Gao+24]. Despite RAG systems using multiple data types for information retrieval, the KG is not adapted as widely and there is a gap in the multimodality aspect of KGs used in RAG systems, which this work is aiming to fill [Gao+24]. Multimodal datasets contain data that may be provided by human beings where human experts annotate the images [Jia+20], by pre-trained classifiers [AH17] and recently by LLMs [Rot+24]. The power of an LLM to understand images is advancing quickly as recent research shows the ability of the LLM to explain medical images of chest X-rays [Lee+23]. LLMs can be used to further extend the information provided on an image-text dataset by using generative models trained on image-text pairs [Rot+24; Rad+21]. Using evaluation measures for information retrieval, we prove that such enrichment with LLM description improves the performance of retrieval models in a RAG framework.

Class expression Learning (CEL) over an *Web Ontology Language* (OWL) or *Resource Description Framework* (RDF) KG, while leveraging *Description Logics* (DLs), can learn explainable concepts used in categorizing instances of that KG [Hei+22;

LH10]. Given that OWL/RDF KGs are used in RAG systems we can integrate CEL into the RAG system to potentially enhance the output relevance. Furthermore, this enhancing process is a white-box approach since class expressions can be explained in human understandable language [Dem23; Leh+11; Hei+22]. The applications of CEL in this scenario can be extended to ontology engineering in order to enrich the KG with more sensible assertions [Leh+11].

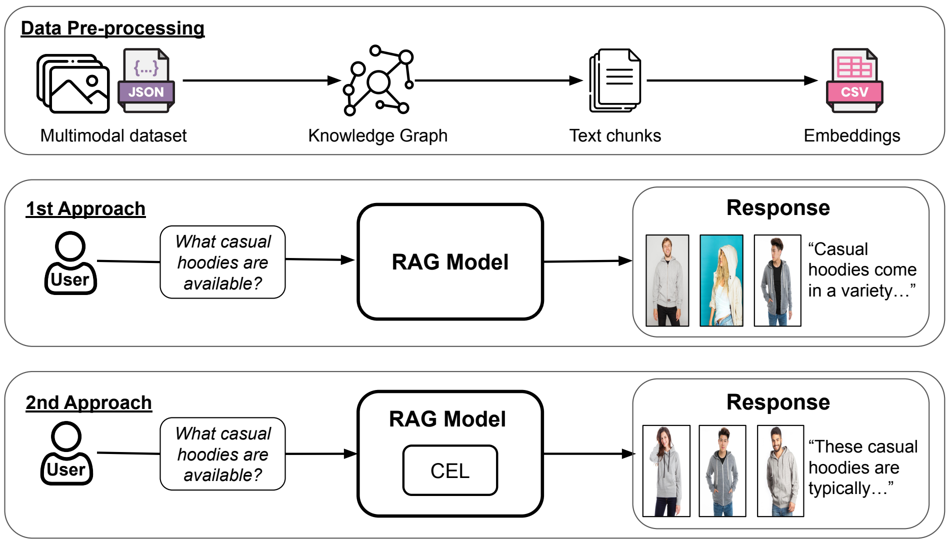

Fig. 1.1.: Overview of the work in this thesis. The pre-processing phase consist of data generation and transformation processes. A document chunk denotes all the textual information about an image. RAG denotes Retrieval-Augmented Generation. CEL denotes class expression learning.

1.2 Methodology

This work presents two novel multimodal RAG models for the task of image-based recommendation with generated textual description, using an RDF KG as the retrieval data type. We use a multimodal dataset about fashion apparel divided into image and text (human-annotation) parts. This data is originally not stored in a KG and during the initial steps of this work, we focus on generating a KG given the raw data. Throughout the process, multiple graph generations take place where the data is transformed to better-fit tasks in the RAG model such as information retrieval or CEL. Besides the information provided by the fashion dataset, the instances representing the images in the KG are further enriched with textual descriptions generated using a multimodal LLM. In the proposed RAG models, during the retrieval phase, we test

two retrieval models, an embedding-based retriever using cosine similarity score to rank the instances, and the BM25 retriever using the BM25 score to rank the instances. The response of the first multimodal RAG framework is a set of top k ranked images accompanied by a generated text that generally describes the images based on the top k documents. A document denotes all the information about an image retrieved from the KG.
We further extend this RAG framework by integrating CEL to re-rank the instances after the retrieval stage. Since a document can be mapped to an image, further in this word we will use the word "instance" to refer to an image-document pair. A portion of the retrieved instances is used to create a learning problem that is fed to a CEL algorithm to learn a classifying expression for the instances. Image 1.1 shows an overview of the work in this thesis.

Findings. Our experimental results show that our retrieval methods score better in the KG enriched with LLM-generated descriptions by measures of *mean reciprocal rank* (MRR) and Hits@K. We notice a drastic increase in performance when the retriever operates over the KG enriched with LLM-generated descriptions, highlighting here the embedding-based retriever which has an 81% increase in MRR and 65% in Hits@10 when used on the enriched KG. Furthermore, in the second RAG model where CEL is used to re-rank the results we show that its retrieval method slightly outperforms the embedding-based retrieval (i.e. retrieval without re-ranking) in terms of Hits@10 and Hits@20 while maintaining the same MRR. We test the performance of different CEL models and show that an F1 score of 0.77 and Accuracy of 0.78 is achieved by the best performing model in classifying clusters of instances.

1.3 Thesis Structure

The rest of this thesis is structured in 5 chapters as described below:

Chapter 2: Background

The background chapter contains the terminology and fundamental information for understanding the presented work. Concepts such as knowledge graph, description logics, OWL class expression learning, and retrieval-augmented generation are covered.

Chapter 3: Related work

In the related work chapter, the existing literature is reviewed to find related projects or studies. The used methodologies are identified in these studies and explained to identify gaps and limitations in the existing body of work.

Chapter 4: Methodology

The methodology chapter is the core chapter of this thesis, which describes the approach and implementation details behind the proposed RAG models. The chapter starts by explaining the knowledge graphs generation used in the RAG models and then continues with the stages of the RAG models, respectively with retrieval and then augmentation & generation.

Chapter 5: Evaluation

In this chapter, we evaluate the retrieval methods and class expression learning by initially describing the evaluation setup and then the experimental results.

Chapter 6: Summary & Discussion

In the final chapter, the main findings, contributions, and outcomes of this work are summarized while addressing limitations and directions for future work.

Background 2

Before we delve deeper into the proposed RAG models, we define some key notations used throughout the work. Firstly, in Section 2.1, the term *Knowledge Graph* is defined. Section 2.2 and Section 2.3 describe *Description Logics* and *Knowledge Base* respectively. In Section 2.4 we explain *Class Expression Learning* by giving some general knowledge about *Supervised Machine Learning* first. Section 2.5 gives an overview of *Retrieval-Augmented Generation* and lastly in Section 2.6 we give specifications about*Fashionpedia* dataset, an image-text dataset.

2.1 Knowledge Graphs

Hogan et al. [Hog+21a] define the knowledge graph (KG) as "*a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities*". The data stored in KGs can be used in a multiple domains of application such as recommendation system, fact-checking, machine learning, and knowledge prediction [Dem23]. The formal definition of a KG is often a set of triple $\mathcal{G} = \{(s, p, o)\}$ s.t. $(s, p, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ stands for the set of entities/nodes in the graph and $\mathcal{R}$ for the set of relations/edges [Dem23; Det+18]. A tuple (s, p, o) represents a triple in an RDF KG with s, p, and o denoting respectively a subject, a predicate, and an object where $s, o \in \mathcal{E}$ and $p \in \mathcal{R}$.
An RDF KG uses the RDF syntax to represent data in the form of subject-predicate-object triples. RDF Schema (RDFS) offers some schema capabilities but it's still not as expressive as OWL [LAS99].
OWL KGs are an extension of RDF KGs because OWL syntax is more expressive than RDF syntax in terms of representing more complex relationships, rules, and constraints on the data [Fer+23]. Therefore we can say that an RDF KG is a valid OWL KG but not the opposite. OWL introduces richer semantics like class hierarchy, property characteristics, restrictions, and more [Fer+23]. OWL syntax allows for the use of ontology reasoners like HermiT [MSH07] or Pellet [Sir+07] to infer new knowledge, verify consistency, or for classification tasks.

2.2 Description Logics

Description Logics (DLs) are a set of description formalities with differing expressiveness that are used for knowledge representation [BHS08a; Baa03]. One important aspect of DLs is that they represent the knowledge of a specific domain in a structured manner [Baa03]. Moreover, they are human-understandable, which helps us express learned concepts or class expressions as they can be translated to OWL [Dem23]. The basic description logic $\mathcal{ALC}$, is commonly used for reasoning service and therefore for class expression learning. Table 2.1 shows $\mathcal{ALC}$ syntax and semantics.

Construct	**Syntax**	**Semantics**
Atomic concept	A	$A^I \subseteq \Delta^I$
Role	r	$r^I \subseteq \Delta^I \times \Delta^I$
Top concept	$\top$	Δ^I
Bottom concept	$\bot$	$\emptyset$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Negation	$\neg C$	$\Delta^I \setminus C^I$
Existential restriction	$\exists r.C$	$\{x \mid \exists y.(x, y) \in r^I \wedge y \in C^I\}$
Universal restriction	$\forall r.C$	$\{x \mid \forall y.(x, y) \in r^I \text{implies } y \in C^I\}$

Tab. 2.1.: Syntax and semantics of $\mathcal{ALC}$ description logic.

2.3 Knowledge Base

The information represented by description logics is usually stored in a Knowledge Base (KB) which is formally defined as $\mathcal{K} = \{TBox, ABox\}$ [Dem23; LH10]. $TBox$ or the *terminological* part describes a set of rules that are applicable to the domain employed by the KB, whereas $ABox$ or the *assertional* part, describes concrete data assertion for individuals of the KB. An individual is a concrete instance of the KB. Baader et al. [BHS08b] define $ABox$ as a set of assertional axioms where such an axiom has the form: $x : C$ or $(x, y) : r$ where C is a concept in the specified DL (e.g. $\mathcal{ALC}$), r is a role and x, y are individuals of the KB. Other common notations used instead of the aforementioned ones are respectively $C(x)$ and $r(x, y)$. $TBox$ and $ABox$ are essential when it comes to reasoning services used to infer implicit or explicit knowledge from the KB. A KB can be translated into a KG and vice-versa [Dem23].

2.4 OWL Class Expression Learning

***Machine Learning* (ML)** is a subfield of *Artificial Intelligence* (AI) and generally defines the ability of a system to learn from training data and improve the system performance in a measurable manner [Leh10]. *Supervised machine learning* is a type of machine learning where an ML algorithm is trained on labeled data [Leh10]. The goal is to learn a mapping from inputs to outputs by finding patterns in the data. During training, the model makes predictions and adjusts its parameters to minimize the difference between its predictions and the actual labels, typically using algorithms like linear regression, decision trees, or neural networks [Leh10]. Once trained, the model can generalize and make accurate predictions on new, unseen data based on what it learned during training. Supervised learning is commonly used in tasks like classification, where the goal is to categorize data, and regression, where the goal is to predict continuous values [Leh10].
***OWL Concept Learning* (CL)** is a form of *Supervised ML* over RDF KGs and is a specific problem of *inductive learning* (also called *inductive reasoning*) which on its end is a subfield of symbolic, supervised ML [Leh10]. Based on the work of Lehmann [Leh10] we make the following claims in this paragraph. CL goal is to find a concept in the logic used to represent the knowledge (e.g. description logics) in such a way that it covers as many entities from the set of members, the so-called *positive examples* and covers as few entities from the set of non-member, or the so-called *negative examples*. The pair of these two sets is called a *learning problem*. The output of a concept learner is a logical description of a class expression otherwise known as a *hypothesis*. An example is *covered* by a hypothesis if that example is classified by that hypothesis.

Formal definition of CEL. We define CL in $\mathcal{ALC}$ description logic based on the definition in the work of Demir [Dem23] and Lehmann [Leh10]. Let $\mathcal{K} = \{TBox, ABox\}$ over $\mathcal{ALC}$ and given the set of positive E^+ and negative E^- examples such that $E^+, E^- \subset N_I \wedge E^+ \cap E^- \neq \emptyset$. CL should find a hypothesis H such that:

$$\forall p \in E^+, \forall n \in E^- \left(\mathcal{K} \models H(p)\right) \wedge \left(\mathcal{K} \not\models H(n)\right), \tag{2.1}$$

where H is a concept in $\mathcal{ALC}$. $\mathcal{K} \models H(p)$ means that the class membership $H(p)$ is a logical consequence of $\mathcal{K}$ and the opposite for $\mathcal{K} \not\models H(n)$, i.e. the class membership $H(n)$ is not a logical consequence of $\mathcal{K}$. CL and CEL are usually used interchangeably because a DL concept can be represented in OWL class expression [Dem23; Leh10;

Leh+11]. Therefore, for simplicity, we will be using the term CEL for the rest of this thesis.

2.5 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) is an approach that emerged as a solution to improve accuracy and reduce hallucinations of the LLM response by incorporating external context [Gao+24]. RAG improves the credibility of the generation, especially when dealing with domain-specific tasks [Kan+23] or when dealing with queries not covered by the LLM's training data, which introduces "hallucinations" in the response [Zha+23a; Gao+24]. RAG has three main stages: *Retrieval*, *Augmentation*, and *Generation*.

Naive RAG. Figure 2.1 shows one of the most typical RAG systems, known as *naive RAG* [Gao+24] or the "retrieve-then-read" framework [Ma+23]. In this paragraph, we describe the naive RAG based on the work of Gao et al. [Gao+24]. According to their study, usually, the systems include an indexing stage during which, data is extracted from various sources, transformed into text of the same format, gets separated into chunks which are encoded in a vector space using an embedding model and stored in a vector database. Having that done in advance, the system can expect the user's query and go through the retrieval phase. During the retrieval phase, the query is encoded into a vector space using the same embedding model as the one used for indexing, and similarity scores, like cosine similarity, are then calculated between the query embedding and the chunks embeddings that were created during indexing. The top k results with the highest similarity score are then retrieved to continue with the next stage. Finally, the query and the external information gathered from the retrieval phase are merged in their textual form, in order to create an enriched prompt that will be used to generate the final result. In the next paragraphs, we will cover the most relevant details and terminology about each stage.

2.5.1 Retrieval

Gao et al. [Gao+24] have identified in their survey the key factors for the retrieval stage, starting with the *retrieval source*. The data of the retrieval source can be *unstructured data* such as text, *semi-structured data* such as PDF, which can contain

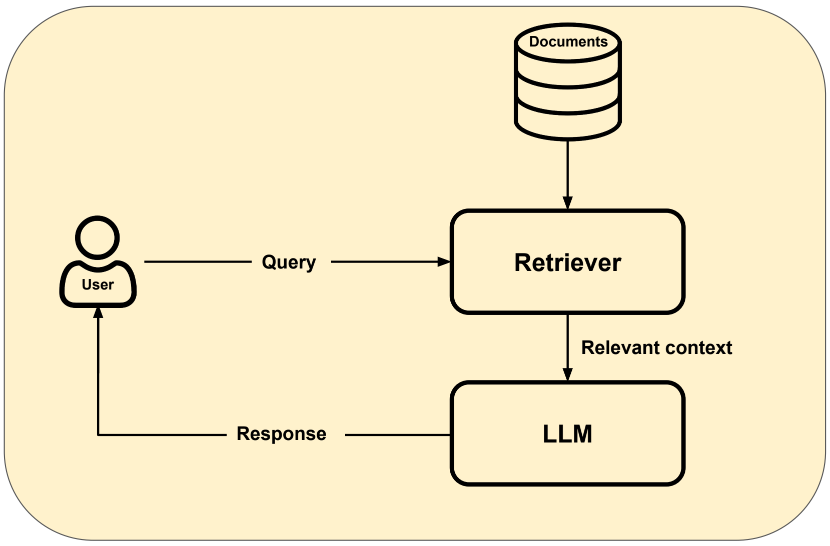

Fig. 2.1.: Workflow of the naive RAG model.

tables as well; and *structured data* such as KGs [Gao+24]. Furthermore, LLMs can be utilized to generate content [Sun+22; Yu+22; Wan+24]. *Retrival granularity* is another important factor belonging to retrieval source [Gao+24]. To avoid redundant information and to guarantee semantic integrity, a trade-off should be found for the retrieval granularity [Yu+23; Shi+23].
For the *indexing* stage, which is part of the retrieval stage, Gao et al. [Gao+24] have identified a few optimizations, starting with *chunking strategy*, which mainly consists of deciding the size of the chunks during indexing, where larger chunks have more context but introduce more noise and smaller chunks may not have sufficient context, leading to hallucinations. Another optimization is *metadata attachments*, which can help during the retrieval process by enabling filtering of data. Indexing can be established in a structured manner, helping RAG systems on time and relevance efficiency [Gao+24].
Besides indexing optimization, RAG systems can perform *query optimization*, which includes *query expansion* [KKH24] where the initial query is expanded into multiple queries, *query transformation/rewriting* [Ma+23; Pen+24] where the user's query is transformed into a more machine-understandable one, and *query routing* where the query undergoes a metadata or semantic routing to narrow down the search scope and increase time efficiency [Gao+24].

2.5.2 Generation

This stage consist of adjustments to the retrieved information or adjustments to the LLM before generating the final results. In the context adjustment part, there are two general approaches, *reranking* and *context selection/compression* [Gao+24]. Reranking has to do with rearranging the document chunks in such a way that the most relevant ones come first. This can be achieved through predefined rules to rank them. Context compression is the process of reducing the amount of data passed together with the final prompt in order to reduce the noise [Gao+24]. Some approaches include using *small language models* (SLMs) to remove unnecessary tokens [Jia+23a; AWR22]. There are also some adjustments or fine-tuning that we can do to an LLM. That includes adding data about a specific domain where the LLM is not trained enough, making the LLM understand specific formats, or using reinforcement learning to adapt to the user's preferences [Gao+24].

2.5.3 Augmentation

Having just a singular retrieval step may not be sufficient all the time, because results can be inconsistent or have too much noise [Gao+24]. To overcome that, RAG systems incorporate additional retrieval steps during the augmentation stage, usually including a judging mechanism that decides when to stop the retrieval [Gao+24].

Augmentation optimizations. Based on the survey conducted by Gao et al. [Gao+24], in this paragraph, we describe the possible augmentation optimizations identified by them: *iterative retrieval*, *recursive retrieval*, and *adaptive retrieval*. A graphical visualization of the workflow for each augmentation type is given in Figure 2.2. In the iterative retrieval case, besides the base flow of generation after retrieval, there is an additional process, the judging process, where the generated result is evaluated based on its robustness. The RAG system may decide to go through the retrieval phase again for additional context, therefore "iterating" over the retrieval-generation process. One downside is that over multiple iterations, a semantic gap can be created between the query and the generated results.
The recursive retrieval consists of refining the generated results in a recursive manner. There is still a judging process after generation, which will break the problem into sub-problems and continue to try to find better results through recursive iterations. In iterative and recursive optimizations, the LLM does not contribute to the judging process. But another approach [Asa+23; Jia+23b] has been introduced where the

LLM is used to decide whether there is a need to retrieve external context or not. This is called the *adaptive retrieval*, and its execution pipeline is similar to the one of recursive retrieval but with the LLM performing the judgment.

Our augmentation approach. In our first proposed RAG model we only perform a singular retrieval step. We combine all documents together and use them to generate a textual description for the recommended images. Whereas in the second RAG model we use CEL to re-rank the classified instances based on a generated summary from the initially retrieved documents. This is an experimental augmentation approach that does not include iteration or recursion.

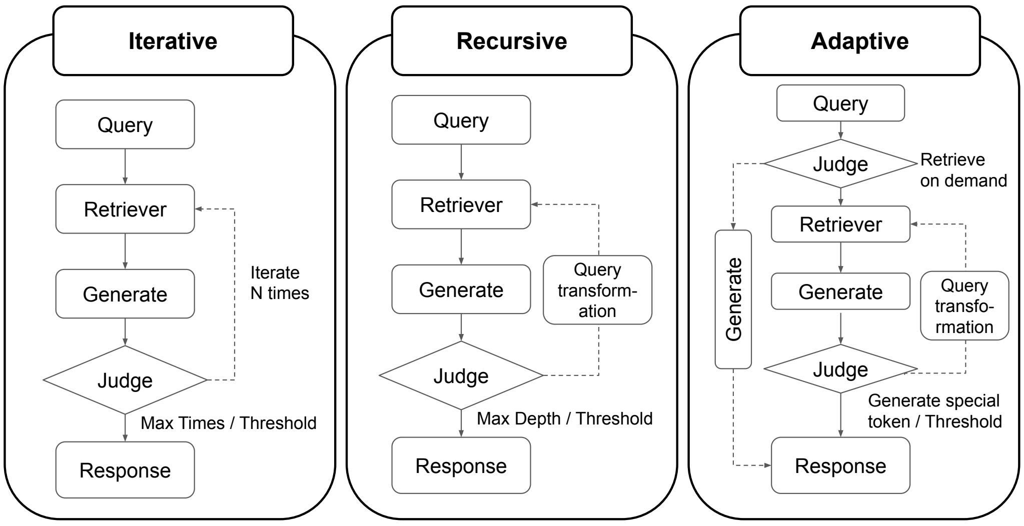

Fig. 2.2.: Three types of retrieval augmentation processes, recreated from Gao et al. [Gao+24].

2.5.4 Multimodal Retrieval-Augmented Generation

Multimodal RAG is an extension of the traditional RAG framework, created specifically to handle multiple types of data modalities like image, video, audio, and code for context to generate a response [Zha+23b; Zha24]. Using multiple data modalities in a RAG framework leads to a richer and more contextually informed generation process [Zha+23b].
An *image-based* RAG model processes visual data to augment textual generation. Usually in such models, context is extracted from domain-of-interest images with the help of embedding models like CLIP [Rad+21], and relevant context to the

query can be used to support the generation phase of the RAG system [Zha+23b; Che+22].
Audio and video modalities, although not so commonly adapted because of a lack of training data on audio-text and video-text pairs, are approached in a similar fashion to image modality [Zha+23b]. In the case of the audio, using an audio embedding model, textual content can be retrieved from some training data and may be further processed, like being fed to a pre-trained language model (PLM) [KSO23], before it can be used as aiding content in the generation phase [Zha+23b; Xu+19]. As for the video-based models, frames of the video accompanied by the audio are processed, and a multimodal embedding model is used to find relevant context for the given query [Zha+23b].
Code-based models in multimodal RAG systems, use code snippets as data input to generate explanations, solutions, or optimized version of the code based on the user's query [Zha+23b; Lu+22; Zho+22].

2.6 Image Dataset: Fashionpedia

For this work we use **Fashionpedia**, a multimodal dataset of fashion apparel designs where each image is annotated with descriptive terms. Fashionpedia consists of two parts: the ontology, stored in *JSON* format, and the images part, which is a folder containing images in *jpg* format Jia et al. [Jia+20]. There are 48,825 images in total that are described using 27 main apparel categories, 19 apparel parts, 294 fine-grained attributes, and their relationships. Jia et al. [Jia+20] have paid special focus to the instance segmentation and categorization of visual attributes. Said simply, segments in the images are selected in a way that they can be assigned different categories and can serve as a self-contained entity based on the apparel in the image.

2.6.1 Fashionpedia Generation

We describe the Fashionpedia ontology generation process and its specifications based on the work of Jia et al. [Jia+20]. A Fashionpedia object is similar to a Wikidata "item" [VK14] or an "object" in COCO [Lin+14]. An image in the dataset contains one or multiple main garments, and each main garment is further divided into its garment parts. In the ontology, such an image is represented by multiple annotations, and each annotation has a category that represents a garment or a garment part. In Figure 2.3 you can see such annotations shown in white color boxes.

An example of a main garment is "jacket" with the garment parts being "collars", "sleeves", "pockets", "buttons", etc. Each of them is considered an annotation for the given image. Garments (annotations) can contain attributes, called *fine-grained attributes*. For example, the "jacket" garment can contain the silhouette attribute with the value "symmetrical". In Figure 2.3, attributes are shown in black boxes.

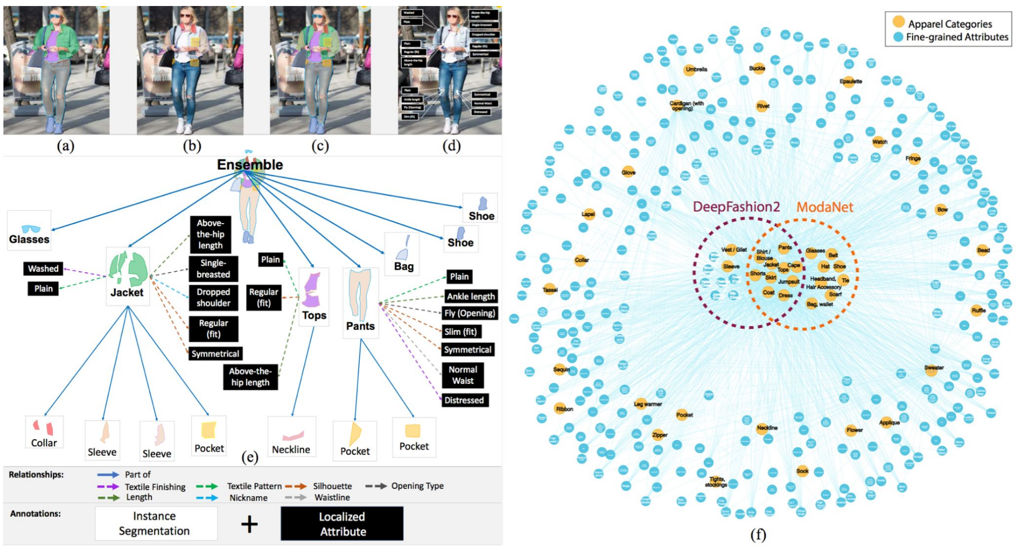

Fig. 2.3.: Illustration of Fashionpedia dataset and ontology. (a) main garment masks; (b) garment part masks; (c) both main garment and garment part masks; (d) fine-grained apparel attributes; (e) an exploded view of the annotation diagram: the image is annotated with both instance segmentation masks (white boxes) and per-mask fine-grained attributes (black boxes); (f) visualization of the Fashionpedia ontology: we created Fashionpedia ontology and separate the concept of categories (yellow nodes) and attributes (blue nodes) in fashion. It covers pre-defined garment categories used by both Deepfashion2 and ModaNet. Mapping with DeepFashion2 also shows the versatility of using attributes and categories. Caption an the figure taken from the work of Jia et al. [Jia+20]

Fashionpedia contains three types of relationships: (1) outfits (images) to main garments/garment parts (annotations), (2) main garments/garment parts to attributes, and (3) is-instance-of type of relationship within garments, garment parts, or attributes. The images (48,825 in total) were taken and filtered from free-license photo websites: Unsplash, Burst by Shopify, Freestocks, Kaboompics, and Pexels. The focus was on scenes diversity and having visible apparel appropriate for the annotation task. The dataset is divided into training (45,623 images), validation (1158 images), and test (2044 images) sets. In order to annotate the images, crowd workers and experts were assigned to specific tasks. 28 crowd workers, who were trained by the experts, annotated segmentation masks with apparel objects. This process was supervised by the authors and a supervisor. 15 fashion experts were given the task to annotate the fine-grained attributes that are used for the annotated segmentation masks.

Related Work 3

In 2020, Lewis et al. [Lew+20b] introduced RAG for the first time as a method to retrieve relevant documents from data source like Wikipedia and integrate them with a generative model like BART [Lew+20a] to reduce the model's "hallucinations" and provide a more accurate response. Since the release or GPT-3 by Brown et al. [Bro+20], there has been a rapid development in RAG systems that supports the generative models divided into three main branches: pre-training, fine-tuning, and inference [Gao+24]. This chapter describes briefly some of the most relevant works, specifically in RAG, related to this thesis and highlights the similarities with them. To identify related work, we used *Google Scholar* as the main source, searching via keywords in the research field and then snowballing through cited studies.

3.1 Retrieval-Augmented Generation over Knowledge Graphs

GraphRAG is a RAG model for *query-focused summarization* (QFS) tasks that enables sensemaking over text content by constructing a hierarchical KG emphasizing the focus on entities, relationships, and claims in the text [Edg+24]. Edge et al. [Edg+24] have proposed a novel framework pipeline where they create a KG by using LLM prompts that extract various elements of a graph index given the text chunks. They further use LLM for element summarization, which helps to create a hierarchical graph where communities are detected and can be summarized. The summaries can then be used to answer community and global questions. Edge et al. [Edg+24] show that their framework achieves superior performance in comprehensiveness over the naive RAG. Among others, GraphRAG displays the novelty of using summarization via LLMs in RAG frameworks to improve their performance, which is something that we practice in this work.

KnowledGPT [Wan+23b] is a RAG system designed to improve the capabilities of LLMs by incorporating external knowledge bases that efficiently retrieve and store knowledge. The system adopts a "program of thought" approach by generating Python code to query external KBs [Wan+23b], in contrast to our approach, which is limited to the Fashonpedia dataset. KnowledgeGPT populates a personalized KB (PKB) based on user queries, and similarly to our approach, the information is stored in an entity-aspect manner as a variation of a triple where the object is a long piece of text adhering to the symbolic memory of LLMs [Wan+23b]. Although KnowledGPT presents significant advancements in RAG approaches over structured data in limiting the gaps in LLM capabilities regarding knowledge faithfulness, its retrieval system is based on a single-round code generation [Wan+23b].

G-Retriever [He+24] proposes a new RAG system for *Question Answering* (QA) on textual graphs. The system supports different types of graphs, including KGs for which the authors don't provide semantics specifications. G-Retriever focuses on QA on textual graphs in the form of a conversational interface to ask questions in a graph [He+24]. The findings in the study of He et al. [He+24] show that retrieval on graphs can mitigate hallucinations in the answer of LLMs. In the study, authors test their framework in the GraphQA benchmark [Bal+21], which is created by leveraging LLMs similarly to the approach we follow to generate our benchmark, but the benchmark setups differentiate in the structure, which is adapted to the systems tested. Although G-Retriever supports *scene graphs*, which contain image data modality, they consider only the additional textual data provided for the images in JSON files and don't use a multimodal embedding model or a multimodal LLM to capture information about the images [He+24].

While GraphRAG, KnowledGPT and G-Retriever present novel approaches and share similarities with parts of this work, we propose a RAG model where we introduce CEL during the retrieval process for the first time.

3.2 Retrieval-Augmented Methods

In contrast to the conventional retrieve-then-read approach, Yu et al. [Yu+22] propose **Generate-Then-Read (GenRead)**, a new workflow for knowledge-intensive tasks by generating contextual documents with LLMs and then using the documents to produce the final answer. This is similar to our approach, but we use the LLM-generated context as an addition to the existing documents. GenRead framework works well in open-domain question answering, and Yu et al. [Yu+22] show that generated documents contain more relevant answers than the retrieved documents. They suggest that the generated documents can be easily combined with the retrieval techniques for further improvements.

Hypothetical Document Embeddings (HyDE) [Gao+22] is a zero-shot dense retrieval framework that combines instruction-following language models with contrastive encoders. By giving instructions to an LLM, the system generates a hypothetical document for a given query, and using the contrastive encoder, it converts the generated document to an embedding vector, which is used to retrieve real documents according to embedding similarities. This method demonstrates significant improvements over state-of-the-art unsupervised dense retrieval models on various tasks, including, among others, question answering. In our RAG approach we follow a similar re-ranking strategy but not in a zero-shot setting. In our approach we don't use the user's query to generate a hypothetical document, but the retrieved documents from the classified instances after CEL are used to generate a summary that we use to re-rank the retrieved documents by comparing their embedding vectors.

3.3 Multimodal Retrieval-Augmented Generation

In this thesis we focus on image-text multimodality. In the task of *visual question answering* (VQA), **PICa** [Yan+22] and **KAT** [Gui+21] are two studies that use an LLM to extract information for a given image, showing improvements over the retrieval process [Zha+23b]. **MuRAG** [Che+22] presents a novel approach where, besides the text content, they consider the image modality as visual tokens to support the generation process. Given they outperform models that use only text content, Chen et al. [Che+22] encourage the use of multiple modalities in retrieval-augmented models. Multimodal RAG systems that use image modality are mainly focused on VQA, image captioning, visually grounded dialogue, and text generation [Zha+23b]. In our RAG approach, we use image modality via a multimodal LLM to enrich the existing image-text content in order to enhance the retrieval process and text generation process.

Methodology 4

This work presents two novel multimodal RAG approaches over KGs for the task of image-based recommendation with generated textual description. The KGs are created in a pre-processing phase using Fashonpedia dataset as the data source. The first proposed RAG model or what we call the baseline model, is similar to the naive RAG where there is the retrieval model and the LLM which is used for generation. We experiment with two different retrieval models: BM25-based and embedding-based. Each of them includes an indexing phase where data from the KG is divided into chunks and further processed, like encoded, when using embedding-based retrieval. We further extend the baseline model by introducing owl class expression learning during the augmentation phase, creating this way a new RAG model which we refer as the second RAG model or framework. The code [1] for the whole project is publicly available on GitHub.

First in Section 4.1 we go through the details behind the steps of generating the KGs used in the RAG models. Then, each stage of the RAG models is described respectively in Section 4.2 and Section 4.3.

4.1 Multimodal Knowledge Graph Generation

The generation of a KG is a fundamental process in this work because it defines what and how information is stored, which will later be used in the RAG models. Fashonpedia ontology is used to create an RDF KG. **For this work we consider only RDF KG and we omit the "RDF" part when referring to RDF KG for simplicity**. This created graph is a direct mapping of the Fashonpida ontology, preserving its structure. This structure is resolved around annotations, which are parts of an image. Annotations are describe using categories and attributes which are identified by and identification number (ID). Identifying limitations in this structure we generate a second KG where the structure is resolved around images as the main component, and the data is summarized together and directly linked to the respective image, making it accessible in a single hop. Although this KG is ready to be used in the RAG system, it lacks the multimodality aspect. Therefore a third KG is created

[1] `https://github.com/dice-group/MRAG-KG`

that is based on the second one, but here a multimodal LLM is utilized to generate a description for each image given as input the image in base64 format. This description is referred to as the *LLM-generated description* throughout the rest of the paper. A new data property is declared, which is used to assert the LLM description to image individuals. The rest of the subsections describe the details behind each KG generation.

4.1.1 First Knowledge Graph Generation: From JSON to RDF/XML

To generate the KG, only the ontology part of the Fashopedia is necessary. The information is stored in a JSON file publicly available. The full data can be found in *fashionpedia* [2] GitHub repository under the downloading link labeled '`instances_attributes_train2020`'. In the repository, the structure of the data in the JSON file is also displayed.
In a closer look, the structure of the data in the JSON file can be easily represented in an RDF KG. The task includes identifying the classes and the relationships, those represented by object properties and those represented by data properties. The main datatype entities, i.e., *info, category, attribute, image, annotation, license*, can be represented by OWL classes. The instances of each class will be represented as instances of the same classes in the KG. In the fashionpedia dataset, the instances are identified by a unique *ID*, and a relationship between two instances is tracked by using their IDs. The most fitting relation for that scenario used in a KG is achieved via *object properties*, which represent relations between two individuals of the KG.
So far we can represent the datatype entities as classes, the instances as named individuals, and the relationships between instances as object properties. The only thing that is left is the literal information of each instance. Since we have to do with literal data, then the most fitting relation to represent that in a KG would be *data properties*, which connect an individual with a literal value. For the first generated KG, the goal was to map the data from the JSON format to the RDF/XML format, although the most redundant metadata is excluded from the mapping.

Figure 4.1 shows a subgraph of the first KG focused on a single individual of type "annotation". Instances of type "annotation" represent a garment or a garment part in the image.

Table 4.1, Table 4.2 and Table 4.3 show the classes and properties of the first-generated KG.

[2] `https://github.com/cvdfoundation/fashionpedia`

Classes
Category
Attribute
Image
License
Annotation

Tab. 4.1.: Classes of the first knowledge graph.

Object Properties	Domain	Range
hasCategory	Annotation	Category
hasAttribute	Annotation	Attribute
hasImage	Annotation	Image
hasLicense	Image	License

Tab. 4.2.: Object properties of the first knowledge graph together with the domain and range of each object property.

Data Properties	Domain	Range
hasFileName	Image	xsd:string
hasName	Category ∪ Attribute ∪ License	xsd:string
hasSupercategory	Category ∪ Attribute ∪ License	xsd:string
hasLevel	Category ∪ Attribute ∪ License	xsd:integer
hasTaxonomy	Category ∪ Attribute ∪ License	xsd:string
hasWidth	Image	xsd:integer
hasHeight	Image	xsd:integer
hasTimeCaptured	Image	xsd:string
hasOriginalUrl	Image	xsd:string
isStatic	Image	xsd:integer
hasKaggleId	Image	xsd:string
hasUrl	License	xsd:string
hasArea	Annotation	xsd:integer
isCrowd	Annotation	xsd:integer

Tab. 4.3.: Data properties of the first knowledge graph together with the domain and range of each data property.

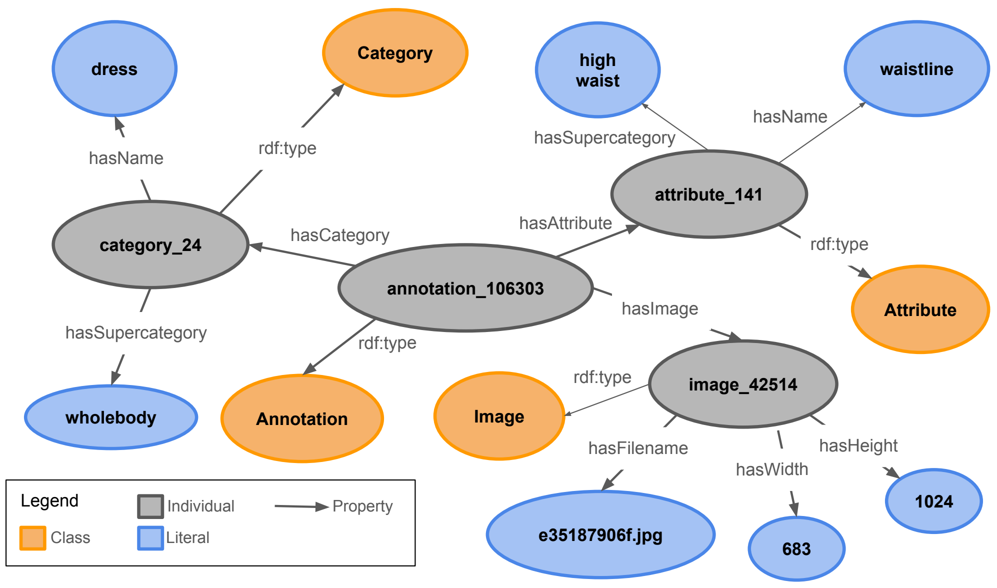

Fig. 4.1.: A subgraph of the first knowledge graph.

4.1.2 Second Knowledge Graph Generation: Subgraph Summarization

The first KG consists of a direct mapping of the data from the JSON format to the XML/RDF format. There are a few points that can be optimized to make it easier and more efficient to use the data in the RAG system: (1) In the first KG, literal data can be retrived using multi-hop reasoning, making the retrieval process more difficult and computationally demanding. (2) The information contained in the literal data consists of keywords that may cover less context, leading to weak embedding similarity between the user's query and the data. (3) The recommendation system is image-based, meaning that it recommends images, but the main entity in the first KG is "Annotation" which is just part of an image.

To ease the retrieval process, it's necessary to address the aforementioned points. Therefore, to address point 2, the followed approach is to summarize the information for an annotation into a single sentence highlighting key words. This way, literal data contains more context which can be compared with the user's query. To address point 1 and 3, the approach is to built the KG around images as the main individuals by using direct relation to connect the literal data with an individual of type "*Image*". This approach makes the data easily accessible. By combining both solutions together, the concrete implementation is to generate a KG where the main individuals are individuals of type "*Image*" and each of these individuals contains the information about each annotation belonging to that image in the form of a data property which

connects that individual with the sentence describing the annotation as the literal value of that property.
Beside the information for each annotation, an individual of type "*Image*" also contain other metadata such as the filename of the image, as well as its width and height. Every data that is not related with the the apparel in the image is considered redundant and is not included in the KG.

A subgraph of the second KG is given in Figure 4.2. Overall, the complexity has decreased, we no longer need to access multi-hop data and the descriptive information of an annotation is stored together in a single data property. This KG contains only one class, that being the class *Image*, and 4 data properties: (1) *hasWidth*, (2) *hasHeight*, (3) *hasFileName*, and (4) *hasDescription*. The first 3 data properties are the same as in the first KG, whereas the last one is the newly introduced property that contains the descriptive sentence for each annotation of an image. The sentence or description has a predefined structure. The process of creating it involves using the first-generated KG: It starts by iterating over the annotations that have an *hasImage* relation to the current image. From the annotation we retrieve the category and the supercategory as string values. With that information, we create the first part of the descriptive sentence, which is: "*Contains* {*cat_name*} *from supercategory* {*cat_supercat*}".
The next part of the descriptive sentence is about attributes of the annotation. Since there can be more than one attribute for a given annotation, we iterate over them and add them in the sentence in a way that makes sense in natural language. In case an annotation contains attributes, the above sentence continues with the following string " *with the following attributes:* ". Then, for each attribute, its name and its supercategory are retrieved and added to the sentence as follows: "{*attr_name*} *of supercategory* '{*attr_supercat*}', ".

An example of a full descriptive sentence would look like the following:

> *"Contains jacket from supercategory 'upperbody' with the following attributes: no non-textile material of supercategory 'non-textile material type', tight (fit) of supercategory 'silhouette'."*

4.1.3 Third Knowledge Graph: Enrichment with Multimodal LLM-generated Context

The descriptions of an image in the second KG follow a strict predefined structure and do not contain any additional information tailored to the image. To go beyond

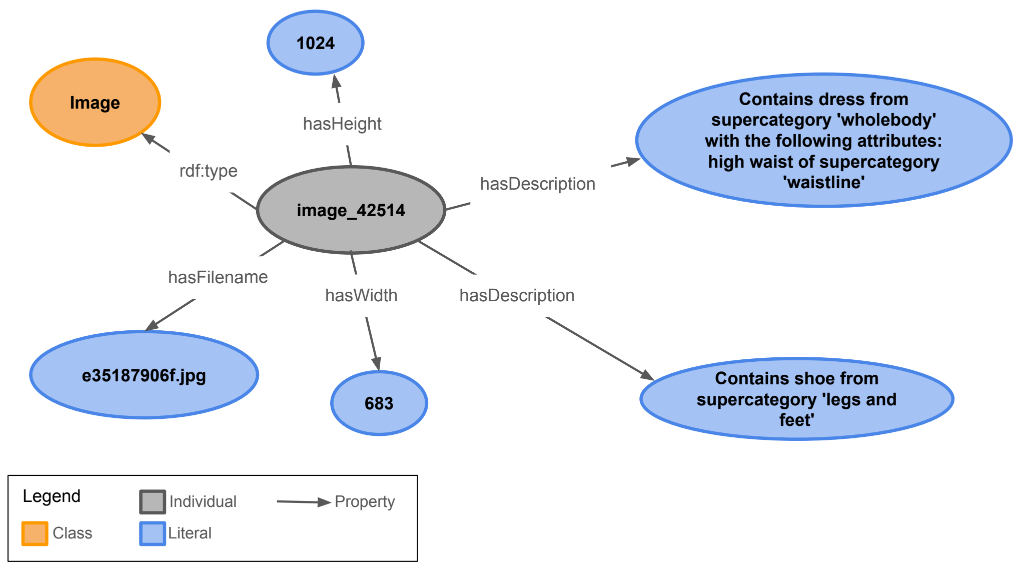

Image
1024
hasHeight
rdf:type
image_42514
hasDescription
Contains dress from supercategory 'wholebody' with the following attributes: high waist of supercategory 'waistline'
hasFilename
hasWidth
hasDescription
e35187906f.jpg
683
Contains shoe from supercategory 'legs and feet'
Legend
Individual
Property
Class
Literal

Fig. 4.2.: A subgraph of the second knowledge graph. The descriptive information of an annotation is stored as a string literal vlaue using *hasDescription* data property. Annotation shown in Figure 4.1, is used here for comparison.

the strict boundaries of that structure and fill the missing gap of not having a tailored description, a multimodal LLM is employed to describe the image. This approach allows for enrichment with additional context that is not part of the original Fashionpedia dataset.

A third KG is created, which is a copy of the second KG, where a new data property is added, namely "*hasLLMDescription*". The domain and the range of this property are the same as "*hasDescriptio*". A multimodal LLM is used to generate a short description, given each image in binary format. In order to provide a compact result and reduce out-of-context information, multiple different queries were tested. For example, during the experiments, the LLM also described the background; therefore, it was necessary to explicitly mention in the query not to do that.

Query and the other parameters used for the LLM:

> *"You are a fashion expert. Your task is to give a short description of the apparel provided in the attached image. You should focus only on the apparel presented to you. Don't describe the background."*

- temperature = 0.1
- seed = 1

Figure 4.3 shows the same example as from Figure 4.2 but now enriched with the LLM-generated description.

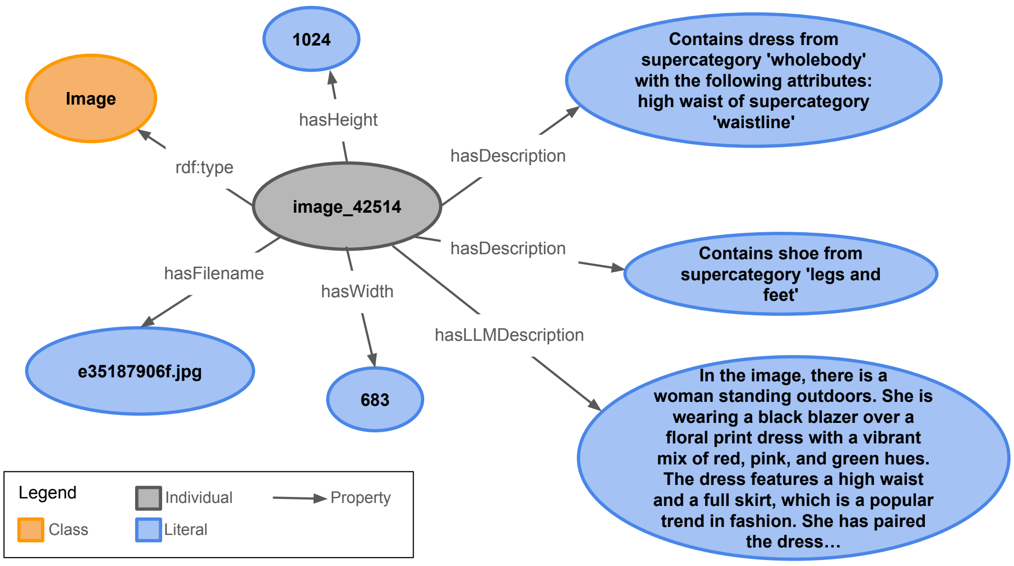
1024
Image
hasHeight
Contains dress from supercategory 'wholebody' with the following attributes: high waist of supercategory 'waistline'
rdf:type
hasDescription
image_42514
hasDescription
Contains shoe from supercategory 'legs and feet'
hasFilename
hasWidth
hasLLMDescription
e35187906f.jpg
683
In the image, there is a woman standing outdoors. She is wearing a black blazer over a floral print dress with a vibrant mix of red, pink, and green hues. The dress features a high waist and a full skirt, which is a popular trend in fashion. She has paired the dress...
Legend
Individual
Property
Class
Literal

Fig. 4.3.: A subgraph of the third knowledge graph where the LLM-generated description is added to each individual.

4.2 Multimodal Retrieval

During the retrieval phase of the RAG system, the goal is to retrieve relevant data to the user's query. To increase the credibility of this work, two different retrieval models are implemented and later evaluated: *Okapi BM25 retriever* and an embedding-based scoring model using *Mistral* [3]. The retrieval process includes an indexing phase where the data in the KG is divided into chunks, and for the retrieval model that uses embedding, it is further encoded using an embedding model. In this section, we go through the approach and implementation details about the retrieval models.

4.2.1 Indexing

Indexing plays a critical role in the retrieval phase, especially when retrieving information based on metrics such as cosine similarity of encoded text [Sab17]. Is an essential step to increase the efficiency of the retriever.

[3]`https://mistral.ai/`

Indexing in Embedding-based Retriever

To embed the description of each image stored in the KG, a word embedding model is required. For this task, a *Mistral-7B-v0.1* embedding model is used, specifically *E5-mistral-7b-instruct* [Wan+22; Wan+23a] which has 32 layers and an embedding size of 4096 with 7.11 billion parameters.
The process of generating and storing embeddings is as follows: given an individual representing an image in the KG, each of its *hasdescription* properties is combined into one string and vectorized by the embedding model. These generated embeddings are then stored in a CSV file to later be used during retrieval. To uniquely identify each embedding vector, the IRI of the individual is also stored in the CSV file as a way to identify to which image the embedding belongs. In the case of the third KG, a document includes the LLM-generated description. The LLM-generated description (stored using the *hasLLMdescription* data property) is combined with the rest of the descriptions of a particular individual before the embedding is generated for that individual's description.

Indexing in BM25 Retriever

In the case of the BM25 retrieval model, there is no embedding generation because BM25 is a traditional probabilistic retrieval model. However, a document preparation phase happens before proceeding with the actual scoring algorithm. We go through every description of an image and merge it together into a single text to generate the document for each image. The list of all documents is transformed into numerical data using a *Term Frequency - Inverse Document Frequency* (TF-IDF) vectorizer. In code implementation, this would be the class `TfidfVectorizer` of *scikit-learn* [4] library version 1.5.2. The TF-IDF score of a term t in a document d, which is an element of the set of all documents D, is calculated as follows:

$$\mathrm{tfidf}_{t,d,D} = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_{t,D}. \tag{4.1}$$

For term frequency (tf), we are using raw count as the weighting schema to keep it simple and easier to calculate the BM25 score. Therefore, $\mathrm{tf}_{t,d}$ is equal to the number of times term t appears in document d.

$\mathrm{idf}_{t,D}$ is the inverse document frequency calculated as:

[4] `https://scikit-learn.org/stable/`

$$\mathrm{idf}_{t,D} = \log \frac{N}{\mathrm{df}_{t,D}}, \tag{4.2}$$

where N is the total number of documents $N = |D|$ and $\mathrm{df}_{t,D} = |\{d : d \in D \text{ and } t \in d\}|$, which is the number of documents where the term t is present, or in other words, where $\mathrm{tf}_{t,d} \neq 0$.

All the data is stored in a two-dimensional *numpy* [5] arrays and ready to use in the BM25 retriever. We use version 1.26.4 of numpy library.

4.2.2 Ranking Scores

BM25

Given a given query Q, the BM25 retriever uses BM25 score to rank the finite amount of documents $d \in D$. The BM25 score is measured using the definition from Manning et al. [MRS08]:

$$\mathrm{RSV}_d = \sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_{t,D}} \right] \cdot \frac{(k_1 + 1)\, \mathrm{tf}_{td}}{k_1 \left((1 - b) + b \times \left(\frac{L_d}{L_{\text{ave}}} \right) \right) + \mathrm{tf}_{td}}, \tag{4.3}$$

where RSV stands for "*retrieval status value*", and it is what we are referring to as the BM25 score. Left factor is the sum of *idf* score for each term of the query q. On the right factor we have k_1 and b, which are free parameters. In this thesis we are going with the usual values where $k_1 = 1.6$ ($k_1 \in [1.2, 2.0]$) and $b = 0.75$. L_d is the length of document d, and L_{ave} is the average document length in D.

Cosine Similarity

Since BM25 retriever is a probabilistic retrieval model, this thesis also considers a word embedding model that captures the semantics of the given text. To rank documents for a given query using the embedding-based retrieval model, cosine similarity is measured between the vectorized query and each vectorized document. Based on Rahutomo et al. [RKA+12] and Salton and Buckley [SB88], after we generated the embeddings for the documents and the query, each document can be described as a vector $\vec{d} = (w_{d0}, w_{d1}, w_{dk})$ and the query as a vector $\vec{q} = (w_{q0}, w_{q1}, w_{qk})$

[5] `https://numpy.org/`

where where w_{di} and w_{qi} $(0 < i < k)$ are float numbers. The cosine similarity is then measured as follows:

$$\text{Sim}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{||\vec{q}||||\vec{d}||} = \frac{\sum_{k=1}^{n} w_{qk} \times w_{dk}}{\sqrt{\sum_{k=1}^{n}(w_{qk})^2} \cdot \sqrt{\sum_{k=1}^{n}(w_{dk})^2}}, \tag{4.4}$$

where n is the length of the vector. While ranking the data, the vectors are stored in *numpy* arrays, and the same library is used to calculate the cosine similarity. The array is then sorted in descending order, where the instances with the highest score can be easily retrieved.

4.3 Augmentation and Generation

The output of a retrieval model are top k ranked instances that are decided to be the most relevant ones to the given query. The next steps left in this RAG approach are the augmentation and generation phases.
The full process is shown in Figure 4.4 and goes as follows: The top k retrieved images are shown to the user as part of the final response. The images are accompanied by a generated textual description. To generate the description, every document for the top k images that is provided by the retriever is concatenated together into a single text. This text combined with the user's query is used to construct a prompt that tells the LLM to generate the final textual response, which accompanies the selected images.

The prompt used for the final result is formed as follows:

- System content: "*You are an apparel-loving AI, and your focus is to give information about apparel.*"
- User content: "*You should find similar points in the assisting information provided to you and present them in a short paragraph tailored to the following query: '<query>'*"
- Assisting content: "*<documents>*"

Example. During experiments, as an example of the textual description for a given query, "What are some midi length lower body pants ?", the system responded with:

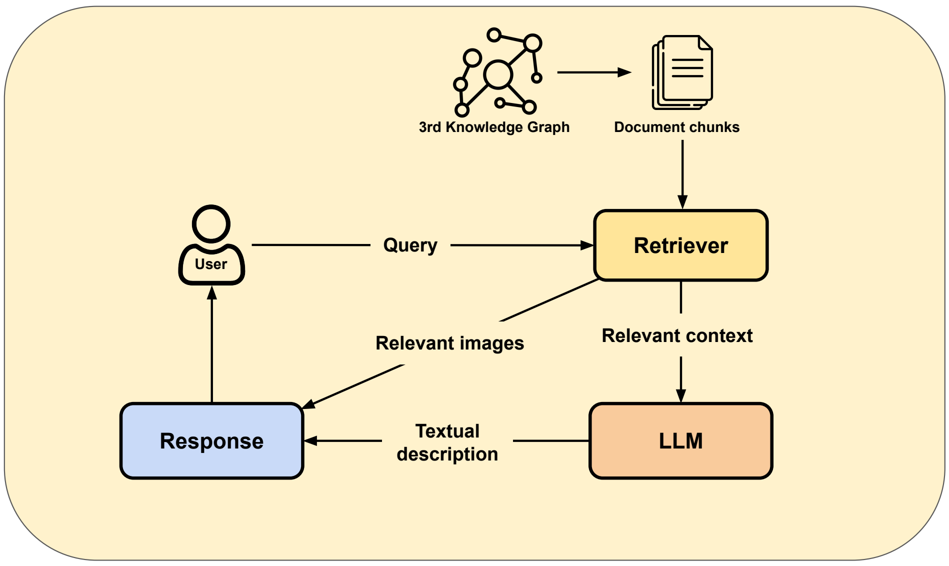

Fig. 4.4.: The workflow of the first multimodal RAG model, where enriched KG denotes the KG enriched with LLM-generated description. The yellow box indicates a model/component implemented by us, the orange box indicates an external model, and the blue box indicates the output of the system.

"*Midi length lower body pants are a versatile and stylish option for those looking for a balance between the length of shorts and full-length pants. These pants typically fall around the mid-calf, making them perfect for transitional seasons. The examples provided include a pair of high-waisted, wide-leg trousers in a smooth black fabric, which could be suitable for a more formal or business-casual setting. Another option is a pair of khaki pants with a straight cut and a flat front, made from a lightweight cotton blend, ideal for a casual, everyday look. These styles feature a midi length, providing a comfortable and chic alternative to traditional full-length pants.*"

For this example, the embedding-based retriever was used.

4.3.1 Integrating OWL Class Expression Learning

As an extension of the proposed RAG framework, we can utilize CEL to further augment the retrieved data. The full workflow is shown in Figure 4.6.

Optimizing the Second Knowledge Graph for CEL

In the second KG, the images are described using data properties with literal values containing the description. This KG is optimal for our retrievers but not for utilizing class expression learning in DLs. There are 2 reasons for that. (1) Individuals in the second KG contain only data properties pointing to string literals and specific string literal values, like the image descriptions, which are limited in DLs because DLs primarily treat data properties through constraints that don't take into consideration string values [MR08; Pat15]. (2) This work aims for explainability, and since DLs' focus is primarily on relationships between individuals and their attributes, we can make adjustments to the generation process of the second KG to create a new variation that has in focus the relations between individuals.
The new variations adhere to the original structure where images are divided into garments and garment parts, represented as annotation categories. In this KG, attributes are related directly to the image individuals. To implement this, we create 5 classes: *Image, AnnotationCategory, AnnotationSupercategory, Attribute, AttributeCategory*. Relations between individuals of these classes are made using the following object properties: *hasAnnotationCategory, hasAnnotationSupercategory, hasAttribute, hasAttributeCategory*. The trivial information for the classifying task with the focus on the apparel, such as filename, width, height, etc., is not added to the dataset. A subgraph of this variation is given in Figure 4.5. Note that the term 'supercategory' of an attribute from the original structure is referred to here as 'category' for simplicity.
The LLM-generated descriptions are created dynamically in an unstructured manner, and mapping such information into the purposed KG is a complex task [MDD22; Aya23; DL22]. Therefore, LLM-generated context is not taken into consideration in this approach due to time constraints.

Augmentation Using CEL

To learn class expressions, CEL algorithms from *Ontolearn* [6] library, like the Tree-based learner (TDL), Deep Reinforcement Learning (Drill) and Class Expression Learning for Ontology Engineering (CELOE) can be used. For the learning task, CEL algorithms require a learning problem

[6]`https://github.com/dice-group/Ontolearn/tree/develop`

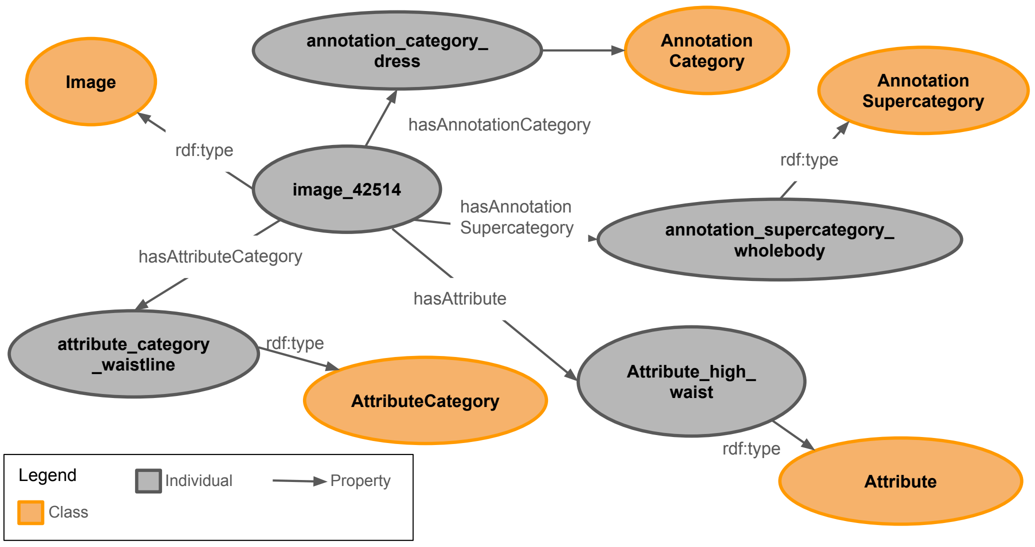

Fig. 4.5.: Subgraph of the knowledge graph tailored to the class expression learning task. This is a variation of the graph shown in Figure 4.2.

specified by sets of positive and negative examples. In the suggested RAG approach aided by CEL, the learning problem is generated as follows: after ranking each image based on the given query using the embedding-based retriever, the top k images are considered as positive examples and the bottom k are considered as negative examples. In this RAG model we only consider embedding-based retriever because, during evaluation, it is shown to perform better then the BM25 retriever.

After fitting the learning problem to the CEL model, the algorithm will return a class expression, which we use for re-ranking the instances after the initial ranking by the embedding-based retrieval model. During the re-ranking process, we first find the classified instances by the generated class expression. In the second variation of the KG that was constructed for the task of class expression learning, there is no LLM-generated context, and the KG is constructed using only the data from the Fashionpedia dataset, i.e., the human-annotated data. That means that the classifying expression will not be able to express certain information that is provided only in the LLM-generated descriptions, like, for example, the color of the apparel. Therefore, to overcome this shortcoming, by using the LLM, the system generates a short summary of the top k retrieved documents by the embedding-based retriever. This summary represents the most relevant instances based on the embedding-based retriever, and it is used

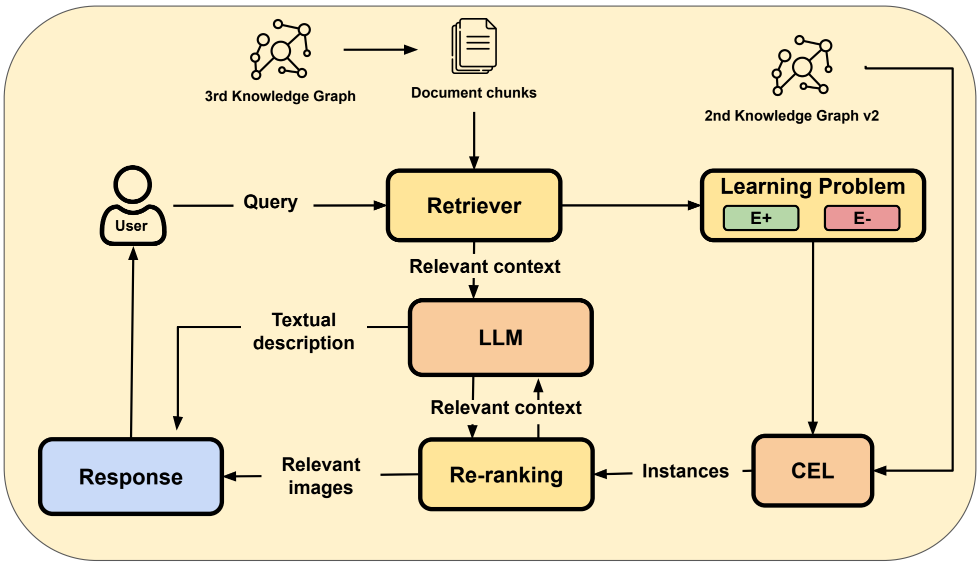

Fig. 4.6.: Diagram of the multimodal RAG with class expression learning. Refer to Figure 4.4 for color specifications.

to rank the classified instances. For the ranking score, we use cosine similarity between the embedding of the generated summary and the document embedding of each classified instance. At this point the ranking list contains only the classified instances. The rest of the instances that are not covered by the learned class expression are placed at the end of the list, in the order set by the embedding-based retriever. After the re-ranking procedure is over, the system returns the top k instances, where images are shown to the user as the final response and documents are used to generate the textual description.

Text Generation

The textual description that accompanies the returned images is generated in the same way as explained in Section 4.3 but this time for the descriptions of the top k instances after re-ranking. A class expression is human-understandable it can be converted to DL syntax which can be verbalized into natural language with the help of an LLM.

5 Evaluation

In this work we presented two RAG models for the task of image-based recommendation with generated textual description. Since the focus of this thesis is on the recommendation relevance and not on the generated textual description, our evaluation is also focused on the retrieval-augmentation part of the RAG models. In this chapter we will refer to the retrieval-augmentation method simply as the retrieval method. The retrieval part is considered the most important part of a RAG system since most LLMs perform better if provided background knowledge containing the relevant information [Edg+24; Yu+24]. In this section we evaluate the performance of the retrieval methods of our proposed RAG models in terms of common information-retrieval metrics. Furthermore, we evaluate the performance of different CEL models using clusters of instances to construct a learning problem, showing the capability of the CEL models in classifying such clusters.

5.1 Evaluation Setup

To evaluate the retrieval methods, a benchmark dataset is constructed. The images are represented by a query/question generated using the LLM, which asks for the apparel in the image. This generated question can then be used as the user's input to test the system. Given initial results on the proposed benchmark, we construct a second benchmark dataset addressing the limitations of the first one.

5.1.1 Models Used for Benchmark Creation and Evaluation Experiments

The LLM model used in this evaluation is *LLaVa-Next* [Liu+24], more concretely *llava-v1.6-mistral-7b-hf* [1] by Hugging Face [2]. LLaVa-Next is a model that leverages *Mistral-7B-Instruct-v0.2* [3]. To utilize the LLM model in the Python coding, the OpenAi [4] client is used. The temperature argument is set to 0.1 and the seed to 1.

The embedding model used in this evaluation is *Mistral-7B-v0.1*, specifically *E5-mistral-7b-instruct* [5] [Wan+23a; Wan+22] which has 32 layers and an embedding size of 4096 with 7.11 billion parameters. LLM and embedding models were hosted on servers provided by the Paderborn University.

Class Expression Learning Models

To evaluate CEL models capabilities that can be used in the second RAG approach, we use the following models from Ontolearn (version 0.8.1): TDL, Drill, and CELOE. All three suggested learning models share the same value for the attribute *knowledge_base*, which is set to be the second variation of the second KG constructed for the task of class expression learning. For TDL and Drill, the argument *use_nominals* is set to *True* in order to support nominals.
Maximum runtimes for each model:

- TDL - 10 seconds
- Drill - 10 seconds
- CELOE - 60 seconds

CELOE is a refine-based model, and it's slower than the other 2 models; therefore, the time restriction was set to 60 seconds. Every other argument that is not mentioned here was left to its default value.
To evaluate the retrieval method of the second RAG system, we consider

[1] `https://huggingface.co/llava-hf/llava-v1.6-mistral-7b-hf`
[2] `https://huggingface.co/`
[3] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`
[4] `https://github.com/openai/openai-python`
[5] `https://huggingface.co/intfloat/e5-mistral-7b-instruct`

only the best performing CEL model which is shown in Section 5.2.3 to be TDL.

5.1.2 Benchmark Datasets & Evaluation Sample

To create the benchmark dataset, we used the third KG with 45,623 instances representing the 45,623 images in the Fashionpedia dataset. The initial step in creating the benchmark dataset is to generate a representative question for each instance using the LLM.

The query used to generate the questions about each image is as following:

> "*Consider you are a user that is looking for clothes/apparel in an online recommendation system. Formulate a query of a prompt-like structure that you would ask in such a way that the attached image would be recommended to you. To generate the query, you can take into consideration the following auxiliary information about the image: <description>. Only write the query, which should be a question, and always end with a question mark.*"

Where <*description*> is a string combining all the descriptions of the image, including the LLM description. The final sentence of the prompt is added because during the experiments, the LLM would not always yield a question.
This prompt will return a single question that tends to cover every garment in the image. Every question is then divided into multiple self-contained questions that ask for different garments/garment parts and/or attributes. This is done to achieve questions that are more general in nature and won't strictly be describing the image they represent. That means that a short question can be valid for multiple images. To separate the long question into shorter questions, the LLM is utilized again using the following prompt:

> "*Separate each part of the following question into self-contained questions. Separate them in new line without using ordering numbers. Here is the question: <question>*"

First benchmark dataset. The output of the just described step is the first benchmark dataset, which is represented as a dictionary with key-value pairs stored in JSON format where the keys are IRIs that represent the image individuals and the values are the separated question for each image.
The **evaluation sample** is made of 100 instances chosen randomly from the benchmark dataset. For each instance, a single question from the set of multiple questions mapped to that instance is selected randomly to represent it. Table A.1 shows the sample entities.
During experiments we noticed similarity in questions of different instances. By using T-distributed Stochastic Neighbor Embedding (TSNE) of the *scikit-learn* library version 1.5.2, we were able to visualize the clusters created between these questions. Figure 5.1 shows the TSNE plot and the images belonging to a cluster of 4 instances. The arguments set for the TSNE tool are specified below, with everything not specified left to its default value:

- n_components=2
- perplexity=2
- init="random"
- max_iter=500
- random_state=0

Second benchmark dataset. A new benchmark dataset was created, this time considering the question similarity. The first benchmark was used for this purpose, and since an image (represented by the IRI) is mapped to multiple questions, only a single question was randomly selected for the next step. Each selected question was encoded using the embedding model to make it easier to find similarities between them. To find the similar questions of a given question, the k-nearest neighbor classifier from the *scikit-learn* library version 1.5.2, was used.
All arguments were left to default, besides the number of neighbors *n_neighbors*, usually referred to as k. For this benchmark setup, we set $k = \sqrt{N}$ as a common choice for measuring performances [LS96; Has+14; Udd+22], where $N = 45,623$ is the total number of images in the dataset.

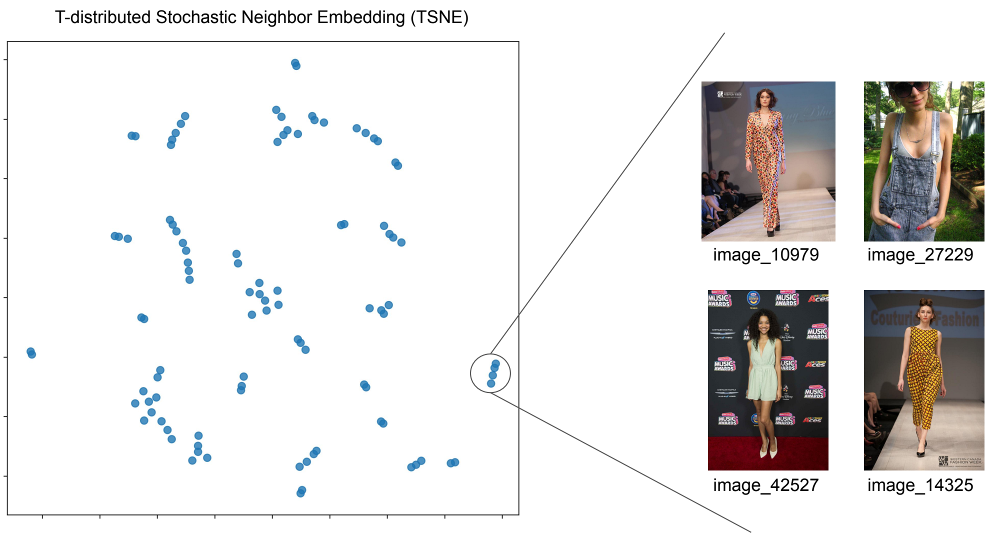

Fig. 5.1.: TSNE plot of the evaluation sample points. **The shown images are the only 4 images that share the word "jumpsuit" in their query**.

Learning Problem For CEL Models Performance Evaluation

The CEL algorithms require a learning problem to learn a class expression. For that purpose we reuse the list of k-nearest neighbors as positive examples to learn a representative class expression for that particular cluster of images. For the negative examples, we chose the same amount of random instances from the set of all instances excluding the positive examples in order to reduce bias in classification.

Learning Problem for Retrieval Method with CEL Re-ranking

To evaluate the retrieval method of the second RAG framework, we have to set the k, which specifies the number of positive examples that we chose from top-ranked instances by the embedding-based retriever. We try different 3 values for k, i.e., 5, 10, and 20. k's upper limit that we test is set to 20 because of the limitation on tokens that the LLM assigns to the k documents passed during the re-ranking step. As negative examples we randomly selected k instances from the set of all instances excluding the positive examples.

5.1.3 Metrics

Metrics for Retrieval Models

The metrics that will be used to evaluate the retrieval methods are Mean Reciprocal Rank (MRR) and Hits@K, which are common metrics in the context of information retrieval. The higher the value of MRR or Hits@K, the better the results. For these two metrics, the equation from Chen et al. [Che+20] will be used. MRR is measured as follows:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}, \tag{5.1}$$

where Q is the set of questions and rank_i is the position of the target IRI, that the question belongs to.
As for the Hits@K the following equation will be used:

$$\text{Hits@K} = \frac{|\{q \in Q : q < k\}|}{|Q|} \in [0,1], \tag{5.2}$$

where k is used to set the limit for the top k candidates. For this evaluation setup we select $k \in \{10, 20, 50, 100\}$.

Metrics for class expression learning models

To evaluate the performance of the class expression learning models, the following metrics are used:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{5.3}$$

where TP stands for *True Positive* and FP stands for *False Positive*,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{5.4}$$

where FN stands for *False Negatives*,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{5.5}$$

where TN stands for *True Negatives*, and

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5.6}$$

5.2 Experimental Results

We discuss the experimental results in this section, where we show the performance of retrieval methods of both proposed RAG frameworks. In the first benchmark we tested only the performance of BM25 and embedding-based retrievers. These initial results lead us to construct the second benchmark while addressing the limitation of 1 relevant instance per question. In the second benchmark we also evaluate the retrieval method with CEL re-ranking. In the end we evaluate the performance of different CEL models to show the capabilities of these models in classifying clusters of instances.

5.2.1 Results on the First Banchmark Dataset

Here we discuss the experimental results of the retrieval models, i.e., BM25 and embedding-based retrievers used in the first RAG model. The evaluation consists of 2 steps: 1) Finding the score of each document by matching the query and ranking them from highest to lowest based on that score. In the case of the BM25 retriever, the score would be the BM25 score, and in the case of the embedding-based retriever, the score would be the cosine similarity between the query embedding and each document embedding. 2) After ranking each document by their score, we find Hits@K for $k \in \{10, 20, 50, 100\}$ and calculate the reciprocal rank for each target IRI on the evaluation sample to finally measure the MRR.

Table 5.1 shows the MRR and Hits@K scores for each retrieval model used in the first RAG framework, where a higher score means a better performance on retrieving a fitting document.

There are a few noticeable takeaways from these evaluation results. There is a significant increase of MRR and Hits@K when the retrievers are used over the enriched KG. One factor that contributes to that is the fact that the asked queries contain keywords that are not part of the

Retrieval model	KG	MRR	H@10	Hits@20	Hits@50	Hits@100
BM25	basic	0.003	0.01	0.01	0.04	0.006
Embedding-based	basic	0.005	0.00	0.03	0.06	0.11
BM25	enriched	0.07	0.12	0.16	**0.29**	**0.35**
Embedding-based	enriched	**0.08**	**0.16**	**0.21**	0.28	0.34

Tab. 5.1.: Evaluation of retrieval models in the first benchmark dataset in terms of Mean Reciprocal Rank (MRR) and Hits@K for $k \in \{10, 20, 50, 100\}$. Each metric is measured on the KG without LLM-generated descriptions referred to as **'basic'** and on the KG enriched with the LLM-generated descriptions referred to as **'enriched'**. The highest value for each metric is marked in bold.

Retrieval model	KG	MRR	H@10	Hits@20	Hits@50	Hits@100
BM25	basic	0.05	0.11	0.16	0.36	0.47
Embedding-based	basic	0.16	0.29	0.44	0.63	0.74
BM25	enriched	0.17	0.40	0.50	0.72	0.80
Embedding-based	enriched	**0.29**	**0.48**	**0.66**	**0.81**	**0.88**

Tab. 5.2.: Evaluation of retrieval models in the second benchmark dataset in terms of MRR and Hits@K. For term definition refer to Table 5.1.

$\lvert E^+\rvert$	$\lvert E^-\rvert$	KG	MRR	H@10	H@20	H@50	H@100
5	5	basic	0.16	0.33	0.48	0.58	0.71
10	10	basic	0.14	0.34	0.46	0.62	0.75
20	20	basic	0.14	0.31	0.37	0.57	0.74
5	5	enriched	**0.24**	**0.50**	0.57	**0.72**	0.79
10	10	enriched	**0.24**	0.45	**0.59**	0.71	**0.81**
20	20	enriched	**0.24**	0.42	0.51	0.67	0.78

Tab. 5.3.: Evaluation of 2nd RAG retrieval method given different numbers of learning problem examples. The evaluation is performed in the second benchmark dataset in terms of MRR and Hits@K. For term definition refer to Table 5.1.

original Fashionpedia dataset, for example, colors, specific descriptive adjectives, uncommon traits on the clothes, etc. The LLM-generated description is more likely to contain these words, and as a result, this leads to more relevant matches during the search. Nonetheless, the LLM-generated description is an addition to the current descriptions given for the images and is undeniably extra context tailored to the image, which means more information for the retriever to use during comparison with the query. We also notice that the embedding-based retriever is outperforming the BM25 retriever.

5.2.2 Results on the Second Benchmark Dataset

Although the results in the first benchmark dataset show some performance improvement on the enriched KG, the measured scores, which are shown in Table 5.1, are relatively low. Delving into the details, we find that the main contributing factor for that is the first benchmark setup. Each image is mapped to a number of queries and does not take into consideration the similarity between the queries. That means that if we were to group similar queries together, we could map them to a cluster of images. Figure 5.1 shows the TSNE plot of 100 queries from the evaluation samples where clusters of similar queries are visible. Furthermore, in the figure you see the images of a 4-element cluster. Diving deeper into what elements in this cluster share, we noticed that all their queries share the keyword "jumpsuit," which is not found in any other query in the sample set.

The second benchmark setup increases the possibility of finding a relevant image given a query because now multiple relevant images can answer the same query. During measurements, this reflects in a higher ranking and thereafter in a higher MRR and Hits@K as displayed by the evaluation results shown in Table 5.2 and Table 5.3. In Table 5.2, we notice a drastic increase in both metrics for both retrieval models, the BM25 and the embedding-based retrievers, when they are used in the enriched KG. The embedding-based retriever significantly outperforms the BM25 one.
We notice the same in Table 5.3, where the retrieval method using CEL for re-ranking is performing better in the enriched KG. Comparing the results of Table 5.2 and Table 5.3, we can say that the retrieval method using CEL outperforms the BM25 retriever on both KGs. MRR change is minimal when testing different lengths of LP.
Worth noting is that when using 5 positive examples (and 5 negative examples), the retrieval method using CEL slightly outperforms the embedding-based model in the basic KG in terms of H@10 and H@20. This finding means that in a knowledge-restricted dataset (not enriched with LLM-generated descriptions) like Fashionpedia, our proposed retrieval method using CEL performs slightly better than the traditional embedding-based retriever for $10 \leq k \geq 20$ in Hits@K.

CEL	F1	Accuracy	Precision	Recall
TDL	**0.77**	**0.78**	**0.80**	0.78
CELOE	0.71	0.61	0.57	0.93
Drill	0.66	0.50	0.5	**1.00**

Tab. 5.4.: CEL algorithms performance in terms of mean values for F1, accuracy, precision, and recall of 100 LP samples.

5.2.3 Performance of CEL Models

The class expression learning models are evaluated in the same evaluation sample as the retrieval models on the second benchmark dataset, where each instance is mapped to its k-nearest neighbors, making it possible to create a learning problem for each of them. Table 5.4 shows the performance of CEL algorithms in the evaluation sample in terms of mean values for F1 score, accuracy, precision, and recall. We notice that TDL is performing better than the other 2 models in terms of F1, accuracy, and precision and is overtaken by Drill in recall. Diving deeper into the class expression learned for each model, we noticed a few things. TDL concepts were complex and lengthy, containing mainly nominal-based class expressions. CELOE does not support nominals, and the learned class expressions were mainly cardinality-restriction-based. Whereas Drill couldn't learn a more specific class expression other than *Thing* ($\top$) class. Even after removing the maximum runtime constraint, the same behavior was recorded for Drill.

6 Summary & Discussion

In this thesis we proposed two multimodal RAG approaches over knowledge graphs for the task of image-based recommendation accompanied by a generated textual description. Creating a KG from a multimodal dataset is a fundamental part of this thesis, which sets the foundations for the other components of the RAG models. Utilizing a multimodal LLM to enrich the KG with additional context promised an overall increase in retrieval performance. We further make use of class expression learning to extend the basic RAG workflow and show that this approach can be promising in human-annotated datasets. In this chapter a short summary of the core parts of the thesis is provided while addressing the main findings, limitations, and directions for future work.

6.1 Knowledge Graphs Generation

During this work, multiple KGs were generated. The first KG was a direct mapping of the Fashonpedia dataset into an RDF KG. This data was restructured and summarized without loss of apparel-descriptive information into a second KG, in a way that image descriptions could be efficiently divided into chunks and used in the retrieval process of the RAG model. To go beyond the limitations that the Fashionpedia dataset provided about the images, we utilized a multimodal LLM to generate a description of each image and store that information in the KG. For the purpose of evaluation, we refer to the KG enriched with LLM-generated descriptions as the third KG or the enriched KG. During experiments, we notice that the LLM description contained information not covered in the Fashionpedia dataset, like the color of the garments and new descriptive adjectives. We also constructed a KG specifically for the task of class expression learning, where focus was shifted into increasing expressiveness through the addition of classes and relationships.

6.2 Retrieval-Augmented Generation Models

In this work we put more emphasise in the retrieval-augmented part wich we refer for simplicity as retrieval method. A brief summary of each retrieval method of each RAG model is given in this section as well as the text generation part.

6.2.1 1st Retrieval Method: Score-based Retrieval

In this work we presented two different retrieval models, the BM25 retriever and the embedding-based retriever. BM25 retriever uses BM25 score to rank the instances relevant to a given query, whereas the embedding-based retriever uses cosine similarity. Evaluation results indicate that the embedding-based retriever performs better than the BM25 one in terms of MRR and Hits@K for $k \in \{10, 20, 50, 100\}$. However, both retrievers have an increase in performance when used in the enriched KG. This result can be attributed to the additional information that the LLM has provided for each image, which helps for a better matching of documents with the user's query during the scoring process.

BM25 retriever. Diving into what makes BM25 retriever perform worse than the embedding-based one, we discuss some of the BM25 retriever limitations. Since BM25 is a probabilistic model and does not take the semantics of the sentences into consideration, it cannot distinguish negative queries or plural forms of the words. Although we do not consider negative queries in the evaluation, this is still a limitation that may further expand the performance gap between the BM25 retriever and the embedding-based one. In the case of plural forms of the words, since BM25 is terms-based, meaning that it looks for an exact match of terms, it may perform worse when it comes to distinguishing between singular and plural forms. For example, given a user's query "What are some knee-high dresses" the word "dresses" is not the same as "dress" which is more common in the dataset, and a comparison between the two terms won't result in a match.

6.2.2 2nd Retrieval Method: Score-based + CEL Re-ranking

In the second RAG model we integrate CEL to re-rank the instances after they are initially ranked using the embedding-based retriever. We create a learning problem by using the top k ranked instances E^+ and k random instances as negative examples E^- in such a way that $E^+ \cap E^- = \emptyset$. The CEL algorithm generates a class expression, which we use to retrieve all classified instances from the KG. Using the LLM, a summary of all documents for each positive example $e^+ \in E^+$ concatenated together is generated and used to rank the classified instances based on cosine-similarity score. Our findings show that this retrieval method outperforms the embedding-based retriever in terms of Hits@10 and Hits@20. The results indicate that this can be a promising approach in structured data where additional LLM-generated description is not available.

6.3 Class Expression Learning Models Performance

We evaluated different CEL models in classifying clusters of instances. These clusters were identified using k-nearest neighbor algorithm to find the most related instances to each instance of the KG. A learning problem was constructed given the k-nearest neighbors as positive examples E^+ and k random instances as negative examples E^- where $E^+ \cap E^- = \emptyset$. Results show that the best model, TDL, achieved an F1 score of 0.77 and accuracy of 0.78. A learned class expression can be considered as an explainable classifier over individuals and further be verbalized into natural language. Furthermore using OWL class expression we can detect accurate clusters/communities which are important when describing a system [Edg+24]. The learned class expression can also be than used for ontology engineering, to enrich the KG.

6.4 Limitations

KG generation. The KG that is used for class expression learning contains only the information from the Fashonpedia dataset and does not take into consideration the LLM-generated context, which could have potentially increased the performance of the CEL models since later we showed that retrievers perform better in the enriched KG. This is a complex task [MDD22; Aya23; DL22] and was not considered because of the time constraints. Another limitation of the whole framework is the need to generate multiple KGs. Although the framework can be optimized to use a single KG that covers all aspects: mapping raw data of the multimodal dataset, efficient retrieval, and expressiveness for CEL, it would require more planning and testing, which we did not consider due to time constraints.

Prompt structure. In this work we utilize LLM frequently in different tasks, like generating descriptions for images of the Fashionpedia dataset, generating textual summaries used for re-ranking during the second RAG model, and question generation during benchmark dataset construction. During this work we do not spend time resources on prompt engineering since it is not the main focus of this thesis. Experimenting on a larger range of prompt structures may reveal which prompts bring the best out of the proposed RAG models.

Class expression learning. Our evaluation for CEL is limited on a few factors. First of all, during the learning problem selection, we tested the same number of positive and negative examples, and therefore trying a different proportion is unexplored. Continuous testing is required to find the best performing values. The k which decides the length of the positive and negative examples is limited by the amount of tokens that the LLM has to represent the k documents passed during re-ranking. Also, we only tried one structure for the KG generated for the CEL task. Multiple ways of structuring the data can be tested to find the one that yields better results.

Retrieval In this work we use only a single retrieval step and do not consider multiple iterations over the documents. We focus on proposing and evaluating a novel augmentation approach where we use CEL combined with LLM to re-rank the documents.

6.5 Directions for Future Work

Our KG creation approach is tailored to the Fashionpedia dataset. However, it can be extended into a general approach that transforms similar datasets from different domains into a KG. This way, we can apply the RAG models not only in fashion apparel but in other domains. Edge et al. [Edg+24] have shown in their study that using an LLM-derived graph index, it's possible to create a KG from text-based documents, opening possibilities to represent unstructured data into a KG that can be used in our system through adaptation. By doing that we can also overcome the limitation of representing the LLM-generated description in the KG used for CEL, leaving it an open topic whether our proposed retrieval method using CEL can outperform the embedding-based retriever in the enriched KG. Furthermore, Wang et al. [Wan+23b] show that a personalized KB (PKB) can be created from the user input, and our framework can adapt such an approach to "memorize" users' preferences and improve future responses in the sense of penalization.
Our work focus is mainly on the task of recommendation, but it can be extended to cover question-answering tasks. Like GenRead [Yu+22], we can apply a similar prompt generation strategy for generating additional context for the given images to adapt our system for zero-shot question-answering about multimodal data. Furthermore, our system can integrate a similar approach as in RoG [Luo+23] to create reasoning paths on the KG, which can be used to aid the retrieval process.

Bibliography

[AH17] Abdullah and Mohammad S. Hasan. “An application of pre-trained CNN for image classification”. In: *2017 20th International Conference of Computer and Information Technology (ICCIT)*. 2017, pp. 1–6 (cit. on p. 1).

[Abu+23] Hasan Abu-Rasheed, Mareike Dornhöfer, Christian Weber, et al. “Building contextual knowledge graphs for personalized learning recommendations using text mining and semantic graph completion”. In: *2023 IEEE International Conference on Advanced Learning Technologies (ICALT)*. IEEE. 2023, pp. 36–40 (cit. on p. 1).

[AWR22] Nathan Anderson, Caleb Wilson, and Stephen D Richardson. “Lingua: Addressing scenarios for live interpretation and automatic dubbing”. In: *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track)*. 2022, pp. 202–209 (cit. on p. 10).

[Asa+23] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. “Self-rag: Learning to retrieve, generate, and critique through self-reflection”. In: *arXiv preprint arXiv:2310.11511* (2023) (cit. on p. 10).

[Aya23] H Ambre Ayats. “Knowledge graph construction from texts with an explainable, human-centered Artificial Intelligence”. PhD thesis. Université de Rennes, 2023 (cit. on pp. 30, 46).

[Baa03] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003 (cit. on p. 6).

[BHS08a] Franz Baader, Ian Horrocks, and Ulrike Sattler. “Chapter 3 Description Logics”. In: *Handbook of Knowledge Representation*. Ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Vol. 3. Foundations of Artificial Intelligence. Elsevier, 2008, pp. 135–179 (cit. on p. 6).

[BHS08b] Franz Baader, Ian Horrocks, and Ulrike Sattler. “Chapter 3 Description Logics”. In: *Handbook of Knowledge Representation*. Ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Vol. 3. Foundations of Artificial Intelligence. Elsevier, 2008, pp. 135–179 (cit. on p. 6).

[Bal+21] Federico Baldassarre, David Menéndez Hurtado, Arne Elofsson, and Hossein Azizpour. “GraphQA: protein model quality assessment using graph convolutional networks”. In: *Bioinformatics* 37.3 (2021), pp. 360–366 (cit. on p. 16).

[Bro+20] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. *Language Models are Few-Shot Learners*. 2020. arXiv: `2005.14165 [cs.CL]` (cit. on p. 15).

[Che+22] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W Cohen. “Murag: Multimodal retrieval-augmented generator for open question answering over images and text”. In: *arXiv preprint arXiv:2210.02928* (2022) (cit. on pp. 12, 17).

[Che+20] Zhe Chen, Yuehan Wang, Bin Zhao, et al. “Knowledge Graph Completion: A Review”. In: *IEEE Access* 8 (2020), pp. 192435–192456 (cit. on p. 38).

[Dem23] Caglar Demir. “Learning Continuous Representations for Knowledge Graphs”. PhD thesis. Universitätsbibliothek, 2023 (cit. on pp. 2, 5–7).

[Det+18] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. “Convolutional 2d knowledge graph embeddings”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 5).

[DL22] Xinyu Du and Ning Li. “Academic Paper Knowledge Graph, the Construction and Application.” In: *ICBASE*. 2022, pp. 15–27 (cit. on pp. 30, 46).

[Edg+24] Darren Edge, Ha Trinh, Newman Cheng, et al. “From local to global: A graph rag approach to query-focused summarization”. In: *arXiv preprint arXiv:2404.16130* (2024) (cit. on pp. 15, 33, 45, 47).

[Fer+23] Mariano Ferreirone, Mario Lezoche, Diego Torres, and Hervé Panetto. “Transformations on knowledge representation between OWL and RDF knowledge graphs: a study case”. In: *Electronic Journal of SADIO* 22 (2023) (cit. on p. 5).

[Gao+22] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. “Precise zero-shot dense retrieval without relevance labels”. In: *arXiv preprint arXiv:2212.10496* (2022) (cit. on p. 17).

[Gao+24] Yunfan Gao, Yun Xiong, Xinyu Gao, et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: `2312.10997 [cs.CL]` (cit. on pp. 1, 8–11, 15, 57).

[Gui+21] Liangke Gui, Borui Wang, Qiuyuan Huang, et al. “Kat: A knowledge augmented transformer for vision-and-language”. In: *arXiv preprint arXiv:2112.08614* (2021) (cit. on p. 17).

[Has+14] Ahmad Basheer Hassanat, Mohammad Ali Abbadi, Ghada Awad Altarawneh, and Ahmad Ali Alhasanat. “Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach”. In: *arXiv preprint arXiv:1409.0919* (2014) (cit. on p. 36).

[He+24] Xiaoxin He, Yijun Tian, Yifei Sun, et al. “G-retriever: Retrieval-augmented generation for textual graph understanding and question answering”. In: *arXiv preprint arXiv:2402.07630* (2024) (cit. on p. 16).

[Hei+22] Stefan Heindorf, Lukas Blübaum, Nick Düsterhus, et al. “Evolearner: Learning description logics with evolutionary algorithms”. In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 818–828 (cit. on pp. 1, 2).

[Hog+21a] Aidan Hogan, Eva Blomqvist, Michael Cochez, et al. “Knowledge Graphs”. In: *ACM Comput. Surv.* 54.4 (July 2021) (cit. on p. 5).

[Hog+21b] Aidan Hogan, Eva Blomqvist, Michael Cochez, et al. “Knowledge graphs”. In: *ACM Computing Surveys (Csur)* 54.4 (2021), pp. 1–37 (cit. on p. 1).

[Jia+20] Menglin Jia, Mengyun Shi, Mikhail Sirotenko, et al. “Fashionpedia: Ontology, segmentation, and an attribute localization dataset”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer. 2020, pp. 316–332 (cit. on pp. 1, 12, 13, 57).

[Jia+23a] Huiqiang Jiang, Qianhui Wu, Xufang Luo, et al. “Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression”. In: *arXiv preprint arXiv:2310.06839* (2023) (cit. on p. 10).

[Jia+23b] Zhengbao Jiang, Frank F Xu, Luyu Gao, et al. “Active retrieval augmented generation”. In: *arXiv preprint arXiv:2305.06983* (2023) (cit. on p. 10).

[Kan+23] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. “Large language models struggle to learn long-tail knowledge”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 15696–15707 (cit. on p. 8).

[KSO23] Minkyu Kim, Kim Sung-Bin, and Tae-Hyun Oh. “Prefix tuning for automated audio captioning”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on p. 12).

[KKH24] Hamin Koo, Minseon Kim, and Sung Ju Hwang. “Optimizing Query Generation for Enhanced Document Retrieval in RAG”. In: *arXiv preprint arXiv:2407.12325* (2024) (cit. on p. 9).

[LS96] Upmanu Lall and Ashish Sharma. “A nearest neighbor bootstrap for resampling hydrologic time series”. In: *Water resources research* 32.3 (1996), pp. 679–693 (cit. on p. 36).

[LAS99] Ora LASSILA. “Resource description framework (RDF) Model and syntax specification”. In: *http://www.w3.org/TR/REC-rdf-syntax/* (1999) (cit. on p. 5).

[Lee+23] Suhyeon Lee, Won Jun Kim, Jinho Chang, and Jong Chul Ye. “LLM-CXR: Instruction-Finetuned LLM for CXR Image Understanding and Generation”. In: *arXiv preprint arXiv:2305.11490* (2023) (cit. on p. 1).

[Leh10] Jens Lehmann. *Learning OWL class expressions*. Vol. 22. IOS Press, 2010 (cit. on p. 7).

[Leh+11] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. “Class expression learning for ontology engineering”. In: *Journal of Web Semantics* 9.1 (2011), pp. 71–81 (cit. on pp. 2, 8).

[LH10] Jens Lehmann and Pascal Hitzler. “Concept learning in description logics using refinement operators”. In: *Machine Learning* 78 (2010), pp. 203–250 (cit. on pp. 2, 6).

[Lew+20a] Mike Lewis, Yinhan Liu, Naman Goyal, et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880 (cit. on p. 15).

[Lew+20b] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474 (cit. on p. 15).

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755 (cit. on p. 12).

[Liu+24] Haotian Liu, Chunyuan Li, Yuheng Li, et al. *Llava-next: Improved reasoning, ocr, and world knowledge*. 2024 (cit. on p. 34).

[Lu+22] Shuai Lu, Nan Duan, Hojae Han, et al. “ReACC: A Retrieval-Augmented Code Completion Framework”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6227–6240 (cit. on p. 12).

[Luo+23] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. “Reasoning on graphs: Faithful and interpretable large language model reasoning”. In: *arXiv preprint arXiv:2310.01061* (2023) (cit. on p. 47).

[Ma+23] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. “Query rewriting for retrieval-augmented large language models”. In: *arXiv preprint arXiv:2305.14283* (2023) (cit. on pp. 8, 9).

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008, p. 482 (cit. on p. 27).

[MDD22] Igor Melnyk, Pierre Dognin, and Payel Das. “Knowledge graph generation from text”. In: *arXiv preprint arXiv:2211.10511* (2022) (cit. on pp. 30, 46).

[MR08] Boris Motik and Riccardo Rosati. “Reconciling description logics and rules”. In: *Journal of the ACM (JACM)* 57.5 (2008), pp. 1–62 (cit. on p. 30).

[MSH07] Boris Motik, Rob Shearer, and Ian Horrocks. “Optimized reasoning in description logics using hypertableaux”. In: *Automated Deduction–CADE-21: 21st International Conference on Automated Deduction Bremen, Germany, July 17-20, 2007 Proceedings 21*. Springer. 2007, pp. 67–83 (cit. on p. 5).

[Pat15] Peter Patel-Schneider. “Using description logics for RDF constraint checking and closed-world recognition”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015 (cit. on p. 30).

[Pen+24] Wenjun Peng, Guiyang Li, Yue Jiang, et al. “Large language model based long-tail query rewriting in taobao search”. In: *Companion Proceedings of the ACM on Web Conference 2024*. 2024, pp. 20–28 (cit. on p. 9).

[Rad+21] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763 (cit. on pp. 1, 11).

[RKA+12] Faisal Rahutomo, Teruaki Kitasuka, Masayoshi Aritsugi, et al. “Semantic cosine similarity”. In: *The 7th international student conference on advanced science and technology ICAST*. Vol. 4. 1. University of Seoul South Korea. 2012, p. 1 (cit. on p. 27).

[Rot+24] Noam Rotstein, David Bensaıd, Shaked Brody, Roy Ganz, and Ron Kimmel. “Fusecap: Leveraging large language models for enriched fused image captions”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 5689–5700 (cit. on p. 1).

[Sab17] Serwah Sabetghadam. “A graph-based model for multimodal information retrieval”. PhD thesis. Technische Universität Wien, 2017 (cit. on p. 25).

[SB88] Gerard Salton and Christopher Buckley. “Term-weighting approaches in automatic text retrieval”. In: *Information processing & management* 24.5 (1988), pp. 513–523 (cit. on p. 27).

[Shi+23] Freda Shi, Xinyun Chen, Kanishka Misra, et al. “Large language models can be easily distracted by irrelevant context”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31210–31227 (cit. on p. 9).

[Sir+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A practical owl-dl reasoner”. In: *Journal of Web Semantics* 5.2 (2007), pp. 51–53 (cit. on p. 5).

[Sun+22] Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. “Recitation-augmented language models”. In: *arXiv preprint arXiv:2210.01296* (2022) (cit. on p. 9).

[Udd+22] Shahadat Uddin, Ibtisham Haque, Haohui Lu, Mohammad Ali Moni, and Ergun Gide. “Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction”. In: *Scientific Reports* 12.1 (2022), p. 6256 (cit. on p. 36).

[VO24] Vineeth Venugopal and Elsa Olivetti. “MatKG: An autonomously generated knowledge graph in Material Science”. In: *Scientific Data* 11.1 (2024), p. 217 (cit. on p. 1).

[VK14] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledgebase”. In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85 (cit. on p. 12).

[Wan+24] Hongru Wang, Wenyu Huang, Yang Deng, et al. “Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems”. In: *arXiv preprint arXiv:2401.13256* (2024) (cit. on p. 9).

[Wan+23a] Liang Wang, Nan Yang, Xiaolong Huang, et al. “Improving Text Embeddings with Large Language Models”. In: *arXiv preprint arXiv:2401.00368* (2023) (cit. on pp. 26, 34).

[Wan+22] Liang Wang, Nan Yang, Xiaolong Huang, et al. “Text Embeddings by Weakly-Supervised Contrastive Pre-training”. In: *arXiv preprint arXiv:2212.03533* (2022) (cit. on pp. 26, 34).

[Wan+23b] Xintao Wang, Qianwen Yang, Yongting Qiu, et al. “Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases”. In: *arXiv preprint arXiv:2308.11761* (2023) (cit. on pp. 15, 16, 47).

[Xu+19] Haiyang Xu, Hui Zhang, Kun Han, et al. “Learning alignment for multimodal emotion recognition from speech”. In: *arXiv preprint arXiv:1909.05645* (2019) (cit. on p. 12).

[Yan+22] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, et al. “An empirical study of gpt-3 for few-shot knowledge-based vqa”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 36. 3. 2022, pp. 3081–3089 (cit. on p. 17).

[Yu+24] Hao Yu, Aoran Gan, Kai Zhang, et al. “Evaluation of Retrieval-Augmented Generation: A Survey”. In: *arXiv preprint arXiv:2405.07437* (2024) (cit. on p. 33).

[Yu+22] Wenhao Yu, Dan Iter, Shuohang Wang, et al. “Generate rather than retrieve: Large language models are strong context generators”. In: *arXiv preprint arXiv:2209.10063* (2022) (cit. on pp. 9, 16, 47).

[Yu+23] Wenhao Yu, Hongming Zhang, Xiaoman Pan, et al. “Chain-of-note: Enhancing robustness in retrieval-augmented language models”. In: *arXiv preprint arXiv:2311.09210* (2023) (cit. on p. 9).

[Zha24] Wenjia Zhai. “Self-adaptive Multimodal Retrieval-Augmented Generation”. In: *arXiv preprint arXiv:2410.11321* (2024) (cit. on p. 11).

[Zha+23a] Yue Zhang, Yafu Li, Leyang Cui, et al. “Siren’s song in the AI ocean: a survey on hallucination in large language models”. In: *arXiv preprint arXiv:2309.01219* (2023) (cit. on p. 8).

[Zha+23b] Ruochen Zhao, Hailin Chen, Weishi Wang, et al. “Retrieving multimodal information for augmented generation: A survey”. In: *arXiv preprint arXiv:2303.10868* (2023) (cit. on pp. 11, 12, 17).

[Zho+22] Shuyan Zhou, Uri Alon, Frank F Xu, et al. “Docprompting: Generating code by retrieving the docs”. In: *arXiv preprint arXiv:2207.05987* (2022) (cit. on p. 12).

List of Figures

List of Tables

Appendix

A

Image instance	Question/Query
1082	What outfits feature black shorts?
42527	What are some jumpsuits with a high waist?
39373	What are some dress options that feature a round neckline?
34524	What are some short-sleeved tops?
38547	What are some fashionable dress options with a high waist?
21806	What casual dress styles with short sleeves and round necklines are available for warm weather?
41097	What formal dresses have a high-low silhouette?
5434	Featuring a black sleeveless top with a scoop neckline, what would you recommend?
889	What are some outerwear options that feature a fur collar?
48024	What are some light pink knitted sweaters?
23951	What are some fashionable outfits that include a red skirt that reaches just above the knees?
34371	Can you recommend any dresses with a heavy, possibly satin or silk fabric?
26553	What are some fashionable dress options with a floral print?
39479	What are some clothing options that include a black tank top with a unique graphic design on the front?
17925	What are some outfits that include a black mesh skirt?
24171	What are some dresses with a fitted bodice?
6829	What casual and comfortable outfit features blue denim jeans with a faded wash?
4014	What are some dress shirts without any special manufacturing techniques or non-textile materials?

36802	What are some clothing options that include a watch on the left wrist?
4174	What fashionable ensemble would you recommend for a high-waisted, gold embellished skirt?
30775	What are some fashionable outfit options for a slim-fit light blue suit?
32118	What is a leopard print blouse with a high neckline?
39494	What casual yet stylish outfits would you recommend featuring orange pants with a straight cut and a flat front?
1069	Dark blue jeans, what would you recommend?
30081	What light blue button-up shirts are available?
14081	What are some outfits featuring sandals with a strap across the foot?
8082	What are some leather outfits with a belted waist?
21051	What are some outfits that include a black belt?
49508	What are some long-sleeved sweatshirts?
14325	What are some jumpsuits with symmetrical silhouette?
1352	What are some dress options suitable for daytime events?
81	What are some outfits that include a black leather corset with a lace-up front?
6051	What are some clothing options that include black high heels?
37671	What are some clothing options that include a single-breasted blazer?
15166	Dark shoes or boots, what would you recommend?
14193	What are some wholebody dresses with a sequin finish?
49546	What are some upperbody garments with a plain pattern?
12045	What are some outfits suitable for a fashion show or similar event?
6720	What fashion ensemble would you recommend for a formal event?
33263	What are some wholebody dresses with a classic and elegant style?
2460	What are some clothing options that have a slash pocket?
26441	What casual hoodies are available?
20880	What are some fashionable and coordinated outfits that include a black handbag?

2418	High-heeled sandals with a strappy design?
47279	What casual and comfortable outfits include a bright red long-sleeved sweater?
45305	What are some edgy t-shirts?
27640	What are some t-shirts with a floral print?
19870	What are some attire options that pair with a dark shirt and brown leather shoes?
11359	What are some fashionable outfits that include a black skirt that reaches just above the knee?
276	What are some options made of a smooth fabric?
30229	What casual spring or summer outfits would you recommend for a person with a set-in sleeve?
967	What type of casual athletic outfit would be suitable for a workout setting?
26357	What are some tops made of lightweight fabric?
2535	What casual, urban-style outfits would you recommend for a relaxed setting?
11368	What are some casual and relaxed outfits suitable for a warm day outdoors?
21672	What type of formal wear dresses would you recommend for events like weddings, galas, or high-profile social gatherings?
10979	What are some jumpsuits with a bold, patterned design?
37384	What are some evening gowns with a fitted bodice?
204	What are some outfits featuring black high-waisted pants with a fly opening and peg silhouette?
12907	What are some outfit recommendations that include black shorts with a smooth finish?
48528	What traditional-style, long-sleeved jackets are available online?
34867	What dresses could be paired with black high heels?
32243	What casual, summery outfits with a polka dot pattern would you recommend for a warm day out?
29033	What formal or evening gowns have a fitted bodice with a sweetheart neckline?
31567	What are some navy blue cardigans with a V-neckline?

11241	What casual, summery outfits with a flared skirt and a fitted top would complement a red headband?
32496	What formal or semi-formal dresses with a sleeveless design would you recommend for an occasion?
36098	What are some formal dress options that feature a deep navy color?
9747	What type of sneakers would you recommend for a casual, summery outfit?
42949	What are some black jackets with a classic design?
31097	What are some dress options made of satin or similar smooth fabric?
27229	What are some fashionable denim jumpsuits?
26932	What are some fashionable and stylish dress options for a formal or semi-formal event?
16304	What color pattern would you recommend for a runway show?
40627	What casual pants with a straight cut would you recommend?
44639	What are some fashionable dresses with a bow neckline?
39575	What are some dress options that are suitable for smart-casual wardrobes?
3500	What are some clothing options that include a black and white patterned top with a round neckline?
50164	What color trousers would complement a classic tuxedo?
6129	What sunglasses would you recommend for a chic, bohemian look?
44775	Can you recommend any dresses with a coral or peach color and a sparkling, embellished look?
39515	What men's suit jackets with a welt pocket are available online?
45006	What casual graphic t-shirts are available for babies or toddlers?
36685	Can you recommend a beige blouse with a ruffled neckline as part of a fashionable outfit?
11571	What are some clothing options that include a light beige blazer with a notched collar and a front zipper?

28528	What are some professional clothing options for a single-button, notch lapel suit?
50016	What hoodies are suitable for everyday wear or casual outings?
29005	What are some North Face jackets?
14200	What casual outfit would you recommend that includes a white tank top?
30547	What are some midi length lower body pants?
43170	What are some fashionable and trendy upperbody outfits that include a black and white faux fur vest?
49984	What type of formal and professional attire would you recommend for a man?
45098	What are some short-sleeved polo shirts?
50336	What comfortable hoodies have a drawstring hood?
37321	What are some long-sleeved tops with a crew neckline?
44497	What are some tank tops made of cotton or cotton blend?
16047	What sunglasses would you recommend for a pink brick wall backdrop?
22278	Can you recommend brown suede boots?
22395	What are some fashionable and chic coats?
15494	What are some dresses with a cinched waist?

Tab. A.1.: Evaluation sample set for retrieval methods where a sample instance is represented by its IRI (here using only the instance number for simplicity) and the respective question/query.

Image instance	Question	BM25 Rank	BM25 + LLM Rank	E-based Rank	E-based + LLM Rank
48024	What are some light pink knitted sweaters?	22957	50	817	**7**
34371	Can you recommend any dresses with a heavy, possibly satin or silk fabric?	35499	**9**	1996	59
26553	What are some fashionable dress options with a floral print?	290	**10**	28	15
49508	What are some long-sleeved sweatshirts?	32393	4962	1615	**146**
81	What are some outfits that include a black leather corset with a lace-up front?	542	**0**	58	**0**
26441	What casual hoodies are available?	1238	25375	272	3
32243	What casual, summery outfits with a polka dot pattern would you recommend for a warm day out?	253	**2**	6823	159
29005	What are some North Face jackets?	1150	**0**	599	**0**
40627	What casual pants with a straight cut would you recommend?	9180	99	19	**1**
26932	What are some fashionable and stylish dress options for a formal or semi-formal event?	588	**528**	15593	7692

Tab. A.2.: Partial selection of sample instances for retrieval models where each selected instance is represented by its IRI (here using only the instance number for simplicity) and the respective question/query. The rest of the columns show the rank of the highest scoring relevant instance evaluated in the first benchmark dataset for each retrieval model-KG combination.+*LLM* indicates that the retrieval was done in the LLM-enriched KG. The highest rank for each instance is marked in bold. Ranking numbers begin at 0 because Python lists use a zero-based indexing system.

Image instance	Question	BM25 Rank	BM25 + LLM Rank	E-based Rank	E-based + LLM Rank
48024	What are some light pink knitted sweaters?	236	1	8	**0**
34371	Can you recommend any dresses with a heavy, possibly satin or silk fabric?	1030	9	**1**	59
26553	What are some fashionable dress options with a floral print?	**9**	10	28	15
49508	What are some long-sleeved sweatshirts?	418	3	65	**2**
81	What are some outfits that include a black leather corset with a lace-up front?	105	**0**	58	**0**
26441	What casual hoodies are available?	192	**1**	6	**1**
32243	What casual, summery outfits with a polka dot pattern would you recommend for a warm day out?	16	**2**	9	9
29005	What are some North Face jackets?	91	**0**	**0**	**0**
40627	What casual pants with a straight cut would you recommend?	7	32	2	**0**
26932	What are some fashionable and stylish dress options for a formal or semi-formal event?	46	**24**	213	244

Tab. A.3.: Partial selection of sample instances for evaluation of retrieval models where each selected instance is represented by its IRI (here using only the instance number for simplicity) and the respective question/query. The rest of the columns show the rank of the highest scoring relevant instance evaluated in the second benchmark dataset for each retrieval model-KG combination. +*LLM* indicates that the retrieval was done in the LLM-enriched KG. The highest rank for each instance is marked in bold. Ranking numbers begin at 0 because Python lists use a zero-based indexing system.

Image instance	**|LP|=40 Rank**	**|LP|=40 + LLM Rank**	**|LP|=20 Rank**	**|LP|=20 + LLM Rank**	**|LP|=10 Rank**	**|LP|=10 + LLM Rank**
48024	1	**0**	**0**	**0**	**0**	**0**
34371	61	38	**15**	126	16	169
26553	107	163	71	85	65	**5**
49508	20	**0**	24	1	6	**0**
81	60	**0**	239	**0**	212	7
26441	**0**	547	3	3	4	2
32243	3	5	6	**1**	12	2
29005	2	**0**	10	**0**	9	**0**
40627	**0**	2	**0**	134	2	16
26932	69	**2**	89	19	99	865

Tab. A.4.: Partial selection of sample instances for evaluation of retrieval-augmented method of the second RAG model where each selected instance is represented by its IRI (here using only the instance number for simplicity). Question/query is omitted for space purposes. The rest of the columns show the rank of the highest scoring relevant instance evaluated in the second benchmark dataset for each |LP|-KG combination we used during our evaluation. LP denotes learning problem: LP = $E^+ \cap E^-$ where in this case $|E^+| = |E^-|$. *+LLM* indicates that the initial retrieval was done in the LLM-enriched KG. The initial retrieval was done using the embedding-based retriever. Class expression learning happens in the second variation of the second KG using TDL. The highest rank for each instance is marked in bold. Ranking numbers begin at 0 because Python lists use a zero-based indexing system.

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I declare that I have completed this work solely and only with the help of the references mentioned in this work.

Paderborn, Januar, 2025

Alkid Baci