Interactive Query Answering on Knowledge Graphs with Soft Entity Constraints

Anonymous authors
Paper under double-blind review

Abstract

Methods for query answering over incomplete knowledge graphs retrieve entities that are *likely* to be answers, which is particularly useful when such answers cannot be reached by direct graph traversal due to missing edges. However, existing approaches have focused on queries formalized using first-order-logic. In practice, many real-world queries involve constraints that are inherently vague or context-dependent, such as preferences for attributes or related categories. Addressing this gap, we introduce the problem of query answering with *soft constraints*. We formalize the problem and introduce two efficient methods designed to adjust query answer scores by incorporating soft constraints without disrupting the original answers to a query. These methods are lightweight, requiring tuning only two parameters or a small neural network trained to capture soft constraints while maintaining the original ranking structure. To evaluate the task, we extend existing QA benchmarks by generating datasets with soft constraints. Our experiments demonstrate that our methods can capture soft constraints while maintaining robust query answering performance and adding very little overhead. With our work, we explore a new and flexible way to interact with graph databases that allows users to specify their preferences by providing examples interactively.

1 Introduction

Knowledge graphs (KGs) are graph-structured databases that store information about a domain using triples of the form *(subject, predicate, object)*. Such a compact representation enables efficient algorithms for answering queries about entities over the domain encoded by the KG (Fensel et al., 2020; Hogan et al., 2021). In principle, answers to these queries can be obtained by traversing the graph and collecting the entities that satisfy the given conditions. However, this strategy misses answers due to connections that are not explicit in a KG that may be incomplete, but that are *likely* to satisfy a query, for example because of their similarity to known answers. The problem of finding likely answers to simple queries over KGs has been studied in areas such as rule mining (Galárraga et al., 2013; Meilicke et al., 2018; 2019; Qu et al., 2020) and link prediction (Nickel et al., 2016; Ji et al., 2022), and complex query answering (Hamilton et al., 2018; Daza & Cochez, 2020; Ren et al., 2020; Ren & Leskovec, 2020; Arakelyan et al., 2021; 2023; Bai et al., 2023; Zhu et al., 2022; Ren et al., 2024). Given a query, these methods score all the entities in the KG, indicating the likelihood of meeting all conditions specified in the query. Notably, prior works have focused on conditions that can be specified using different subsets of first order logic.

We identify a relevant and complementary problem: answering queries with constraints that cannot be specified formally as further conditions in a query. We refer to these as *soft constraints*, to contrast them with the *hard* constraints that can be specified using first order logic. Consider the following query: "*What are award nominations received by movies starring Leonardo DiCaprio?*". A user might be additionally interested in nominations "*related to the audio-visual design of the movie*", such as *Best Costume Design*, or *Best Sound*, and not in nominations about the artistic vision of a film (see 1). Such a constraint cannot be specified using the language of first order logic, simply because such an idea of relatedness is too vague to be formalized into precise logical statements. Our goal is thus to broaden the expressivity of queries on KGs which are specified using first order logic, by incorporating additional soft constraints to refine the set of answers.

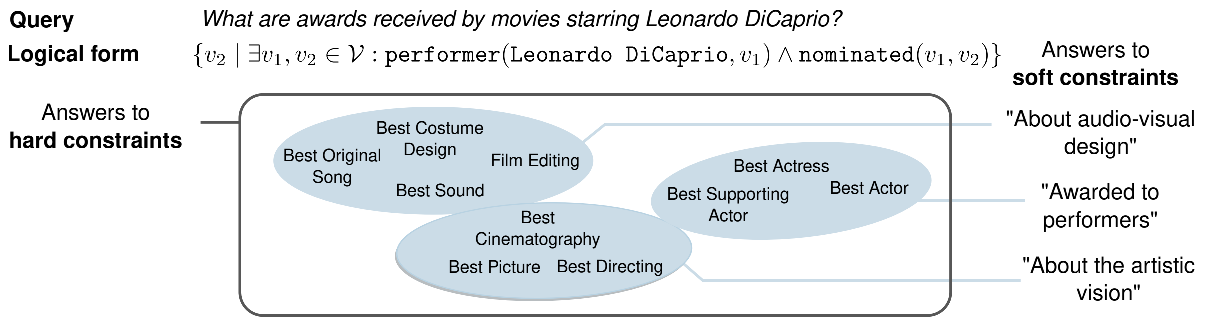
Query
What are awards received by movies starring Leonardo DiCaprio?
Logical form
$\{v_2 \mid \exists v_1, v_2 \in \mathcal{V} : \texttt{performer}(\texttt{Leonardo DiCaprio}, v_1) \land \texttt{nominated}(v_1, v_2)\}$
Answers to soft constraints
Answers to hard constraints
Best Costume Design
Best Original Song
Film Editing
Best Sound
Best Cinematography
Best Picture
Best Directing
Best Actress
Best Supporting Actor
Best Actor
"About audio-visual design"
"Awarded to performers"
"About the artistic vision"

Figure 1: A query over a KG can be written in a logical form that specifies *hard constraints* that an entity must meet. We investigate the problem of incorporating *soft constraints* that cannot be specified in logical form, such as "*awards given to performers.*" Such constraints correspond to subsets of entities, shown in shaded clusters, that share some similarities.

The challenge of incorporating soft constraints lies in adjusting answer scores with minimal deviation from the original distribution of scores, so that hard logical constraints imposed by the query are still met. Training a model for this task requires datasets that capture the relationships between query answers and soft constraints – data that is not available in existing query answering benchmarks over incomplete KGs. Additionally, the reranking process must be efficient and integrate seamlessly with state-of-the-art QA models.

To tackle these challenges, we formalize the problem and introduce efficient methods designed to capture soft constraints while preserving the global structure of the original list of scores for a given query. In summary, our work makes the following key contributions:

1. We introduce the problem of query answering on knowledge graphs with soft entity constraints, defining it as an interactive process where preferences are specified incrementally.
2. We introduce two computationally efficient methods for reranking query answers based on an initial list of scores from an existing query answering system. The methods are lightweight, requiring tuning two parameters or a small neural network optimized to capture soft constraints while maintaining the original ranking structure.
3. We extend existing benchmarks for query answering with automatically generated soft constraints comprising a large and diverse set of queries with different notions of similarity. Through comprehensive evaluations, we show how our methods effectively capture these constraints without significantly compromising global ranking performance or runtime performance.

Our results point toward a more flexible interaction paradigm that complements existing methods for query answering on graphs, where answers can be adjusted according to preferences specified interactively.

2 Related Work

Approximate Query Answering.

We build on methods for learning on KGs that retrieve query answers that are not reachable via graph traversal. Early work focused on simple link prediction, where an edge is predicted between two entities (Nickel et al., 2011; Bordes et al., 2013; Trouillon et al., 2016; Lacroix et al., 2018; Sun et al., 2019). More recently, the set of queries that learning-based methods can tackle has been broadened to complex queries (e.g., paths of length two or three with intermediate variables), using a formulation of queries based on a small (Hamilton et al., 2018; Daza & Cochez, 2020; Ren et al., 2020; Ren & Leskovec, 2020; Arakelyan et al., 2021; 2023; Bai et al., 2023; Zhu et al., 2022; Ren et al., 2024) or a larger (Cucumides et al., 2024; Yin et al., 2024) subset of first order logic. In all these cases, the output of a query answering model is a score for each entity in the graph that indicates how likely it is to meet constraints specified using first order logic. Our work complements these approaches by supporting soft constraints that cannot be expressed in the same way, due to the inherent limitations of the language of first order logic.

Active Learning on KGs. The idea of collecting entities that exemplify soft constraints is closely related to the problem of active learning (Fu et al., 2013; Ren et al., 2022). Active learning allows training models incrementally when data is of limited availability. In the context of KGs, it has been applied to sampling training data for link prediction models under a constrained computational budget (Ostapuk et al., 2019), error detection (Dong et al., 2023), and KG alignment (Huang et al., 2023); all cases in which a prediction is well-defined. Applying active learning directly to an existing query answering model in order to capture soft constraints is not trivial, as a single query can have different types of soft constraints (as illustrated in Fig. 1), potentially leading to contradicting model updates. We avoid this by training a lightweight neural query reranker via supervised learning, using automatically generated data of soft preferences.

Information Retrieval (IR) and Recommender Systems. A common task in IR is to retrieve documents relevant to a given query (Manning et al., 2008), for example based on their similarity to a textual query using sequence embeddings (Devlin et al., 2019). We adopt a similar approach in one of the baselines in our experiments. We also draw inspiration from Learning to Rank (LTR), where models are trained to order items according to their relative relevance. Pairwise methods such as RankNet and LambdaRank (Burges et al., 2005; 2006) and their boosted extension LambdaMART (Burges, 2010) learn from preference pairs by directly optimizing ranking metrics such as NDCG, and have been shown to outperform regression- or classification-based ranking approaches. Modern implementations, such as the one provided in LightGBM (Ke et al., 2017), efficiently extend these methods to large datasets while supporting graded relevance labels and fast tree-based inference. These approaches generate a new ranking from scratch given a set of preferences; we, on the other hand, aim to preserve the original score relationships while biasing the ranking towards soft constraints. Other learning-to-rank methods take a list-wise approach that considers permutations of candidate lists (Cao et al., 2007; Xia et al., 2008; Zhu & Klabjan, 2020), but these approaches do not scale well to the number of entities typically found in a knowledge graph. Lastly, the field of Recommender Systems has explored the use of interaction data (e.g., clicks, ratings, and browsing history) to generate recommendations (Koren et al., 2021). Various collaborative filtering models exist in the literature (Ning & Karypis, 2011; Steck & Liang, 2021; Koren et al., 2009; Rendle et al., 2012; He et al., 2017; Christoffel et al., 2015), some of which demonstrate the effectiveness of neural networks at capturing preference data.

3 Preliminaries

We denote a KG as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{E}$ is the set of edges of the form (h, r, t) where $h, t \in \mathcal{V}$ are the head and tail entities, and $r \in \mathcal{R}$ is the relation between them. A triples (h, r, t) in a KG implies the truth of the statement $r(h, t)$ in first-order logic, where $r \in \mathcal{R}$ is a binary predicate, and $h, t \in \mathcal{V}$ its arguments.

Logical queries over KGs. A query q over $\mathcal{G}$ defines a set of constraints that an entity must satisfy to be considered an answer. For example, consider the query *What are award nominations received by movies starring Leonardo DiCaprio?* The *target* variable of this query is an award, and the query places two constraints on it: 1) the award nomination is given to movies, and 2) the movie features Leonardo DiCaprio. Formally, we can express the set A_q of answers to the query using first order logic as follows:

$$A_q = \{v_2 \mid \exists v_1, v_2 \in \mathcal{V} : \texttt{performer}(\texttt{Leonardo DiCaprio}, v_1) \wedge \texttt{nominated}(v_1, v_2)\}. \tag{1}$$

Answering this query requires assigning entities from $\mathcal{V}$ to variables v_1 and v_2 such that the conjunction specifying the constraints is true according to the information contained in the KG. More generally, and following the notation in Bai et al. (2023), we consider first-order logic (FOL) queries that can include conjunctions, disjunctions, and negations, which we write in disjunctive normal form as follows:

$$\mathcal{A}_q = \{v_t \mid \exists v_1, \ldots, v_N \in \mathcal{V} : (c_1^1 \wedge \cdots \wedge c_{m_1}^1) \vee \cdots \vee (c_1^n \wedge \cdots \wedge c_{m_n}^n)\}, \tag{2}$$

where $v_1, \ldots, v_N$ are variables in the query, and v_t is the *target* variable. The constraints c_j^i of the query consist of binary relations between two variables, or between a variable and a known entity in $\mathcal{V}$, while optionally including negations:

$$c_j^i = \begin{cases} r(e,v) \text{ or } r(v',v) \\ \neg r(e,v) \text{ or } \neg r(v',v) \end{cases} \qquad \text{with } e \in \mathcal{V}, v, v' \in \{v_t, v_1, \dots, v_N\}, \text{ and } r \in \mathcal{R}.$$

Approximate Query Answering. In order to provide answers to queries that are not limited to the information explicitly stated by the edges in the graph, recent works have introduced machine learning models for *approximate* query answering (QA) (Hamilton et al., 2018; Daza & Cochez, 2020; Ren et al., 2020; Ren & Leskovec, 2020; Arakelyan et al., 2021; 2023; Bai et al., 2023; Zhu et al., 2022; Ren et al., 2024; Cucumides et al., 2024; Yin et al., 2024). For a given query, these methods compute a score for all the entities in the KG according to how likely they are to be answers to a query. While the specific mechanisms for achieving this vary among methods, in general we denote a **QA model** as a function a that maps a query q to a vector $a(q) \in \mathbb{R}^{|\mathcal{V}|}$ of real-valued scores, with one score for each entity in the KG.

4 Query Answering with Soft Constraints

FOL queries over KGs allow imposing a broad range of constraints over entities, but this approach is still limited by the expressivity of first-order logic and the schema of the KG. In some applications one might require incorporating notions of similarity, which can be hard or impossible to express using FOL. We address this limitation by introducing the concept of *soft constraints*.

Definition 1 (Soft constraint (informal))**.** *A* ***soft constraint*** *is a loose requirement based on semantic similarity, which can guide answers to a query towards desirable types of answers without imposing strict logical criteria. For example, the soft constraint "*awards involving performers*" suggests prioritizing answers related to acting awards, without strictly excluding other awards.*

We consider a scenario where soft constraints are specified by means of specific examples of entities that represent the desired (or undesired) type of answer. We formalize this through labeled **preferences**. A preference is a tuple (e, l), where $e \in \mathcal{V}$ is an entity in the KG, and $l \in \{0, 1\}$ is a label indicating whether entities similar to e should be preferred ($l = 1$) or de-emphasized ($l = 0$).

Since specifying an exhaustive list of preferences can be impractical, we consider an *interactive* setting in which preferences are provided incrementally.

Definition 2 (Interactive query answering with soft constraints)**.** *Let q be a query with hard logical constraints evaluated over a knowledge graph, and let $a^{(0)}(q)$ denote the initial list of scores produced by a query answering model. In the* ***interactive query answering with soft constraints*** *setting, the scores are adjusted iteratively based on a growing set of user-provided preferences: after t interactions, the preference set is*

$$P(t) = \{(e_1, l_1), \dots, (e_t, l_t)\}. \tag{3}$$

At each iteration $t \geq 1$, we denote the adjusted list of scores as

$$a^{(t)}(q) = f\big(a^{(0)}(q), P(t)\big), \tag{4}$$

where f is a function that updates the initial scores according to the accumulated preferences.

This formulation enables incorporating soft constraints dynamically, guiding the ranking of answers toward entities semantically aligned with the provided examples. Consider the query "*What are award nominations received by movies starring Leonardo DiCaprio?*" The corresponding logical form retrieves all entities v_2 such that there exists a movie v_1 with $\texttt{performer}(\texttt{Leonardo DiCaprio}, v_1)$ and $\texttt{nominated}(v_1, v_2)$. Answers to the hard constraints include awards such as *Best Actor*, *Best Picture*, and *Best Sound*. In the case where we are interested in nominations *"related to the audio-visual design of the movie"*, but not those *"about the artistic vision"*, we can express this through the preference set

$$P(2) = \{(\texttt{Best Sound}, 1), (\texttt{Best Picture}, 0)\}. \tag{5}$$

Here, the label $l = 1$ marks `Best Sound` as a positive example (entities similar to it, such as `Best Original Song`, should receive higher scores) while $l = 0$ marks `Best Picture` as a negative example, encouraging the model to de-emphasize similar entities (e.g., `Best Director`). After incorporating $P(2)$, the updated scores $a^{(2)}(q)$ should give priority to entities that reflect the soft constraint.

5 Incorporating Soft Constraints

We now turn our attention to the problem of adapting answer scores based on preference sets that represent a soft constraint. The goal is to develop a method that can dynamically update the ranking of answers in response to incremental feedback of preferences, while maintaining the structure of the initial scores provided by the base QA model. Following our running example, given a soft preference like "*awards involving performers*", we would like awards such as "*Best Actress*" to rank higher than "*Best Sound*", but both entities should still rank higher than incorrect entities, such as "*Nobel Prize*".

To achieve this, we propose to modify the original score of each entity with a linear function with three inputs: (1) the original score computed by the base QA model, (2) its average similarity to *positively* labeled entities, and (3) its average similarity with *negatively* labeled entities. Formally, we start by defining a partition of a preference set into the following two sets:

$$P^+ = \{e \mid (e, l) \in P(t),\, l = 1\}, \tag{6}$$

$$P^- = \{e \mid (e, l) \in P(t),\, l = 0\}, \tag{7}$$

where we have dropped the dependency on t for brevity. Let $a[e]$ denote the score of entity e extracted from the vector of scores $a(q)$ computed by the base QA model. Our proposed score update is given by the following expression:

$$a^{\text{new}}[e] = f(a[e], \delta(P^+, e), \delta(P^-, e)), \tag{8}$$

where f is a linear function of its arguments and δ is a function that computes the mean cosine similarity of a preference set and an entity e:

$$\delta(P^\odot, e) = \frac{1}{|P^\odot|} \sum_{e_i \in P^\odot} \text{sim}(e_i, e). \tag{9}$$

The majority of existing QA models learn entity embeddings for computing query scores (Hamilton et al. (2018); Daza & Cochez (2020); Ren et al. (2020); Ren & Leskovec (2020); Arakelyan et al. (2021; 2023); Bai et al. (2023); Zhu et al. (2022), among others). We propose to reuse the embeddings of the base QA model (which are already required to compute the initial vector of scores $a(q)$) to compute the cosine similarity $\text{sim}(e_i, e)$ required in Equation (9).

Our proposed score update has two benefits: first, the use of a linear function **preserves monotonicity** in the scores, such that if two entities have equal similarities with respect to a preference set, then differences in their scores will be based exclusively on the initial scores computed by the QA model. Second, it is **efficient**, since it relies on already trained embeddings from the base QA model, and its application has linear time complexity, as we show next.

Time complexity. Let $n = |\mathcal{V}|$ be the number of entities, d the embedding dimension, and $t = |P(t)|$ the number of preferences. For each entity $e \in \mathcal{V}$ we compute $\delta(P^\odot, e)$ as the mean of t cosine similarities. Each cosine reduces to a dot product between d-dimensional vectors, so it costs $\mathcal{O}(d)$; hence one entity requires $\mathcal{O}(td)$ time to accumulate t dot products and take their mean, and processing all n entities costs $\mathcal{O}(ntd)$ time.

We now introduce two implementations of the update function f in Equation (8): a linear combination with fixed coefficients, and a neural reranker where the coefficients are computed conditionally.

5.1 Linear with fixed coefficients (Cosine).

Motivated by prior work on reranking methods for search over knowledge graphs (Gerritse et al., 2020; Daza et al., 2021), we first propose a score update that is computed as a convex combination of the original score, and a linear combination of the average cosine similarities with P^+ and P^-. We refer to this as the **Cosine** update:

$$f_{\text{Cos}}(a[e], \delta(P^+, e), \delta(P^-, e)) = \alpha \cdot a[e] + (1-\alpha)\left(\frac{1+\beta}{2}\delta(P^+, e) - \frac{1-\beta}{2}\delta(P^-, e)\right) \tag{10}$$

The cosine update thus requires tuning two parameters, $\alpha \in (0,1)$ and $\beta \in (-1,1)$ that control the weight given to the original score, and the average cosine similarities, which can be done using a validation set.

5.2 Neural query reranker (NQR).

We can further adapt the weights given to each term in the score update function in Equation (10) by computing them with a small multi-layer perceptron (MLP) with four inputs: the original score $a[e]$, the average similarities $\delta(P^+, e)$ and $\delta(P^-, e)$, and the embedding $\mathbf{e} \in \mathbb{R}^d$ of entity e (which we reuse from the base QA model and do not fine-tune).

Intuitively, making the weights conditional allows to contextualize the score update to specific values of scores, similarities, and the entity as captured by its embedding. The MLP outputs two scalars $\alpha_p, \alpha_n \in (0,1)$, which we then use in the update rule. We refer to this as the Neural Query Reranker (**NQR**) update:

$$f_{\text{NQR}}(a[e], \delta(P^+, e), \delta(P^-, e)) = a[e] + \alpha_p \delta(P^+, e) - \alpha_n \delta(P^-, e). \tag{11}$$

Time complexity. If the hidden dimension of the MLP is h, the forward pass per entity multiplies a $(d+3) \times h$ matrix and an $h \times 2$ matrix (plus activations), which is $\mathcal{O}(dh)$ time; across all entities this adds $\mathcal{O}(ndh)$. Thus the NQR update over all entities runs in $\mathcal{O}(n(td + dh))$ time, remaining linear in the number of entities in the KG.

Training NQR. The main goal of NQR is to adjust entity scores such that the information conveyed by the preference data is reflected in the ranking, while preserving the overall quality of the original query answers. This can be achieved with a training set of logical queries and their corresponding answers, together with a partition of the set of answers into disjoint preference sets P^+ and P^-. The training objective is then two-fold: (1) entities in P^+ should be ranked higher than those in P^-, and (2) all entities in $P^+ \cup P^-$ (which satisfy the original logical query) should still rank above entities that do not meet the logical constraints.

To achieve this, we adopt the RankNet loss (Burges et al., 2005), which encourages pairs of relevant and non-relevant items to be ordered correctly. Given two entities e_i and e_j with scores $a[e_i]$ and $a[e_j]$, RankNet minimizes the binary cross-entropy over the probability $\sigma(a[e_i] - a[e_j])$ that e_i should be ranked above e_j. For NQR, we define two complementary loss terms:

$$\mathcal{L}_{\text{NQR}} = \underbrace{\mathbb{E}_{(e^+, e^-) \sim (P^+, P^-)}\left[-\log \sigma\big(a[e^+] - a[e^-]\big)\right]}_{\text{(1) Preference-based ranking}} + \underbrace{\mathbb{E}_{(e^+, e_r) \sim (P^+ \cup P^-, \mathcal{N}_m)}\left[-\log \sigma\big(a[e^+] - a[e_r]\big)\right]}_{\text{(2) Query-consistency ranking}}, \tag{12}$$

where $\mathcal{N}_m$ is a set of m randomly sampled negative entities from $\mathcal{V}$ that are not valid answers to the query. The first term enforces the ordering implied by the preference set, while the second term ensures that preferred and non-preferred but valid answers remain ranked higher than irrelevant entities. We use gradient descent on this loss to update the parameters of the MLP in NQR.

6 Experiments

We aim to evaluate to what extent we can incorporate soft constraints as expressed in preference sets, while preserving overall query answering performance. We extend current benchmarks for query answering with preference data, and we run experiments comparing the performance of the Cosine and NQR updates, together with a baseline based on gradient-boosting approaches for learning to rank.

6.1 Datasets

Prior work on query answering on KGs has relied on datasets containing query-answer pairs $(q, \mathcal{A})$, where q is a query and $\mathcal{A} \subset \mathcal{V}$ is the set of answers computed over a KG (Ren et al., 2020; Ren & Leskovec,

Table 1: Statistics of the datasets used in our experiments.

	Knowledge Graph			Queries			Preference Sets		
Dataset	Entities	Relations	Edges	Train	Validation	Test	Train	Validation	Test
FB15k237	14,505	237	310,079	8,472	24,435	24,505	42,360	122,175	122,525
Hetionet	45,158	24	2,250,198	55,772	24,427	24,494	278,860	122,135	122,470

2020). We extend these datasets by deriving preference data from observable clusters of entities in the set $\mathcal{A}$. Motivated by prior work showing that textual embeddings correlate with human notions of similarity and relatedness (Reimers & Gurevych, 2019; Grand et al., 2022; Merkx et al., 2022), we achieve this by clustering embeddings of answers to a query, which we use as a computational proxy of similarity that allows us to study the problem at a scale of thousands of queries and comprising different notions of similarity. Concretely, this process consists of three steps (we provide additional details in Appendix B):

1. Embedding textual descriptions. In order to capture soft similarity relationships that go beyond the structure of the graph and are harder to encode via logical constraints, we retrieve textual descriptions of entities in the KG and compute embeddings for each of them using a text embedding model.

2. Clustering. We apply Hierarchical Agglomerative Clustering to the entities in $\mathcal{A}$, using cosine similarity as the distance metric. As shown in Fig. 2, the result is a dendrogram that iteratively merges the most similar entities into clusters, capturing similarities at different levels.

3. Answer set partitioning. We traverse the dendrogram from the root node towards the leaves, and if a cluster contains at least 20% of the entities in the answer set[1], we use it to form a preference set. For each entity $e_i \in \mathcal{A}$, we define its preference label as $l_i = 1$ if it is in the cluster, and $l_i = 0$ otherwise. An example of a preference set is shown in Fig. 2, with entities labeled as $l_i = 1$ highlighted in green, and the rest of the entities labeled as $l_i = 0$. This partition forms the preference set $P(T) = \{(e_1, l_1), \ldots, (e_T, l_T)\}$. We then denote an instance in our extended dataset as $(q, \mathcal{A}, P(T))$, with instances having values of T between 10 and 100.

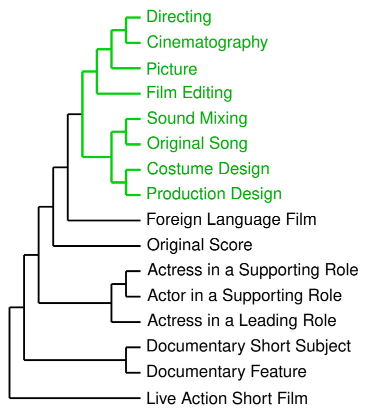

Figure 2: Example dendrogram used to generate preference sets. Highlighted in green are preferred entities, other entities are non-preferred.

Prior work has relied on subsets of encyclopedic KGs such as NELL (Carlson et al., 2010) and Freebase (Bollacker et al., 2008) to benchmark methods for query answering. These KGs contain statements about people, organizations, and locations around the world. For our experiments, we select FB15k237 (Toutanova & Chen, 2015) and the complex queries generated for it by Ren & Leskovec (2020). To further explore the potential of query answering in a different domain we use Hetionet (Himmelstein et al., 2017), a biomedical KG containing ∼2M edges between genes, diseases, and drugs, among other types. For Hetionet, we extract complex queries of different types, and then for both datasets we run the three steps above to collect preference data. We consider 14 types of first order logic queries of varying complexity, including negation. We present statistics of the final datasets and splits in Table 1. For training NQR, we only use 1-hop, link prediction queries. For the validation and test sets we use the full set of 14 types of complex queries. More detailed statistics can be found in Appendix C.

6.2 Evaluation

We evaluate performance on the task of interactive query answering with soft constraints as follows: given an instance $(q, \mathcal{A}, P(T))$ in the test set and a list of initial scores $a(q)$ computed by a base QA model, we select subsets of $P(t) \subseteq P(T)$ **incrementally**, with $1 \leq t \leq 10$. For each subset we compute the adjusted score (Eq. 8), and then we evaluate the following:

[1]We define this limit to avoid clusters with too few entities.

1. Pairwise accuracy: Measures the extent to which preferred entities are ranked higher than non-preferred entities. Let $\mathsf{rank}(e, a(q))$ be the position of entity e when sorting the list of scores $a(q)$ in descending order. We define the pairwise accuracy as follows:

$$\mathrm{PA} = \sum_{e^+ \in P^+} \sum_{e^- \in P^-} \mathbb{1}\left[\mathsf{rank}(e^+, a(q)) < \mathsf{rank}(e^-, a(q))\right], \tag{13}$$

where $\mathbb{1}[\cdot]$ is an indicator function. Importantly, while score updates are computed using a subset of $P(T)$ of maximum 10 preferences, PA is computed using the full set, which can contain up to 100 preferences. This allows us to evaluate whether methods can generalize from a limited number of examples.

2. Ranking metrics: Following prior work on CQA, we evaluate evaluate query answering performance using ranking metrics. For an answer $e \in \mathcal{A}$ to a query, the reciprocal rank is $1/\mathsf{rank}(e, a^{(t)})$, and the Mean Reciprocal Rank (MRR) is the average over all queries[2]. Hits at k (H@k) is defined by verifying if, for each answer e, its rank $\mathsf{rank}(e, a^{(t)})$ is less than or equal to k (in which case H@k = 1, and H@k = 0 otherwise), and averaging over all queries. We compute these ranking metrics using the original and adjusted scores to determine any effect on overall query answering performance.

3. NDCG: We further compute the Normalized Discounted Cumulative Gain at k (NDCG@k), which jointly captures the two criteria measured by PA and traditional ranking metrics: the relative ordering of preferred versus non-preferred answers, and their positions within the ranked list. We assign relevance scores of $0, 1, 2$ to non-answers, answers in P^-, and in P^+, respectively. The discounted cumulative gain and its normalized version are

$$\mathrm{DCG@}k = \sum_{e \in A_i(q)} \frac{2^{\mathrm{rel}(e)} - 1}{\log_2(j+1)}, \qquad \mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}, \tag{14}$$

where IDCG@k is the DCG of the ideal ranking. An NDCG@k of 1 indicates that all preferred answers appear above non-preferred and non-answers.

6.3 Baselines

We consider a traditional Learning-to-Rank (LTR) baseline based on LightGBM (Ke et al., 2017), a gradient boosting framework implementing the LambdaRank objective (Burges et al., 2005; 2006). For each training query, the model receives as input three scalar features for every entity e: the original score $a[e]$, and the average positive and negative similarities, $\delta(P^+, e)$ and $\delta(P^-, e)$. Graded relevance labels are assigned following standard LTR practice (Burges, 2010): entities in P^+ receive a label of 2, entities in P^- receive 1, and a random set of 100 negative entities is assigned 0. The model is trained to predict ranking scores that order entities according to their relative relevance by directly optimizing NDCG via the LambdaRank objective. This baseline allows us to compare our proposed linear score update based on shifts from the original score, with an approach that predicts the score using a nonlinear function of relevant features.

6.4 Experimental details

For all models, we find the best values of hyperparameters using the validation set via grid search. In all cases, we use QTO (Bai et al., 2023) as the base QA model. More details about training and hyperparameter tuning can be found in Appendix D. We implement all of our experiments using PyTorch (Paszke et al., 2019), and run them on a machine with one NVIDIA A100 GPU and 120GB of RAM.

7 Results

Soft constraints and ranking quality. We present results of pairwise accuracy, MRR, and H@10, and NDCG@10 in Fig. 3. For reference, we denote the metrics of the base QA model as **Unconstrained**, which allows us to determine how the adjustments in scores introduced by the methods we consider affect the performance of the original scores. We include 95% confidence intervals as shaded regions around the curves; these are narrow and often indistinguishable, indicating that the differences are consistent across queries.

[2]We compute *filtered* ranking metrics, where other known answers to a query are removed before computing $\mathsf{rank}(e, a^{(t)})$.

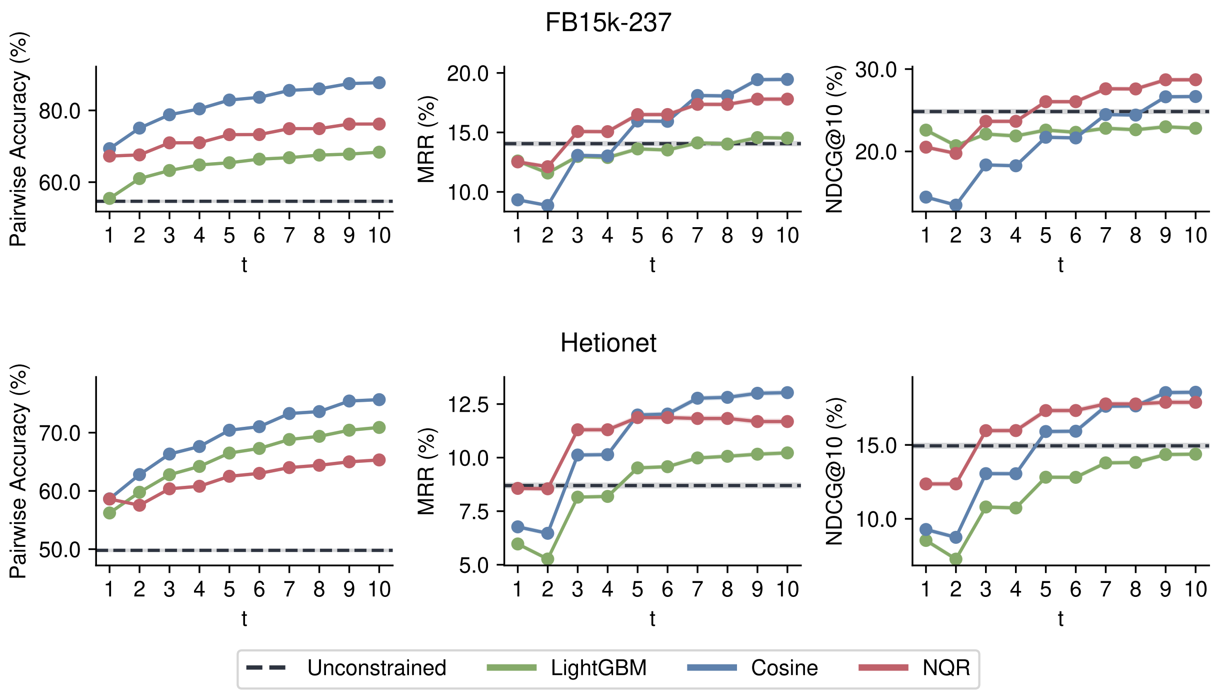

Figure 3: Results on interactive query answering with soft preferences, on FB15k237 and Hetionet.

Pairwise accuracy. Across both datasets, we observe that all methods manage to achieve a pairwise accuracy significantly higher than the starting value (when no preferences are taken into account), which increases consistently as larger preference sets are provided. This indicates that all methods correctly adjust the scores in a way that entities in P^+ are ranked higher than those in P^-. Notably, the Cosine method achieves the highest values, which reaches a 87.7% in FB15k237, and 75.7% in Hetionet.

Ranking metrics. Ranking metrics (MRR and H@10) behave differently in comparison with PA. When preference sets are small, between 1 and 4 in size, score adjustments often cause a degradation in ranking quality with respect to the original scores. The fact that PA is already high at such sizes of preference sets indicates that the adjusted scores greedily place entities in P^+ above P^-, possibly by placing wrong answers between them, which harms ranking quality. Considering all methods, and especially in Hetionet, NQR causes the smallest drops in performance at small preference sets. As preference sets grow larger, the performance of all methods (with the exception of LightGBM, which struggles on FB15k237 and in H@10 on Hetionet) increases above the original. Both Cosine and NQR yield the highest ranking metrics, indicating that linear score updates are better for preserving ranking quality in comparison with the LightGBM baseline that predicts the new score directly.

NDCG. The NDCG@10 results provide a unified view of the two underlying criteria: pairwise ordering and overall ranking quality. For small preference sets, all constrained models achieve higher NDCG@10 than the Unconstrained ranking, primarily due to better pairwise accuracy. Since both PA and ranking metrics improve as the size of the preference sets increases, NDCG@10 exhibits the same upward trend. Importantly, NDCG@10 allows comparing which method best satisfies the dual objective of query answering under soft constraints. We observe that for small preference sets, NQR performs best, suggesting that its learned conditional weights help preserve ranking quality while applying fine-grained score adjustments. For larger sets, the Cosine update surpasses NQR, matching or slightly improving ranking quality while achieving higher PA. This indicates that Cosine is a suitable general-purpose update rule, whereas NQR is preferable when fewer preferences are available.

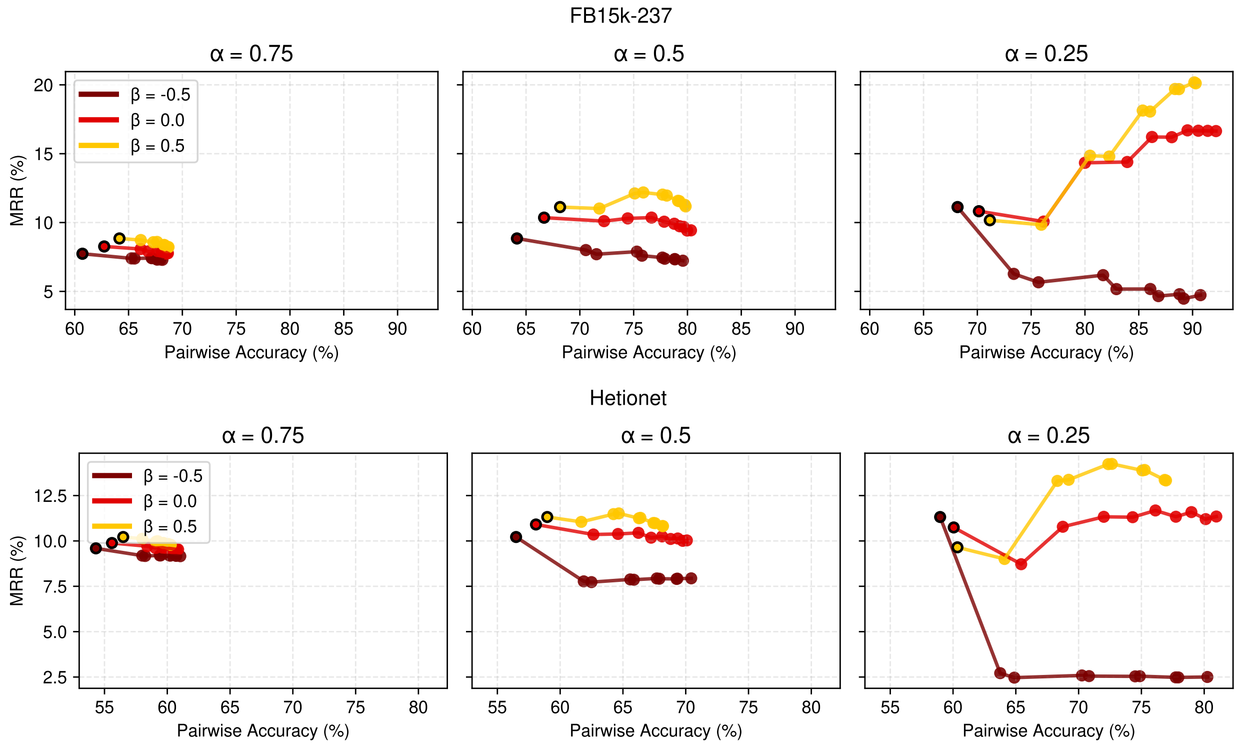

Figure 4: Trajectories of pairwise accuracy vs MRR for the Cosine update. Each subplot corresponds to a different value of α, with lines colored by values of β. Highlighted circles indicate the initial interaction step (t=1), and the following points show increasingly larger preference sets.

We present additional results in Appendix E where metrics are categorized according to each of the types of complex queries in the datasets. The results show that on average Cosine performs best at balancing similarity constraints with global ranking performance, followed by NQR, which for certain query types, preserves best global ranking performance.

Trade-offs induced by the Cosine update. Figure 4 illustrates how pairwise accuracy (PA) and mean reciprocal rank (MRR) evolve jointly for different values of α and β in the Cosine update (Equation (10)), as the number of interaction steps increases. Across both datasets, increasing α (which implies placing more weight on the original scores), prevents significant drops in MRR, but also limits the maximum PA that can be reached. Smaller α values, on the other hand, allow stronger preference propagation, and the effect on MRR varies depending on the value of β. This behavior reflects an important trade-off: higher α yields conservative and stable adjustments, while lower α enables more aggressive shifts guided by similarity signals.

Actionable control via β. Within each α setting, varying β determines the relative emphasis placed on positive versus negative preferences. Larger β values (yellow curves) consistently increase PA with limited loss in MRR, while negative β values (dark red) degrade both, indicating that over-penalizing negative examples distorts the global ranking. When $\alpha = 0.25$ and after 10 interactions, the values closer to the Pareto front are $\beta = 0$ and $\beta = 0.5$. Values of $\beta = 0.5$ increase PA further, at the cost of lower MRR, and viceversa for $\beta = 0$. This shows that β offers an interpretable control mechanism: users prioritizing ranking fidelity can select lower β values to protect MRR, whereas those seeking stronger preference alignment can increase β to boost PA. This tunability makes the Cosine update practically useful for downstream applications, enabling domain experts to adjust the balance between preserving original rankings and enforcing similarity constraints according to their needs.

Table 3: Example 2-hop query from FB15k-237, representing the question *"Which awards have had nominees who were born in New York City?"* and specific preference sets for it. We show the top-5 ranked target entities before and after applying similarity constraints using the Cosine update. ▲ indicates an entity promoted after adjusting the scores, and ▼ a demoted one.

$q(v_1, v_2) = \texttt{PlaceOfBirth}(\texttt{New York City}, v_1) \land \texttt{NominatedFor}(v_1, v_2)$	
P^+	P^-
Hugo Award for Best Novel	Grammy Award for Best Rock Instrumental Performance
World Fantasy Award for Best Novel	Grammy Award for Best Country Song
World Fantasy Award for Best Novella	Grammy Award for Best Country Performance
Initial Top-5	**Top-5 after Cosine update**
Golden Globe Award for Best Original Score	▲ Nebula Award for Best Novel
Academy Award for Best Original Song	▲ Hugo Award for Best Novella
Academy Award for Best Director	▲ Hugo Award for Best Short Story
Academy Award for Best Original Screenplay	▲ Hugo Award for Best Professional Artist
Nobel Prize in Literature	▲ Hugo Award for Best Novel
NDCG@10: 0.0	NDCG@10: 59.9

Qualitative example. Table 3 shows a 2-hop query from FB15k-237, where the task is to rank awards associated with people born in New York City. Applying similarity constraints on the target variable using the Cosine update sharply improves ranking quality: literary awards in the positive set are promoted into the Top-5, while unrelated film and TV awards drop in rank. NDCG@10 increases from 0.0 to 0.60, illustrating how a small preference set can adjust the ranking towards the soft constraint that awards returned as an answer to the query should be literary rather than film or music awards.

Runtime. We present the average runtime in milliseconds per query in the FB215k237 dataset for all methods in Table 2, where we include the added runtime (Δ) with respect to the Unconstrained baseline. We note that LightGBM results in very high overhead, by adding 13.8 ms per query, due to the use of sequential tree traversal. On the other hand, the Cosine and NQR methods add very little overhead of up to 1.4 ms per query. While the average runtime of Cosine seems higher than NQR, a paired t-test does not show that the difference is statistically significant.

Table 2: Average execution time per query for different reranking methods on FB15k237.

Method	Runtime (ms)	Δ (ms)
Unconstrained	0.6	—
LightGBM	14.4	+13.8
Cosine	2.0	+1.4
NQR	1.8	+1.2

8 Conclusion

We have introduced the problem of interactive knowledge graph query answering with soft entity constraints. We formulate it as an extension that addresses the limitations of logical queries over knowledge graphs, which are restricted by the expressivity of first order logic and the schema of the graph.

We introduce efficient methods for incorporating soft constraints in complex queries. Since they reuse embeddings of a base CQA model, the methods are lightweight, requiring tuning only two parameters or a small MLP. To evaluate these models over a large and diverse set of queries and notions of similarity, we extend existing query answering benchmarks with preference sets derived from clusterings of embeddings. In our experiments, we find that these methods are able to adjust query scores to capture preferences about their answers, while increasing ranking performance as more exemplary preferences are provided. In practice, we observe that our methods are fast, adding only up to 1.4 ms overhead to the base QA model.

With soft constraints, we explore a new and flexible way to interact with graph databases, which can allow users to specify their preferences by providing examples interactively, leading to query answering systems that can quickly adapt to user feedback.

References

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Mos9F9kDwkz.

Erik Arakelyan, Pasquale Minervini, Daniel Daza, Michael Cochez, and Isabelle Augenstein. Adapting neural link predictors for data-efficient complex query answering. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. Answering complex logical queries on knowledge graphs via query computation tree optimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1472–1491. PMLR, 2023. URL https://proceedings.mlr.press/v202/bai23b.html.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In Jason Tsong-Li Wang (ed.), *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pp. 1247–1250. ACM, 2008. doi: 10.1145/1376616.1376746.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In Luc De Raedt and Stefan Wrobel (eds.), *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pp. 89–96. ACM, 2005. doi: 10.1145/1102351.1102363.

Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to Rank with Nonsmooth Cost Functions. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann (eds.), *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pp. 193–200. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html.

Christopher J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In Zoubin Ghahramani (ed.), *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pp. 129–136. ACM, 2007. doi: 10.1145/1273496.1273513.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr, and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In Maria Fox and David Poole (eds.), *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pp. 1306–1313. AAAI Press, 2010. doi: 10.1609/AAAI.V24I1.7519. URL https://doi.org/10.1609/aaai.v24i1.7519.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation Prediction as an Auxiliary Training Objective for Improving Multi-Relational Graph Representations. In Danqi Chen, Jonathan Berant, Andrew McCallum, and Sameer Singh (eds.), *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*, 2021. doi: 10.24432/C54K5W.

Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 163–170, 2015.

Tamara Cucumides, Daniel Daza, Pablo Barceló, Michael Cochez, Floris Geerts, Juan L. Reutter, and Miguel Romero. Unravl: A neuro-symbolic framework for answering graph pattern queries in knowledge graphs. In Guy Wolf and Smita Krishnaswamy (eds.), *Learning on Graphs Conference, 26-29 November 2024, Virtual*, volume 269 of *Proceedings of Machine Learning Research*, pp. 2. PMLR, 2024. URL https://proceedings.mlr.press/v269/cucumides25a.html.

Daniel Daza and Michael Cochez. Message passing query embedding. In *ICML Workshop - Graph Representation Learning and Beyond*, 2020. URL https://arxiv.org/abs/2002.02406.

Daniel Daza, Michael Cochez, and Paul Groth. Inductive entity representations from text via link prediction. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (eds.), *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pp. 798–808. ACM / IW3C2, 2021. doi: 10.1145/3442381.3450141. URL https://doi.org/10.1145/3442381.3450141.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Junnan Dong, Qinggang Zhang, Xiao Huang, Qiaoyu Tan, Daochen Zha, and Zhao Zihao. Active Ensemble Learning for Knowledge Graph Error Detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, pp. 877–885, New York, NY, USA, February 2023. Association for Computing Machinery. ISBN 978-1-4503-9407-9. doi: 10.1145/3539597.3570368. URL https://dl.acm.org/doi/10.1145/3539597.3570368.

Dieter Fensel, Umutcan Simsek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. *Knowledge Graphs - Methodology, Tools and Selected Use Cases*. Springer, 2020. ISBN 978-3-030-37438-9. doi: 10.1007/978-3-030-37439-6.

Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowl. Inf. Syst.*, 35 (2):249–283, 2013. doi: 10.1007/S10115-012-0507-8. URL https://doi.org/10.1007/s10115-012-0507-8.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon (eds.), *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pp. 413–422. International World Wide Web Conferences Steering Committee / ACM, 2013. doi: 10.1145/2488388.2488425.

Emma J. Gerritse, Faegheh Hasibi, and Arjen P. de Vries. Graph-embedding empowered entity retrieval. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins (eds.), *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I*, volume 12035 of *Lecture Notes in Computer Science*, pp. 97–110. Springer, 2020. doi: 10.1007/978-3-030-45439-5_7. URL https://doi.org/10.1007/978-3-030-45439-5_7.

Gabriel Grand, Idan A. Blank, Francisco Pereira, and Evelina Fedorenko. Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nature Human Behaviour*, 6(7): 975–987, Jul 2022. doi: 10.1038/s41562-022-01316-8.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 2030–2041, Red Hook, NY, USA, 2018. Curran Associates Inc.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726, September 2017. ISSN 2050-084X. doi: 10.7554/eLife.26726. URL https://doi.org/10.7554/eLife.26726. Publisher: eLife Sciences Publications, Ltd.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Number 22 in Synthesis Lectures on Data, Semantics, and Knowledge. Springer, 2021. ISBN 978-3-031-00790-3. doi: 10.2200/S01125ED1V01Y202109DSK022. URL https://kgbook.org/.

Jiacheng Huang, Zequn Sun, Qijin Chen, Xiaozhou Xu, Weijun Ren, and Wei Hu. Deep Active Alignment of Knowledge Graph Entities and Schemata. *Proc. ACM Manag. Data*, 1(2):159:1–159:26, June 2023. doi: 10.1145/3589304. URL https://dl.acm.org/doi/10.1145/3589304.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2):494–514, 2022. doi: 10.1109/TNNLS.2021.3070843. URL https://doi.org/10.1109/TNNLS.2021.3070843.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3146–3154, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Yehuda Koren, Steffen Rendle, and Robert Bell. Advances in collaborative filtering. *Recommender systems handbook*, pp. 91–142, 2021.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2869–2878. PMLR, 2018. URL http://proceedings.mlr.press/v80/lacroix18a.html.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion. In Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl (eds.), *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pp. 3–20. Springer, 2018. doi: 10.1007/978-3-030-00671-6_1. URL https://doi.org/10.1007/978-3-030-00671-6_1.

Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In Sarit Kraus (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 3137–3143. ijcai.org, 2019. doi: 10.24963/IJCAI.2019/435. URL https://doi.org/10.24963/ijcai.2019/435.

Danny Merkx, Stefan Frank, and Mirjam Ernestus. Seeing the advantage: visually grounding word embeddings to better capture human semantic knowledge. In Emmanuele Chersoni, Nora Hollenstein, Cassandra Jacobs, Yohei Oseki, Laurent Prévot, and Enrico Santus (eds.), *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pp. 1–11, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.cmcl-1.1. URL https://aclanthology.org/2022.cmcl-1.1/.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 809–816, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016. doi: 10.1109/JPROC.2015.2483592. URL https://doi.org/10.1109/JPROC.2015.2483592.

Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th international conference on data mining*, pp. 497–506. IEEE, 2011.

Natalia Ostapuk, Jie Yang, and Philippe Cudre-Mauroux. ActiveLink: Deep Active Learning for Link Prediction in Knowledge Graphs. In *The World Wide Web Conference*, WWW '19, pp. 1398–1408, New York, NY, USA, May 2019. Association for Computing Machinery. ISBN 978-1-4503-6674-8. doi: 10.1145/3308558.3313620. URL https://dl.acm.org/doi/10.1145/3308558.3313620.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. *arXiv*, October 2020. URL http://arxiv.org/abs/2010.04029. arXiv: 2010.04029 Publisher: arXiv.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1410. URL https://doi.org/10.18653/v1/D19-1410.

Hongyu Ren and Jure Leskovec. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e43739bba7cdb577e9e3e4e42447f5a5-Abstract.html.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJgr4kSFDS.

Hongyu Ren, Mikhail Galkin, Zhaocheng Zhu, Jure Leskovec, and Michael Cochez. Neural graph reasoning: A survey on complex logical query answering. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=xG8un9ZbqT.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A Survey of Deep Active Learning. *ACM Comput. Surv.*, 54(9):180:1–180:40, 2022. doi: 10.1145/3472291.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

Harald Steck and Dawen Liang. Negative interactions for improved collaborative filtering: Don't go deeper, go higher. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pp. 34–43, 2021.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HkgEQnRqYQ.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In Alexandre Allauzen, Edward Grefenstette, Karl Moritz Hermann, Hugo Larochelle, and Scott Wen-tau Yih (eds.), *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, CVSC 2015, Beijing, China, July 26-31, 2015*, pp. 57–66. Association for Computational Linguistics, 2015. doi: 10.18653/V1/W15-4007. URL https://doi.org/10.18653/v1/W15-4007.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2071–2080. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/trouillon16.html.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In William W. Cohen, Andrew McCallum, and Sam T. Roweis (eds.), *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pp. 1192–1199. ACM, 2008. doi: 10.1145/1390156.1390306.

Hang Yin, Zihao Wang, and Yangqiu Song. Rethinking Complex Queries on Knowledge Graphs with Neural Link Predictors. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=1BmveEMNbG.

Xiaofeng Zhu and Diego Klabjan. Listwise Learning to Rank by Exploring Unique Ratings. In James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (eds.), *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, pp. 798–806. ACM, 2020. doi: 10.1145/3336191.3371814.

Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-Symbolic Models for Logical Queries on Knowledge Graphs. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27454–27478. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhu22c.html.

A Appendix

B Generating preference data

In our experiments, we obtain preference data by clustering answers to queries over a KG using embedding models that map the description of an entity to an embedding of fixed dimension. In this section we provide additional technical details of this process.

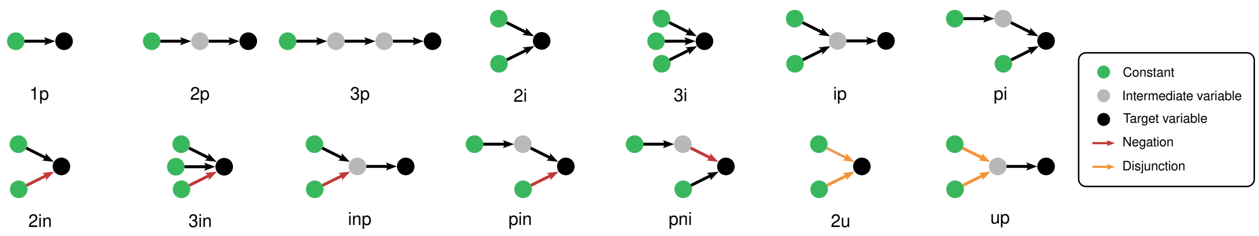

Figure 5: Types of complex queries used in our experiments.

Table 4: Statistics of the datasets for query answering with soft preferences used in our experiments.

Split	Structure	1p	2p	3p	2i	3i	ip	pi	2in	3in	inp	pin	pni	2u	up	Total
								FB15k-237								
Train	Queries	8,472														8,472
	Preferences	42,360														42,360
Valid	Queries	281	2,120	2,660	779	410	2,054	1,211	1,793	1,439	2,230	2,523	2,074	2,095	2,766	24,435
	Preferences	1,405	10,600	13,300	3,895	2,050	10,270	6,055	8,965	7,195	11,150	12,615	10,370	10,475	13,830	122,175
Test	Queries	327	2,101	2,717	843	500	1,942	1,184	1,805	1,569	2,257	2,460	2,031	2,052	2,717	24,505
	Preferences	1,635	10,505	13,585	4,215	2,500	9,710	5,920	9,025	7,845	11,285	12,300	10,155	10,260	13,585	122,525
								Hetionet								
Train	Queries	55,772														55,772
	Preferences	278,860														278,860
Valid	Queries	1,469	2,319	2,573	802	491	2,116	1,365	1,564	956	2,475	2,557	1,142	1,936	2,662	24,427
	Preferences	7,345	11,595	12,865	4,010	2,455	10,580	6,825	7,820	4,780	12,375	12,785	5,710	9,680	13,310	122,135
Test	Queries	1,472	2,309	2,562	850	556	2,087	1,376	1,556	1,004	2,473	2,535	1,137	1,938	2,639	24,494
	Preferences	7,360	11,545	12,810	4,250	2,780	10,435	6,880	7,780	5,020	12,365	12,675	5,685	9,690	13,195	122,470

Embedding textual descriptions. We rely on text embedding models publicly available on Hugging Face[3] for computing embeddings of entities based on their textual descriptions. For entities in FB15k237, we use `NovaSearch/stella_en_400M_v5`[4], with an embedding dimension of 1,024. For Hetionet we employ `Alibaba-NLP/gte-Qwen2-7B-instruct`[5], which at the time of writing is ranking 1st in the Massive Text Embedding Benchmark[6] in the medical domain. The embedding dimension of this model is 3,584.

Clustering. We execute hierarchical clustering using the average linkage method and cosine distance as the dissimilarity metric, applied to the target embeddings. This is implemented using the `linkage` function from the SciPy package. Average linkage is selected because it considers the mean pairwise distance between points in different clusters, which tends to produce more balanced and interpretable cluster structures. Cosine distance is used as it effectively captures angular similarity, which is more meaningful than Euclidean distance in high-dimensional embedding spaces, where the direction of vectors often encodes more relevant information than their magnitude.

C Dataset statistics

We consider 14 types of complex queries over KGs introduced in prior work (Ren & Leskovec, 2020), which include conjunctions, disjunctions, and negation. These queries can be represented using query graphs, which we illustrate in Fig. 5.

In our experiments we consider complex queries over the FB15k237 and Hetionet KGs, selecting those containing at least 10 answers and at most 100. For each query, we execute the algorithm for generating preference data (Appendix B) and generate 5 preference sets per query. The final number of queries and preference sets for each KG is shown in Table 4.

[3]`https://huggingface.co/`
[4]`https://huggingface.co/NovaSearch/stella_en_400M_v5`
[5]`https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct`
[6]`https://huggingface.co/spaces/mteb/leaderboard`

Table 5: Layer-wise description of the model architecture in NQR. t indicates the number of input preferences, which is variable. The embedding dimension is 1,000, and in specific layers we concatenate the score of an entity, indicated as +1.

	Input Dimension	**Output Dimension**
Preference embedding		
Self-attention	$(t, 1000 + 1)$	$(t, 1001)$
LayerNorm	$(t, 1001)$	$(t, 1001)$
Linear + ReLU (fc1)	$(t, 1001)$	$(t, 1000)$
Mean-pooling	$(t, 1000)$	1000
Score adjustment		
Linear + ReLU	$(2000 + 1)$	1000
Linear + tanh	1000	1

D Experimental details

Base query answering model. We make use of QTO as the base query answering model Bai et al. (2023). QTO relies on a neural link predictor, for which we use ComplEx (Trouillon et al., 2016; Lacroix et al., 2018) trained with an auxiliary relation prediction objective (Chen et al., 2021). We use an embedding dimension of 1,000, learning rate of 0.1, batch size 1,000, and a regularization weight of 0.05.

Cosine similarity. We search for the best values of α_p, α_n via a grid search where each can take values in $\{0.1, 0.25, 0.5, 0.75, 0.9\}$, based on performance on the validation set. We define performance as the sum of pairwise accuracy and ranking performance.

NQR. We describe the architecture of NQR in Table 5. When using the RankNet loss we select the best learning rate in $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}\}$. For NQR, we perform a grid search where the learning rate takes values in $\{1 \times 10^{-5}, 1 \times 10^{-4}\}$, the margin in the preference loss in $\{0.05, 0.1, 0.25\}$, and the KL divergence weight in $\{0.1, 1.0, 10\}$.

E Additional results

In our main experiments, we report the performance of different methods when presented with incremental feedback with preference sets of size 1 up to 10, averaged over all queries. This requires queries that have at preference set (combining both preferred and non-preferred entities) of at least 10 entities, such that we can evaluate methods over the entire range of preference set sizes.

We present here results over all types of complex queries we consider in our experiments. As in our main experiments, we compute the metrics for preference sets of increasing sizes, from 1 up to 10. Here, we report the metrics averaged over all 10 steps. We present results categorized by query structure (illustrated in Fig. 5) in Tables 6 and 7.

We observe that in average, Cosine and NQR result in the best results, with Cosine achieving a good balance between pairwise accuracy and global ranking performance, as shown by the results in NDCG@10. The results by query type also indicate cases where all models struggle to maintain the ranking performance of the Unconstrained model: in the FB15k237 dataset and for ip and pi queries, all models result in lower MRR, with NQR being the closest. Similarly, in Hetionet all models underperform the Unconstrained baseline in 2u queries, though NQR achieves a close value.

Table 6: Per query type results of interactive query answering with feedback on the FB15k237 dataset.

Method	**1p**	**2p**	**3p**	**2i**	**3i**	**ip**	**pi**	**2in**	**3in**	**inp**	**pin**	**pni**	**2u**	**up**	**avg**
Average Pairwise Accuracy															
Unconstrained	58.44	55.67	55.30	55.55	53.45	56.19	54.97	58.06	54.99	53.41	54.40	58.70	59.08	55.74	56.00
LightGBM	80.10	66.07	59.85	71.98	66.97	61.55	61.70	74.45	69.43	61.25	62.89	73.14	75.09	67.58	68.00
Cosine	**85.66**	**85.06**	**85.80**	**81.42**	**77.23**	**84.68**	**82.70**	**82.99**	**81.64**	**84.61**	**84.45**	**83.60**	**83.54**	**83.99**	**83.38**
NQR	76.80	76.11	77.45	71.75	68.76	76.56	74.09	73.78	71.99	74.66	75.29	74.34	73.71	74.27	74.26
Average MRR															
Unconstrained	19.97	16.32	15.73	24.02	27.73	**21.90**	**22.27**	11.32	15.39	11.55	9.53	3.88	16.08	16.44	16.58
LightGBM	16.93	15.34	14.69	19.82	24.25	18.90	19.31	11.81	15.18	12.03	10.85	**8.96**	13.28	14.96	15.45
Cosine	24.09	16.65	14.52	23.25	27.78	16.77	17.51	**15.08**	16.31	14.47	13.02	7.89	17.88	16.12	17.24
NQR	**25.37**	**19.08**	**16.28**	**26.22**	**29.17**	19.15	20.34	14.60	**17.79**	**15.98**	**13.11**	5.63	**18.37**	**18.84**	**18.57**
Average NDCG@10															
Unconstrained	39.92	36.79	36.94	51.40	54.67	44.44	48.28	33.56	43.10	29.86	28.28	10.86	43.37	39.88	38.67
LightGBM	45.62	43.73	42.06	53.42	56.09	48.13	51.49	43.07	49.81	39.32	38.45	**36.44**	45.42	46.72	45.70
Cosine	**58.64**	55.67	53.23	**62.01**	**63.42**	56.37	56.52	**54.77**	**56.52**	**53.36**	**51.76**	33.15	**60.11**	56.84	**55.17**
NQR	57.07	**56.39**	**54.71**	61.95	62.79	**57.66**	**58.33**	49.37	55.30	51.77	48.81	20.87	56.70	**57.00**	53.48

Table 7: Per query type results of interactive query answering with feedback on the Hetionet dataset.

Method	**1p**	**2p**	**3p**	**2i**	**3i**	**ip**	**pi**	**2in**	**3in**	**inp**	**pin**	**pni**	**2u**	**up**	**avg**
Average Pairwise Accuracy															
Unconstrained	51.75	50.73	50.55	51.78	51.19	51.00	51.06	48.03	41.51	49.99	48.84	51.73	52.62	50.58	50.10
LightGBM	**79.18**	65.73	63.76	**75.71**	**75.20**	64.63	69.48	71.17	68.62	63.37	64.24	74.14	**74.24**	67.16	69.76
Cosine	75.27	**73.30**	**71.65**	72.13	71.48	**73.19**	**73.57**	**74.19**	**74.58**	**73.29**	**71.12**	**75.06**	70.70	**69.21**	**72.77**
NQR	68.17	63.19	60.78	65.96	65.30	65.40	67.12	66.84	69.39	61.60	60.67	67.28	63.44	59.59	64.62
Average MRR															
Unconstrained	25.91	4.83	7.86	21.74	26.20	7.98	15.00	13.27	8.53	4.25	3.80	3.27	**18.89**	5.69	11.95
LightGBM	13.37	8.47	**11.31**	11.42	14.28	10.26	11.47	9.26	7.87	8.19	7.23	**9.87**	9.19	7.32	9.97
Cosine	22.07	**9.26**	9.66	20.40	24.45	10.61	13.94	13.52	12.75	**8.77**	**8.30**	9.24	15.06	**8.23**	13.30
NQR	**26.39**	8.87	9.32	**22.89**	**27.23**	**10.77**	**15.51**	**15.11**	**13.20**	7.82	7.66	7.23	18.72	8.22	**14.21**
Average NDCG@10															
Unconstrained	44.20	19.83	26.34	43.06	45.51	24.72	38.31	33.33	22.95	19.29	17.58	9.62	48.75	23.92	29.81
LightGBM	41.45	41.17	**49.29**	39.12	41.06	43.87	40.70	35.96	29.84	43.06	39.60	**38.24**	40.35	43.64	40.52
Cosine	**53.28**	**45.09**	49.16	**53.79**	**56.23**	**46.80**	47.95	**46.59**	**44.64**	**46.79**	**44.57**	35.94	52.21	**47.01**	**47.86**
NQR	53.04	42.28	41.44	53.16	55.79	45.77	**48.91**	45.97	43.34	40.97	40.23	27.65	**54.24**	43.83	45.47