




# LLsiM: Large Language Models for Similarity Assessment in Case-Based Reasoning<sup>\*</sup>

Mirko Lenz<sup>1,2</sup> , Maximilian Hoffmann<sup>1,2</sup> , and Ralph Bergmann<sup>1,2</sup> 

<sup>1</sup> Artificial Intelligence and Intelligent Information Systems, Trier University,  
Universitätsring 15, 54296 Trier, Germany, [www.wi2.uni-trier.de](http://www.wi2.uni-trier.de)  
[info@mirko-lenz.de](mailto:info@mirko-lenz.de), [{hoffmannm,bergmann}@uni-trier.de](mailto:{hoffmannm,bergmann}@uni-trier.de)

<sup>2</sup> German Research Center for Artificial Intelligence (DFKI)  
Behringstraße 21, 54296 Trier, Germany, [ebis.dfki.de](mailto:ebis.dfki.de)  
[{mirko.lenz,maximilian.hoffmann,ralph.bergmann}@dfki.de](mailto:{mirko.lenz,maximilian.hoffmann,ralph.bergmann}@dfki.de)

**Abstract.** In Case-Based Reasoning (CBR), past experience is used to solve new problems. Determining the most relevant cases is a crucial aspect of this process and is typically based on one or multiple manually-defined similarity measures, requiring deep domain knowledge. To overcome the knowledge-acquisition bottleneck, we propose the use of Large Language Models (LLMs) to automatically assess similarities between cases. We present three distinct approaches where the model is used for different tasks: (i) to predict similarity scores, (ii) to assess pairwise preferences, and (iii) to automatically configure similarity measures. Our conceptual work is accompanied by an open-source Python implementation that we use to evaluate the approaches on three different domains by comparing them to manually crafted similarity measures. Our results show that directly using LLM-based scores does not align well with the baseline rankings, but letting the LLM automatically configure the measures yields rankings that closely resemble the expert-defined ones.

**Keywords:** Case-Based Reasoning · Large Language Models · Similarities · Ranking · Evaluation.

## 1 Introduction

Case-Based Reasoning (CBR) [1,29] is based on the core principle that similar problems have similar solutions [28]. Given some new problem, CBR systems determine the most relevant past cases and reuse their solutions with the goal of maximizing the *utility* of the solution for the user. In real-world applications, the notion of utility is hard to assess objectively and thus often approximated via the concept of *similarity*. The computation of similarities within parts of a CBR application (e.g., during case retrieval) is conducted via similarity *measures*. Depending on the complexity of the case representation, defining an appropriate similarity *configuration* is a challenging task that may involve domain knowledge and close collaboration with domain experts. This *knowledge-acquisition bottleneck* can potentially hinder the deployment of CBR systems [13].

---

<sup>\*</sup> The Version of Record is available online: [doi.org/10.1007/978-3-031-96559-3\\_9](https://doi.org/10.1007/978-3-031-96559-3_9)

At the same time, the field of Machine Learning (ML) has made significant advancements in recent years, particularly with the rise of Deep Learning (DL) and Large Language Models (LLMs) [8]. These models are widely used in various domains and often serve as baselines for the development of new ML techniques. With CBR being a problem-solving methodology that is open for new developments, such techniques have already been investigated by the community—for instance, as an embedding-based similarity method for texts [6] or an explanation technique for procedural cases [24]. Such hybrid approaches allow to combine the powerful text generation and understanding capabilities of LLMs with the structured domain knowledge and symbolic reasoning capabilities of CBR systems. Compared to traditional supervised ML approaches that require extensive amounts of labeled training data, LLMs use few-shot or even zero-shot learning to generate predictions based on a few examples or even just the prompt. This means that the information available in a CBR system, such as the case base and the vocabulary, can be used to guide the model in generating the desired output, accelerating the development of robust CBR systems.

This paper aims to explore the potential of LLMs for similarity assessment and case retrieval in CBR systems. We see an opportunity in using LLMs to automatically assess similarities between cases, thereby reducing the knowledge-acquisition effort. Besides directly generating similarity scores, the models may also be valuable for configuring established CBR applications—for instance, by selecting appropriate measures for certain attributes. Preliminary work in this direction has already been done by Wilkerson and Leake [36], who conducted several experiments with LLMs to assess similarities in a CBR system. While their results indicate limited usefulness of LLMs for this task, we argue that integrating more advanced prompting techniques and abstracting from traditional similarity-based retrieval can enhance their effectiveness—especially when dealing with different model sizes. To address this research gap, our paper contains the following contributions: (i) Three distinct retrieval strategies tailored for different types of LLMs, (ii) an experimental evaluation on three different domains with a selection of proprietary and open-weight models, and (iii) an open-source Python implementation of the approaches with a command-line interface.

The remainder of this paper is structured as follows: Section 2 introduces foundations regarding similarity measures, followed by an overview of related work in Section 3. Section 4 presents our approaches that are evaluated in Section 5. Finally, Section 6 concludes the paper and presents future work.

## 2 Foundations

In this section, we provide an overview of the basic concepts relevant to our work. After a brief introduction to traditional similarity-based retrieval and its potential shortcomings in Section 2.1, we discuss alternative retrieval techniques involving pairwise case preferences and rankings in Section 2.2. Additionally, we present the potential of LLMs to reduce the knowledge-acquisition effort in CBR systems by automatically learning similarity measures from data in Section 2.3.

## 2.1 Similarity-Based Retrieval

This paper takes a more general view on the retrieval process—simply speaking, as a selection of the most relevant cases of a case base w.r.t. to a query. In the traditional CBR process, the retrieval step selects the most relevant cases from the case base based on their similarity to the query. Thereby, the similarity between cases acts as a proxy for their utility which, in turn, determines the reusability of a case to solve the current problem [28,29].

This traditional approach comes with some shortcomings. One of the main issues is the complex and time-consuming process of defining the similarity measures, known as the knowledge-acquisition bottleneck [13]. It requires a deep understanding of the domain and the use case to define proper similarity measures which is often not available or difficult to obtain. While the knowledge-acquisition bottleneck refers to all types of knowledge in a CBR system [29], it can be particularly pronounced for the similarity knowledge. Additionally, the concrete similarity values themselves come with some challenges that further increase the manual effort [15]. First, they are not directly interpretable, making it difficult to understand the reasoning behind the retrieval process and even hindering domain experts from working with them. Second, the similarity values are often not consistent across different cases, similarity measures, and similarity models, leading to unreliable retrieval results. For example, the similarities of a Levenshtein measure are usually not comparable to scores based on embeddings. Finally, the values are only meaningful in relation to each other—for instance, a case with a similarity of 0.8 is more useful relative to a case with a similarity of 0.4, but may not actually provide twice the utility for solving the problem.

## 2.2 Abstracting from Similarities

There are several ways of overcoming the aforementioned challenges with traditional similarity-based retrieval. One approach is to abstract from similarities by using *preference relations* [15] and, more general, case ranking techniques [11]. Thereby, the goal is to determine the most relevant cases for a given query based on their relative preferences rather than their absolute similarity values. For example, given two cases  $c_i$  and  $c_j$ , the preference relation  $c_i \succ c_j$  indicates that case  $c_i$  is preferred over case  $c_j$  for the given query. This allows for a more intuitive understanding of the retrieval process, as it focuses on the relative importance of cases rather than their absolute similarity values. Determining preferences rather than specific similarity values usually aligns better with the human perception of ranking and retrieval [15], as pairwise similarities can be hard to interpret and only meaningful in comparison to other pairwise similarities. On the other hand, pairwise preference are often interpretable on their own (“A is more relevant than B”) and, thus, easier to determine for domain experts.

As pairwise preferences do not provide a complete ranking of the cases, they need to be aggregated to create a ranked list of cases. This step is usually referred to as *rank aggregation* and is a common task in various fields, including information retrieval, recommender systems, and social choice theory [15]. Rank

aggregation can be performed using various techniques [2]—we want to point out specific rank aggregation techniques from the field of information retrieval for the web which are subsumed under the term *centrality* measures [11]. Centrality measures assess node importance in networks, offering a way to combine multiple rankings into a single consensus. In rank aggregation, items are nodes and their pairwise comparisons form edges, so measures like degree, closeness, betweenness, and eigenvector centrality (e.g., PageRank [25]) reveal consistently influential items. We see strong potential of using centrality measures in CBR, because a list of pairwise case preferences closely resembles a network of nodes and edges and runtime optimizations such as power iteration [25] can be employed for efficient computation even on large case bases.

### 2.3 Automatic Learning with LLMs

Whether the retrieval process is based on similarity values or preferences, the usage of ML techniques can help to reduce the knowledge-acquisition effort in CBR systems. Several approaches have been proposed in the past [23,37,14] with the most recent and prominent approaches being based on DL techniques—more specifically, on LLMs [36,7,19,35].

LLMs are neural networks specifically designed to process and generate natural language. A comprehensive introduction to the topic is beyond the scope of this paper—instead, we give a brief overview. These models are built on the transformer architecture, leveraging the attention mechanism to process an input sequence of text [33]. OpenAI’s Generative Pre-trained Transformer (GPT) [27] series, widely recognized through ChatGPT, employs a decoder-only architecture that generates the next token based on the preceding sequence. Rather than fine-tuning the model for specific tasks, prompting techniques can be used to steer the model toward the desired output. This can be achieved through few-shot learning, where examples of input-output pairs are provided, or zero-shot learning, which relies solely on user input without example outputs [8].

Among other aspects, LLMs can help mitigate the knowledge-acquisition bottleneck by capturing domain expertise from large text corpora. Their learned representations allow CBR engineers to create, refine, or expand case structures, vocabularies, and similarity measures. This reduces the need for deep domain knowledge and accelerates the development of robust CBR systems.

## 3 Related Work

Research on the synergies between CBR and LLMs is a prominent topic in the CBR community, resulting in dedicated workshops at ICCBR 2024 and 2025 and a research manifesto of many researchers from the community [3]. One of the motivations is to combine the strengths of LLMs and CBR methods to overcome the respective weaknesses based on the concept of neuro-symbolic Artificial Intelligence (AI) [30,18]. Most approaches can either be categorized as LLMs improving CBR or CBR improving LLMs [3]. Approaches of the first category are mostly

focused on the knowledge engineering process, specifically using the rich capabilities of LLMs to deal with textual data—for instance, modeling of planning knowledge [7], adaptation of arguments [19], or case elicitation [35]. Approaches of the second category are mostly focused on integrating the symbolic knowledge and reasoning capabilities of CBR systems into the text generation process of LLMs—for instance, to enhance LLMs with a Retrieval-Augmented Generation (RAG) [21] process guided by a CBR system [38], to use the CBR knowledge as persistent memory for LLMs improving their performance in specific tasks [34], and other use cases [32,24,12]. Our paper mostly fits in the former category, addressing opportunities 5.4, 6.1, and 6.2 of Bach et al. [3].

The work of Wilkerson and Leake [36] is the one most closely related to our work. They discuss the use of LLMs for similarity-based retrieval and case adaptation and compare several scenarios where they vary the information that is provided to the model. This information contains the case features but, depending on the prompting strategy, also the best results of a  $k$ -NN retrieval conducted before prompting. The experiments show that LLMs generally perform poorly in similarity assessment and case ranking but additional knowledge such as the  $k$ -NN retrieval results can improve their performance. Our approach differs from this work mainly in two aspects: First, we take a more general view on the retrieval process, simply speaking, as a selection of the most relevant cases of a case base w.r.t. to a query. More specifically, we abstract the task of the LLM from predicting similarities to predicting pairwise preferences, rankings and similarity measures, which is the base of the three LLM-based approaches we propose. Second, we focus strongly on reducing the knowledge-acquisition effort and, thus, design the approaches to function without the need for manually defined similarity measures. While a hybrid approach as proposed by Wilkerson and Leake [36] is a valid idea—for instance, if an existing CBR system is to be improved, we want to reduce the effort of defining the similarity measures in the first place. This lowers the barrier of entry for inexperienced CBR engineers and is especially important for domains where the knowledge-acquisition bottleneck is particularly strong. The common problem of LLMs returning inconsistent or even contradictory information is also considered in our work: We propose a method that outputs a similarity configuration which can then serve as a “first draft” for experienced CBR engineers to come to a final configuration faster.

## 4 Assessing Similarities via LLMs

Our approaches for assessing similarities using LLMs are motivated by the shortcomings of traditional similarity-based retrieval (see Figure 1a): CBR engineers analyze the respective domain with the available cases and possibly consult domain experts to come up with the required definitions. It may happen that multiple iterations are needed to find a suitable configuration—potentially delaying the deployment of the CBR system if the domain is not well understood or the domain expert is not available. All of this can be done *offline* with the acquired knowledge being incorporated into the application. For subsequent queries, the

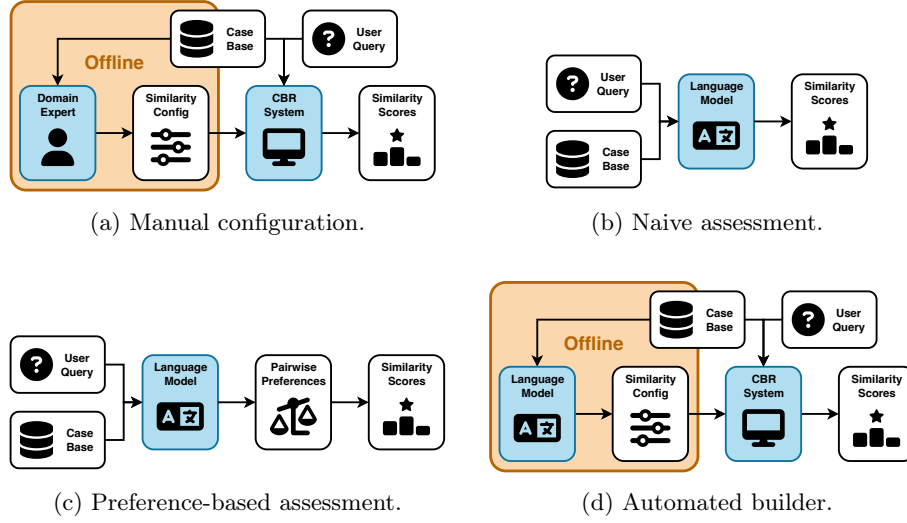


Fig. 1: Approaches for assessing similarities using LLMs.

CBR system then uses this configuration to compute the similarities to all available cases in an *online* phase. The runtime of this process depends on the complexity of the underlying similarity measures.

#### 4.1 Naive Similarity Assessment

The naive way of introducing LLMs into the traditional similarity-based retrieval process is to let them *directly* predict the similarities between the query and all available cases (see Figure 1b). One way of achieving that is to send query-case pairs to the LLM and let it predict some score for each pair—just like the idea of Wilkerson and Leake [36]. However, requesting individual similarity scores may not lead to a sensible ranking of the cases because the model does not have any information about the other cases in the case base. To solve this, we use a feature called *structured output* that is available in some LLMs such as those offered by OpenAI.<sup>3</sup> This allows us to provide a JSON schema of the expected response format that we want the model to follow—making it possible to pass the entire case base in the same context as the query. We propose two variants of this approach to deal with the fact that generating floating point numbers is not a trivial task for LLMs: (i) In the *similarity-based* variant, we define the expected output as a list of similarity scores in the range  $[0, 1]$  for each case in the case base. (ii) In the *ranking-based* variant, we define the expected output as a list of case IDs sorted by their similarity to the query. Since our end goal is to have similarity scores and not only a ranking, we use Equation (1) to convert the ranking to similarity scores. Here,  $\text{rank}(q, c_i)$  is the rank of case  $c_i$  w.r.t. query  $q$

<sup>3</sup> <https://platform.openai.com/docs/guides/structured-outputs>

and  $|C|$  is the number of cases in the case base. As a result of this transformation, the similarity scores are evenly distributed, making it impossible to distinguish between cases with small or large differences in similarity. If capturing these is important, the similarity-based variant should be used instead.

$$\text{sim}(q, c_i) = 1 - \frac{\text{rank}(q, c_i) - 1}{|C| - 1} \quad (1)$$

## 4.2 Preference-Based Similarity Assessment

Due to the aforementioned problems with LLM-generated similarity scores, the second approach is based on the idea of replacing the absolute similarity values with *pairwise preferences* (see Figure 1c) which are well-known in the CBR community [15]. Pairwise preferences add an additional layer of abstraction to the similarity assessment, thus, reducing the complexity of the task from predicting a coherent ranking or similarity scores to correctly predicting which of two cases is more relevant to the query. Enumerating all possible pairs of cases leads to a combinatorial explosion of the requests sent to the LLMs, so we propose two variants of this approach: (i) In the *isolated* variant, we send a triple  $(q, c_i, c_j)$  to the LLM and ask it to predict whether  $c_i$  or  $c_j$  is more relevant to  $q$ . (ii) In the *batched* variant, we send the query and all available cases to the LLM and request the preferences for all pairs of cases. The isolated variant can be applied to any LLM, but the batched variant requires a model that can handle larger contexts and is able to return structured outputs. Similar to the naive approach, the batched variant is prone to missing or extra pairs in the output—the isolated is not because there is exactly one request for each pair. This problem can be mitigated by computing all possible case pairs as an explicit list in advance and appending them to the prompt. In addition, we send retry requests with relevant cases for all missing pairs to not overload the LLM with irrelevant data.

As a list of case pairs with predicted preferences does not provide a complete ranking of the cases, rank aggregation is employed (see Section 2.2). We propose to use graph centrality measures for this purpose that can be adapted to case retrieval by transforming the pairwise preference into a directed graph  $G$  where each case is represented as a node and each preference as a directed edge from the more relevant case to the less relevant one. Depending on the use case, different metrics may be useful to turn this graph into a case ranking—for instance, the link analysis algorithms PageRank [25] and HITS [17]. These have been developed for ranking web pages, so they are well-suited to deal with incomplete or even contradictory information. This graph representation also allows for another optimization of the batched variant: After the first request to the LLM, we check for implicit/transitive preferences in the response and explicitly add them to the graph. For instance, if the LLM predicts the preferences  $c_i \succ c_j$  and  $c_j \succ c_k$ , we add  $c_i \succ c_k$  to the graph. This reduces the number of pairs to be sent to the LLM in the second request, making it faster and cheaper. We repeat the inference process after the second request to avoid sparse data and consequently produce a more complete graph.

### 4.3 Automated Similarity Builder (SimBuilder)

All techniques discussed above are only involved in the online phase, meaning that the LLM is invoked for each individual query. This does not match the traditional way of configuring similarity measures in CBR systems, where the configuration is done once and reused for all queries. To bridge this gap, we propose a third approach called SIMBUILDER shown in Figure 1d where the domain expert is replaced by an LLM that is instructed to create a similarity configuration based on the available cases—enabling to shift the LLM inference to an offline phase. It is assumed that there exists a set of simple, atomic similarity measures in the underlying CBR system. The LLM is tasked to select, combine, and parameterize these measures based on the available cases, creating an elaborate similarity measure on the case level. The two main challenges are to expose the available similarity measures to the LLM in an understandable manner and to represent the necessary parameters—such as min/max values for interval-based measures—for each function. The bridge between the underlying CBR system and its similarity metrics and the model can be achieved via *function/tool calling* as offered by an increasing number of LLMs. Similar to the structured outputs used in some of the other approaches, this feature allows exposing functions to the model in a structured way and receive parameters to call them locally—these can be different for each similarity measure and control their behavior. For instance, an interval-based measure could have min/max values as parameters, while a taxonomy-based measure could expect a tree structure as input. The central difference to a structured output is that function calling allows to provide multiple JSON schemas in the same request and let the model decide which one to use. We filter the available measures based on the case representation since it would not make sense to use a taxonomy-based measure for numerical values or an edit distance for textual attributes. Then, the LLM is prompted to select the best function and predict values for all available parameters of the method—allowing to execute it locally later in the online phase. For example, given an attribute “miles” of a car, the LLM could decide to use an interval-based similarity measure and provide the parameters “min” and “max” as 0 and 100,000, respectively. This enables the model to create a more tailored similarity configuration that aligns with the specific characteristics of the data.

## 5 Experimental Evaluation

Having introduced the different techniques for using LLMs for assessing the similarity in CBR applications, we evaluate them in this section. We first discuss our hypotheses, outline the experimental setup—including descriptions of the three domains at hand—and then present the results of our experiments. The overall goal of our experiments is to assess the *effectiveness* (i.e., accurate similarities) and *efficiency* (i.e., fast computations and low cost) of the proposed approaches. Thus, we compare the LLM-based approaches to the traditional way of configuring similarity measures in CBR systems.



**H1.** The naive ranking method is more effective than the naive similarity method as LLMs typically struggle to generate precise numerical values.

**H2.** The preference-based method is more effective than both naive methods because pairwise assessments are more tolerant to potential errors introduced by LLMs. Moreover, the batched variant is more effective than the isolated one as the larger amount of context allows for better informed decisions.

**H3.** The SIMBUILDER approach is more efficient than the naive and preference-based methods because the LLM inference can be done in advance. It is also more effective because it can be used with larger LLMs that better understand the semantics of the cases provided.

## 5.1 Experimental Setup

In this section, we describe the experimental setup of our experiments—that is, the selection of datasets, evaluation metrics, and language models as well as notes regarding our publicly available implementation.

**Datasets.** A total of three datasets is used in our experiments, each representing a different domain and case representation. The *cars* corpus [22] with 100 cases uses a simple attribute-value representation with numerical and textual attributes. The *recipes* corpus [5] with 40 cases contains semantic graphs representing cooking recipes. The graphs consist of three different types of nodes with different attributes. Finally, the *argumentation* corpus [26] with 110 cases contains argumentative microtexts represented as semantic graphs according to the Argument Interchange Format (AIF) [10] standard. While the first two datasets do not provide a distinct set of queries, the argumentation dataset contains 24 queries and corresponding gold standard relevance assessments [6]. The argumentation dataset is also notable for its heavy use of larger textual attributes, making it infeasible to use taxonomy-based or edit-based distances and instead relying on embeddings.

**Evaluation Metrics.** In our work, we are interested in the difference between multiple rankings, so standard metrics like precision and recall are not suitable. Instead, we use completeness (CP) and correctness (CR) as proposed by Cheng et al. [9] which are based on comparing available *orders* in the form of concordant and discordant pairs.  $CP \in [0, 1]$  measures how many elements of the ground truth ranking are also in the generated ranking while  $CR \in [-1, 1]$  indicates whether the generated ranking aligns well (1) or poorly (0) with the ground truth ranking, or even contradicts it (-1). In addition, we compute the normalized cumulative discounted gain (nDCG) [16] by using the inverse ranks as query relevance scores. Compared to CR, this metric incorporates the rank of the cases, meaning that higher importance is given to the top-ranked cases. Another relevant factor is the runtime  $t$  of the experiments.

To compute these metrics, we need a ground truth ranking of the cases for each query. As such an expert ranking is not available for some of our datasets, we use the *manual similarity* configuration to obtain the ground truth rank( $q, c_i$ ) for each query. In addition, we compute a variant where the rank is normalized

Table 1: Overview of the different approaches and the characteristics of the tested models. Parameters are measured in billions and price in USD per million input tokens as reported by OpenRouter.ai.

Size	Params	Price	Models	Approaches
Small	< 4	< 0.01	Llama-3.2-3b	preferences-isolated
Medium	40–120	0.1–0.2	GPT-4o-mini, Llama-3.3-70b	naive, preferences-batched
Large	> 400	> 0.8	GPT-4o, o3-mini, DeepSeek-V3, DeepSeek-R1, Llama-3.1-405b	SIMBUILDER

to the range  $[0, k]$  to ignore smaller and less relevant differences in the ranking via Equation (2) where  $\lfloor \cdot \rfloor$  denotes rounding to the nearest integer.

$$\text{rank}'_k(q, c_i) = \left\lfloor \frac{\text{sim}(q, c_i) - \min \text{sim}(q, C)}{\max \text{sim}(q, C) - \min \text{sim}(q, C)} \times k \right\rfloor \quad (2)$$

**Language Model Selection.** For our experiments, we distinguish between three classes of LLMs—small, medium, and large—as shown in Table 1. Depending on the specific use case at hand, these boundaries may be adjusted—here they serve as categories to select cost-effective models for the different approaches. In addition to the model size, we also report the price as it is a crucial aspect for real-world applications. Initial experiments showed that preferences-isolated is the most expensive and slowest approach—for each query, all possible query-case pairs are requested separately—so we use a single small model to evaluate it. For the preferences-batched and both naive approaches, the case base is sent once to the model for each query, so we can use medium-sized LLMs for them (but not large ones due to the cost). The SIMBUILDER approach, on the other hand, can be used with larger models. Here, the case base is sent to the LLM once to create the similarity configuration in the offline phase.

**Implementation.** We provide an open-source implementation<sup>4</sup> in Python that builds on top of CBRKIT [20]. It offers a declarative way of configuring similarity measures and supports various case representations, including attribute-value and graphs. To convert pairwise preferences to similarity scores, we use the arithmetic mean of the two centrality measures PageRank and HITS. Initial experiments showed that sparse graphs as returned by some LLMs lead to many identical similarity scores, meaning that the ranking of the affected cases is arbitrary—combining multiple centrality measures helped mitigate this problem. We also experimented with individual centrality measures, but did not observe large differences in the results. For the SIMBUILDER, we convert the available similarity functions to JSON in a generic way—currently supporting attribute-value and graph-based representations. Some of these measures accept rather complex parameters, so we wrap those in a custom function that only exposes

<sup>4</sup> <https://github.com/wi2trier/llsim> (MIT license)

Table 2: Cars results using baseline retrieval as ground truth.

Experiment	CP		CR		nDCG		$t$
	rank	rank' <sub>5</sub>	rank	rank' <sub>5</sub>	rank	rank' <sub>5</sub>	
builder-deepseek-r1	1.000	1.000	0.733	0.907	0.991	0.994	0.463
builder-deepseek-v3	1.000	1.000	0.754	0.920	0.991	0.995	0.485
builder-gpt-4o	1.000	1.000	0.802	0.941	0.994	0.996	0.803
builder-llama-405b	1.000	1.000	0.757	0.923	0.993	0.996	0.438
builder-o3-mini	1.000	1.000	0.787	0.943	0.994	0.996	0.562
naive-rank-gpt-4o-mini	0.335	0.342	0.263	0.352	0.586	0.595	15.9
naive-rank-llama-70b	0.745	0.746	0.164	0.235	0.738	0.754	1936
naive-sim-gpt-4o-mini	0.067	0.069	0.157	0.266	0.324	0.339	51.3
naive-sim-llama-70b	0.642	0.640	0.016	0.166	0.684	0.693	2704
preferences-gpt-4o-mini	0.868	0.871	-0.018	-0.025	0.799	0.807	462
preferences-llama-3b	0.980	0.980	0.069	0.100	0.855	0.874	8673
preferences-llama-70b	0.385	0.385	-0.351	-0.333	0.386	0.396	4128

Table 3: Recipes results using baseline retrieval as ground truth.

Experiment	CP		CR		nDCG		$t$
	rank	rank' <sub>5</sub>	rank	rank' <sub>5</sub>	rank	rank' <sub>5</sub>	
builder-deepseek-r1	1.000	1.000	0.669	0.854	0.980	0.991	149
builder-deepseek-v3	1.000	1.000	0.438	0.623	0.927	0.953	133
builder-gpt-4o	1.000	1.000	0.713	0.883	0.982	0.992	198
builder-llama-405b	1.000	1.000	0.658	0.841	0.975	0.989	173
builder-o3-mini	1.000	1.000	0.663	0.832	0.976	0.988	197
naive-rank-gpt-4o-mini	0.704	0.702	-0.051	-0.061	0.719	0.745	13.5
naive-rank-llama-70b	0.866	0.867	-0.039	-0.033	0.734	0.751	844
naive-sim-gpt-4o-mini	0.594	0.597	-0.136	-0.126	0.648	0.683	36.0
naive-sim-llama-70b	0.827	0.821	-0.063	-0.047	0.705	0.717	1185
preferences-gpt-4o-mini	0.940	0.948	-0.000	0.001	0.799	0.811	305
preferences-llama-3b	0.949	0.954	-0.030	0.020	0.785	0.810	857
preferences-llama-70b	0.809	0.813	-0.139	-0.141	0.696	0.702	4068

a subset to the LLM. In case a request fails—for instance, due to an invalid response—we retry the generation.

## 5.2 Results and Discussion

Based on the setup described in the previous section, we now present the results of our experiments for the three domains, starting with those using the baseline retrieval as ground truth. For all datasets combined, the cost of our experiments for the naive and preference-based techniques totaled to a few dollars per model. Even though using the larger models, the SIMBUILDER was actually cheaper and used less than one dollar per model. Its offline runtime—which is not included in the tables as it is upfront and does not affect the actual retrieval—ranges from 5 to 500 seconds depending on the domain and model.

**Baseline Retrieval.** The results for the cars and recipes domains are shown in Table 2 and Table 3, respectively. For the naive and preference-based approaches, the completeness is less than one, indicating missing or invalid LLM responses. The cars dataset (which contains more than double the number of cars compared to the recipes dataset) yields completeness scores below 0.1, meaning that the model excluded most of the cases from the output. The correctness scores for

Table 4: Argumentation results using expert rating as ground truth.

Experiment	CP	CR	nDCG	$t$
baseline	1.000	0.220	0.899	147
builder-deepseek-r1	1.000	0.220	0.899	130
builder-deepseek-v3	1.000	0.220	0.899	166
builder-gpt-4o	1.000	0.220	0.899	142
builder-llama-405b	1.000	0.220	0.899	145
builder-o3-mini	1.000	-0.836	0.379	30.2
naive-rank-gpt-4o-mini	0.417	0.008	0.338	27.8
naive-rank-llama-70b	0.703	0.101	0.402	822
naive-sim-gpt-4o-mini	0.260	-0.472	0.302	47.5
naive-sim-llama-70b	0.680	-0.159	0.318	733
preferences-gpt-4o-mini	0.935	0.055	0.335	551
preferences-llama-3b	0.986	-0.142	0.338	2547
preferences-llama-70b	0.628	0.014	0.283	4941

these techniques are mostly close to zero, indicating random rankings. Overall, the scores are higher for the less complex cars dataset—the graph-based representation of the recipes could make it harder for the LLM to understand the semantics of the cases. The SIMBUILDER approach however achieves a completeness of 1.0 for all models and a correctness of around 0.75 for the cars dataset and 0.5 for the recipes dataset. As such, the generated similarity configuration seems to closely match the expert-driven definitions. We consider this to be a very promising result, especially given the fact that both domains make use of rather complex taxonomy-based similarity functions that require a lot of domain knowledge to configure. Analyzing the effect of  $\text{rank}'_5$  (i.e., not requiring strict adherence to the baseline), we see overall improved scores for all approaches. The metrics for the argumentation domain are missing here as we focus on the expert ratings in the next section for this corpus. We still ran the experiment for this corpus and observed similar results as for the recipes dataset.

**Expert Ratings.** The results for the argumentation domain shown in Table 4 are based on expert ratings as ground truth. Consequently, we do not report the  $\text{rank}'$  scores as no transformation from similarity scores to ranks is needed. We observe that even the manually defined *baseline* configuration does not achieve a correctness or nDCG of 1.0—meaning that in real-world scenarios, it may not be possible to achieve a perfect ranking of the cases, even with lots of domain knowledge. Overall, the findings are consistent with the previous results with one notable exception: When used with the reasoning model o3-mini, the SIMBUILDER achieves the worst correctness score of all approaches. All other SIMBUILDER experiments result in the exact same configuration as the expert one, so the different scores are solely caused by variations of the underlying A\* algorithm (e.g., the order in which mapping candidates are evaluated [4]).

**Hypotheses.** Regarding the hypotheses, we see that the naive ranking method is indeed more effective than the naive similarity method, meaning that H1 can be accepted. However, the preference-based technique is not universally more effective than the naive ones. The isolated variant tends to outperform the batched one, ultimately leading to the rejection of H2. Finally, the SIMBUILDER is more effective than all other approaches, but not necessarily more efficient. For the

simple representation of the cars dataset, the SIMBUILDER is the fastest, but that is not the case for the graph-based recipes and argumentation datasets. Thus, we can only partially accept H3.

## 6 Conclusion, Limitations and Future Work

In this paper, we present three different approaches for using LLMs in similarity-based retrieval of CBR applications. LLMs are used to assess pairwise similarities between cases directly, to assess pairwise preferences between cases, and to generate an expert-like similarity configuration via the SIMBUILDER approach. We evaluate them on three different domains using a selection of LLMs to assess their effectiveness and efficiency. The results show that the SIMBUILDER performs best and is capable of almost perfectly matching the ranking obtained via manually crafted similarity configurations, while the other two approaches struggle to achieve similar results.

We see limitations of our work regarding scaling to large case bases and the availability of atomic similarity measures: First, scaling to larger datasets could be challenging with the limited context size of LLMs—especially for preferences-batched and the naive approaches. This could be tackled through chunking where the case base is split into smaller parts and that are processed separately and then aggregated later on. Such strategies are already implemented in CBRKIT and showed decent results in initial experiments. Second, using the SIMBUILDER approach requires a set of atomic similarity measures to use as building blocks for the generated similarity configuration. This could hinder its applicability in some CBR systems, but there already exist CBR frameworks with a decent amount of such ready-to-use functions [31].

In future work, we plan to extend our work in several ways. Extending the evaluation to other domains and case representations to assess the generalizability of the approaches is an important next step. We also think that it would be interesting to evaluate the approaches with user-defined similarities or preferences, serving as a more realistic ground truth. Additionally, experimenting with fine-tuned LLMs to improve the performance of the approaches is a possible direction for future work. This could be especially useful for the SimBuilder approach, where the LLM could be fine-tuned on different case representations and similarity models to create a very specialized model. As another means of future work, we plan to use the approach as part of a RAG process [38]. Compared to classical RAG, the approach can work on structured data such as databases by using local similarities instead of simply converting the data into strings and relying on a vector store. This would allow us to use the LLM as a generator of the initial solution and the CBR system as a means of refining the solution by retrieving similar cases and adapting them to the current problem.

**Acknowledgments.** This work is funded by the *Federal Ministry of Education and Research* under grant № 02WAZ1745A (SOWEKI), the association *Eifelkreis Digital* with its project KITEi, and the *Studienstiftung*.

## References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning - Foundational Issues, Methodological Variations, and System Approaches. *AI Commun.* (1994)
2. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.* pp. 113–141 (2000)
3. Bach, K., et al.: Case-Based Reasoning Meets Large Language Models: A Research Manifesto For Open Challenges and Research Directions (2025), <https://hal.science/hal-05006761>, Working Paper
4. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. *Information Systems* pp. 115–127 (2014)
5. Bergmann, R., Grumbach, L., Malburg, L., Zeyen, C.: ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In: *Workshops Proceedings for the Twenty-seventh ICCBR*. pp. 156–161. CEUR (2019)
6. Bergmann, R., Lenz, M., Ollinger, S., Pfister, M.: Similarity Measures for Case-Based Retrieval of Natural Language Argument Graphs in Argumentation Machines. In: *Proceedings of the 32nd FLAIRS Conf.* pp. 329–334. AAAI Press (2019)
7. Brand, F.: Modeling and Acquiring Knowledge with Large Language Models: A Study in the Cyber-Physical Domain. Master’s thesis, Trier University (2024)
8. Brown, T., et al.: Language Models are Few-Shot Learners. In: *Advances in Neural Information Processing Systems*. pp. 1877–1901. Curran Associates, Inc. (2020)
9. Cheng, W., Rademaker, M., De Baets, B., Hüllermeier, E.: Predicting Partial Orders: Ranking with Abstention. In: *Machine Learning and Knowledge Discovery in Databases*. pp. 215–230. Springer (2010)
10. Chesñevar, C.I., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G.R., South, M., Vreeswijk, G., Willmott, S.: Towards an argument interchange format. *The Knowledge Engineering Review* p. 293 (2006)
11. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. pp. 613–622. ACM (2001)
12. Feng, B., Gao, H., Zhang, P., Zhang, J.: Cbr-ren: A case-based reasoning driven retriever-generator model for hybrid long-form numerical reasoning. In: *Case-Based Reasoning Research and Development - 32nd ICCBR*. pp. 111–126. Springer (2024)
13. Hanney, K., Keane, M.T.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: *Case-Based Reasoning Research and Development*, pp. 359–370. Springer (1997)
14. Hoffmann, M., Bergmann, R.: Using graph embedding techniques in process-oriented case-based reasoning. *Algorithms* p. 27 (2022)
15. Hüllermeier, E., Schlegel, P.: Preference-based CBR: first steps toward a methodological framework. In: *Case-Based Reasoning Research and Development - 19th ICCBR 2011, London, UK, September 12-15, 2011*. pp. 77–91. Springer (2011)
16. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* pp. 422–446 (2002)
17. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* pp. 604–632 (1999)
18. Leake, D., Crandall, D.J.: On bringing case-based reasoning methodology to deep learning. In: *Case-Based Reasoning Research and Development - 28th ICCBR 2020, Salamanca, Spain, June 8-12, 2020*. pp. 343–348. Springer (2020)
19. Lenz, M., Bergmann, R.: Case-Based Adaptation of Argument Graphs with WordNet and Large Language Models. In: *Case-Based Reasoning Research and Development*. pp. 263–278. Springer Nature Switzerland (2023)

20. Lenz, M., Malburg, L., Bergmann, R.: CBRkit: An Intuitive Case-Based Reasoning Toolkit for Python. In: Case-Based Reasoning Research and Development. pp. 289–304. Springer Nature Switzerland (2024)
21. Lewis, P., et al.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: Advances in Neural Information Processing Systems. pp. 9459–9474. Curran Associates, Inc. (2020)
22. Malburg, L., Hoffmann, M., Trumm, S., Bergmann, R.: Improving similarity-based retrieval efficiency by using graphic processing units in case-based reasoning. In: Proceedings of the Thirty-Fourth FLAIRS, Florida, USA, May 17-19, 2021 (2021)
23. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. *Prog. Artif. Intell.* pp. 129–143 (2020)
24. Minor, M., Kaucher, E.: Retrieval Augmented Generation with LLMs for Explaining Business Process Models. In: Case-Based Reasoning Research and Development. pp. 175–190. Springer Nature Switzerland (2024)
25. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report, Stanford InfoLab (1999)
26. Peldszus, A., Stede, M.: An Annotated Corpus of Argumentative Microtexts. In: Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation. pp. 801–816. College Publications (2016)
27. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
28. Richter, M.M.: Foundations of similarity and utility. In: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, May 7-9, 2007, Key West, Florida, USA. pp. 30–37. AAAI Press (2007)
29. Richter, M.M., Weber, R.: Case-Based Reasoning: A Textbook. Springer-Verlag (2013)
30. Sarker, M.K., Zhou, L., Eberhart, A., Hitzler, P.: Neuro-symbolic artificial intelligence. *AI Commun.* pp. 197–209 (2021)
31. Schultheis, A., Zeyen, C., Bergmann, R.: An Overview and Comparison of Case-Based Reasoning Frameworks. In: Case-Based Reasoning Research and Development. pp. 327–343. Springer Nature Switzerland (2023)
32. Sourati, Z., Ilievski, F., Sandlin, H.Â., Mermoud, A.: Case-Based Reasoning with Language Models for Classification of Logical Fallacies. In: Proceedings of the Thirty-Second IJCAI. pp. 5188–5196 (2023)
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. In: Advances in Neural Information Processing Systems. Curran Associates, Inc. (2017)
34. Watson, I.: A Case-Based Persistent Memory for a Large Language Model (2023)
35. Weitz-Harms, S., Hastings, J.D., Powell, J.H.: Toward automated knowledge discovery in case-based reasoning. In: Proceedings of the Thirty-Seventh FLAIRS 2024, Sandestin Beach, FL, USA, May 19-21, 2024. AAAI Press (2024)
36. Wilkerson, K., Leake, D.: On Implementing Case-Based Reasoning with Large Language Models. In: Case-Based Reasoning Research and Development. pp. 404–417. Springer Nature Switzerland (2024)
37. Wilkerson, Z., Leake, D., Crandall, D.J.: On combining knowledge-engineered and network-extracted features for retrieval. In: Case-Based Reasoning Research and Development - 29th ICCBR 2021. pp. 248–262. Springer (2021)
38. Wiratunga, N., et al.: CBR-RAG: Case-Based Reasoning for Retrieval Augmented Generation in LLMs for Legal Question Answering. In: Case-Based Reasoning Research and Development. pp. 445–460. Springer (2024)