Knowledge Base Question Answering by Transformer-Based Graph Pattern Scoring

Marcel Lamott[1,*], Jörn Hees[2,3] and Adrian Ulges[1]

[1]RheinMain University of Applied Sciences, Wiesbaden, Germany
[2]Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany
[3]Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany

Abstract

Question Answering (QA) has gained significant attention in recent years, with transformer-based models improving natural language processing. However, issues of explainability remain, as it is difficult to determine whether an answer is based on a true fact or a hallucination. Knowledge-based question answering (KBQA) methods can address this problem by retrieving answers from a knowledge graph. This paper proposes a hybrid approach to KBQA called FRED, which combines pattern-based entity retrieval with a transformer-based question encoder. The method uses an evolutionary approach to learn SPARQL patterns, which retrieve candidate entities from a knowledge base. The transformer-based regressor is then trained to estimate each pattern's expected F1 score for answering the question, resulting in a ranking of candidate entities. Unlike other approaches, FRED can attribute results to learned SPARQL patterns, making them more interpretable. The method is evaluated on two datasets and yields MAP scores of up to 73 percent, with the transformer-based interpretation falling only 4 pp short of an oracle run. Additionally, the learned patterns successfully complement manually generated ones and generalize well to novel questions.

Keywords
knowledge graphs, question answering, transfer learning

1. Introduction

The field of question answering (QA) has received significant attention, particularly with the rise of Deep Learning approaches that often focus on large language models. However, while these models have shown promising results, they suffer from explainability issues and often focus on knowledge from large text corpora instead of previously extracted and curated knowledge. Knowledge graphs offer an alternative approach for ML models to conduct transparent reasoning, leading to the emergence of the field of knowledge base question answering (KBQA), where the system retrieves the answer to a question not from a text corpus but from a knowledge graph.

In this paper, we present a hybrid approach towards KBQA. We leverage graph patterns in combination with neural (transformer-based) question interpretation to address the limitations of large text models and the lack of use of knowledge bases. Given a training set of questions, each associated with a source entity and answer entities, we use an evolutionary graph pattern

ICCBR TMG'23: Workshop on Text Mining and Generation at ICCBR2023, July 17 – 20, 2023, Aberdeen, Scotland

*Corresponding author.

marcellamott+at+web.de (M. Lamott); joern.hees+at+h-brs.de (J. Hees); adrian.ulges+at+hs-rm.de (A. Ulges)

learning algorithm to define a set of graph patterns covering the space of questions represented in the training set.

Given a new question, we match it to the graph patterns by fusing a neural question representation with the graph pattern set. To do so, a transformer-based regressor is trained to predict graph patterns' F1 scores when retrieving the input question's answer, and the resulting predictions are used to fuse the graph pattern's result candidate sets. Our main contributions in this paper are:

1. A novel and explainable hybrid KBQA system, combining two main components: a graph pattern learner (GPL) and a transformer-based scoring approach. The GPL component learns graph patterns to query a SPARQL endpoint. The patterns are learned with an evolutionary algorithm, but can also be complemented with manually created patterns by experts. Additionally, we propose a novel scoring component fusing the pattern-based generated candidates based on a transformer architecture to understand the natural language input question.
2. We conduct extensive experiments on versions of the well-known WebQuestionsSP [1] and SimpleQuestions [2] datasets. Our experimental results show that our approach achieves MAP scores of up to 73 %, with the transformer-based interpretation falling only 4 percentage points short of an oracle run.
3. Furthermore, we compare learned and hand-crafted expert patterns, as well as their combination, and show that the learned patterns successfully complement manually generated patterns and generalize well to novel questions.
4. To support future research in the area of KBQA, we will make our datasets and models publicly available.

In the following, we provide an overview of related work in Section 2 before providing more details on our proposed hybrid approach towards KBQA in Section 3. We describe the datasets used in the experiments in Section 4 and present our experimental results in Section 5. Finally, we discuss our results in Section 6.

2. Related Work

Information retrieval (IR) based QA relies on corpora of documents to retrieve answers from. An example are search engines, many of which provide interfaces to IR based QA by indexing large amounts of web pages, ranking them based on the search query, and retrieving answer spans from passages. Recently, a spike in research and public interest in large language models (LLMs) has not only provided an addition to search engines, but to the field of QA in general. Commercial LLMs, such as ChatGPT [3] and Bard [4], or open source alternatives such as LLaMA [5], Alpaca [6] and Dolly [7] have received much attention due to their ability to produce humanlike responses to open-domain inputs. They rely on large text corpora and sophisticated training mechanisms incorporating methods from Deep Learning, Reinforcement Learning and expert annotations. Their fundamental mechanism is to predict the next token which is most likely to follow the previous tokens of an output and also incorporate the context from a conversation, i.e., the dialog preceding the current prompt. While these models have demonstrated their power in text generation, their usage for QA remains disputed: Not only are LLMs intransparent and not easily explainable with regard to how the model arrived at a certain conclusion, they are also known to suffer from *hallucinations*, i.e., cases when an LLM outputs made-up but seemingly plausible facts not grounded on any training data. Despite recent efforts in *faithful question answering* [8], this poses a dangerous drawback for QA applications.

An alternative to this approach is presented by KBQA, which is not only fact grounded, but also more transparent and explainable, since every answer can be traced back directly to the underlying KB. KBQA can be divided into two categories: First, methods that directly yield a ranking of target entities, typically by relying on Deep Learning and/or embeddings of the knowledge graph. Second (and more similar to our approach), methods that explicitly generate queries to retrieve information from the underlying KB. Representatives from the latter category provide advantages with regard to transparency and explainability, as they not only provide an answer, but also the exact query which retrieved it from the KB. Two examples for each category are given in the following:

Regarding approaches from the first category, one method that employs Deep Learning on the knowledge graph is KEQA [9], which leverages KG embeddings and a manually designed distance metric to perform KBQA. To this end, the underlying KG is embedded using a pre-trained KG embedding algorithm (e.g., TransE [10] or TransR [11]). A predicate learning model is trained, which maps a question to a vector in the relation embedding space as the predicted relation presentation. Similarly, a head entity learning model is trained, which maps a question to a vector in the entity embedding space as the predicted head entity representation. Furthermore, a head entity detection model is trained to identify several tokens in the question as the predicted head entity name, which is used to reduce the number of head entity candidates in the KG. The output from the three models is used in a manually designed distance metric, which returns a fact in the KG that minimizes the distance the predicted head entity, predicted relation and the predicted tail entity, where the latter is predicted using the KG embedding algorithm's relation function. KEQA focuses on simple questions referring to a single head entity and a single relation and uses the SimpleQuestions [2] dataset.

ReaRev [12] is a recent work achieving start-of-the-art (or nearly) performance on the three common KBQA datasets WebQuestions [13], ComplexWebQuestions [14] and MetaQA [15]. Question- and KG-representations are aligned in a common latent space for reasoning over the KG. To this end, the question is decoded into an initial set of instructions, which are dense representations that are matched against relations in the KG. ReaRev then uses two alternating steps: The *reasoning* step uses the instructions to perform a KG traversal and results in a set of node representations, and the *revise* step uses the reasoning output to refine the instructions. During the reasoning step, the execution order of the instructions is decided by the model on the fly by emulating breadth-first search with graph neural networks. Both steps are used alternately for T times to iteratively refine the resulting node representations. Afterwards, a binary, GNN-based node classification is performed to select nodes which represent an answer to the question. Our approach differs from KEQA and ReaRev in that both either rely on Deep Learning / KG embeddings rather than explicit graph patterns.

Regarding the second category of KBQA systems, one approach based on query construction is SPBERT [16], which uses an adapted BERT [17] model (named BERT2SPBERT) to construct a SPARQL query for a given question. To do so, changes in BERT's self-attention layers are

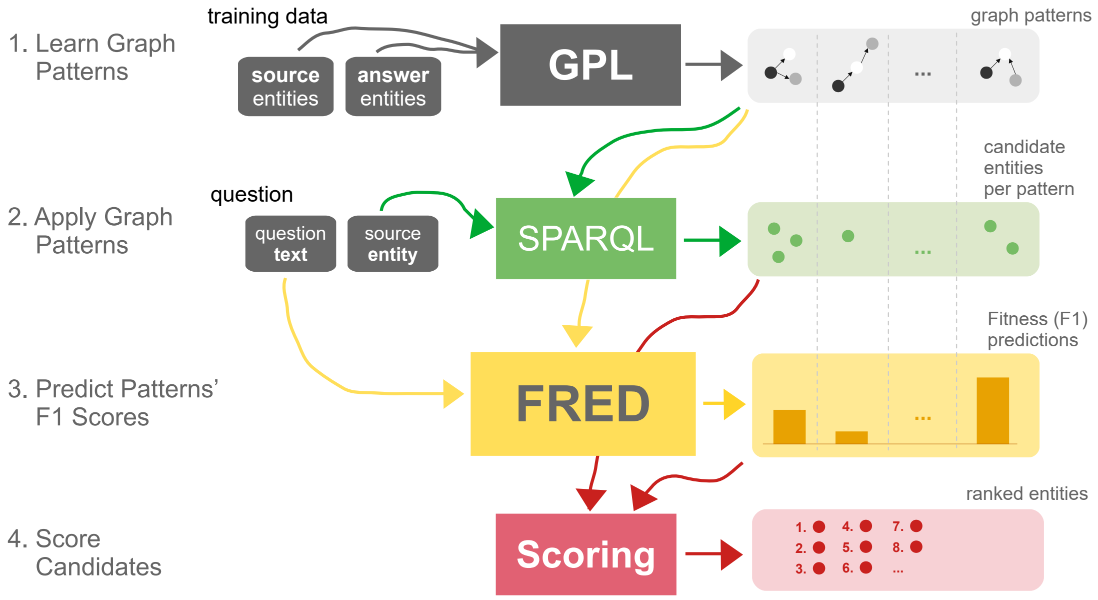

Figure 1: Our approach learns graph patterns (1.) which retrieve candidates given a target questions (2.). We rank these candidates (4.) by predicting the goodness-of-fit between the question and each graph pattern (3.).

introduced to allow the model to be used as a decoder, as BERT usually only represents an encoder. Pre-training of the model employs a massive amount of SPARQL query logs (6.8M) from the public DBpedia endpoint on two tasks: Masked Language Modeling and Word Structural Objective. Fine-tuning uses five different datasets in SPARQL query construction: QALD-9 [18], LC-QuAD [19] and three adaptations of Mon [20].

QUINT [21] learns an ensemble of SPARQL-style query templates from questions and their answers. During training, questions are mapped to relations and types using lexicons, which have been created using distant supervision from a corpus of web pages annotated with Freebase entities. QUINT constructs query templates in three steps: 1. The smallest subgraph containing all of the question's and its answer entities is retrieved and converted into a backbone query. 2. The backbone query is aligned to the question using the lexicons and the question's dependency parse tree. 3. A generalization step converts both the question and its corresponding query into templates. During interference, these templates are matched against a given question and ranked with a random forest classifier. For evaluation purposes, QUINT uses the datasets WebQuestions [13] and Free917 [22].

Overall, our approach differs from SPBERT in that it does not generate queries on the fly but rather learns a repository of graph patterns, and from QUINT since we generate the graph patterns based on an evolutionary search.

3. Approach

Our approach is illustrated in Figure 1: We assume a set of training questions to be given, each question q containing a single *source entity* s from a knowledge graph. We assume the mention of the entity in the question to be known, as well as a set of correct answer (or *target*) entities $\mathcal{T}$. Given this information, our approach features three key components: A graph pattern learner (GPL, gray) learns typical constellations in the KG between source entities and target entities, resulting in a set of *graph patterns* $\mathcal{P}_1, ..., \mathcal{P}_n$. Each pattern is a SPARQL query, and acts as a function that maps source entities s to sets of answer entities $\mathcal{P}_i(s)$. For example, given the source entity s=m.06mt91 (Rihanna), the following pattern could retrieve her nationality: `SELECT DISTINCT ?t WHERE { ?s ns:people.person.nationality ?t.}` Other, more complex patterns could retrieve the names of her albums, etc.

Given an input question, we feed its source entity s to all graph patterns, yielding sets of candidate entities $\mathcal{P}_1(s), ..., \mathcal{P}_n(s)$. The key concern is to estimate which patterns answer the question "well", such that their candidates should be prioritized. To do so, we use a transformer-based prediction model (yellow) which – given a question q, source entity s, and pattern $\mathcal{P}_i$ – estimates the F_1 score of using $\mathcal{P}_i$'s candidate set compared to the question's true answer set $\mathcal{T}$. We use this estimated score to rank the patterns' candidates (red).

3.1. Graph Pattern Learning

The foundation of our approach is a set of graph patterns, each a SPARQL query that takes a source entity (mentioned in a question) and returns a set of target entities. We would like the overall set of SPARQL patterns to match the true target entities with high recall and precision. To learn said patterns, we utilize the graph pattern learner (GPL) by Hees et al. [23, 24]: A population of SPARQL queries is grown by an evolutionary search, which applies various mating and mutation operations such as inserting and grounding variables, splitting and merging variables, expanding nodes and simplifying the pattern, among others. As a pattern's fitness score, the GPL uses several indicators of the coverage and precision with which the pattern yields target entities, how well the pattern complements the population, and how complex it is.

As training data, the GPL utilizes a set of pairs of source and target entities. We derive these pairs from training questions: Assuming that a question features a *single* source entity s and a limited number of target entities $t_1, .., t_m$, we derive m pairs $(s, t_1), ..., (s, t_m)$. We collect all these pairs in a ground truth set $\mathcal{GT}$. Given this ground truth, we apply the GPL with its default parameters.

Preprocessing For performance reasons, the number of source-target pairs should be limited when executing GPL training[1]. Therefore, we partition $\mathcal{GT}$ into chunks $\mathcal{GT}_1, ..., \mathcal{GT}_K$ and run a separate GPL training per chunk. To ease the discovery of patterns, we group semantically similar questions into the same chunk using their relation IDs (as later explained in Section 4). Alternatively, questions could be clustered based on their text embeddings, for example (which we leave for future work).

[1] This is because the GPL – to evaluate patterns' fitness during the training process – executes a SPARQL query against the knowledge graph for each pattern and for all source-target pairs in batch fashion, which can lead to timeouts.

Post-processing Note that the sets of patterns resulting from different chunks may contain redundant and/or similar patterns. Therefore, we apply clustering with different models and metrics, and then select the best patterns (with minimal loss in precision) from each cluster as a representative (for details, refer to [23]). This reduces the overall set of graph patterns by ≈ 91 percent.

3.2. F_1 pREDictor (FRED)

Given a question q with source entity s and a set of graph patterns, we estimate which patterns match the question "well". We measure this goodness-of-fit by the F_1 score $F_1(\mathcal{P}(s), \mathcal{T})$ between the pattern's retrieved candidates $\mathcal{P}(s)$ and the true answer entities $\mathcal{T}$, i.e. a graph pattern $\mathcal{P}$ is defined to match a question q "well" if it retrieves the desired answer entities $\mathcal{T}$ with high precision and recall.

Since the true answer set $\mathcal{T}$ is unknown in practice, we estimate the F_1 score using a regressor. This regressor combines a representation of the question q (derived by a transformer encoder) with a representation of the pattern (derived from its F_1 scores on training questions).

Pattern Representation: Given a pattern $\mathcal{P}$ and m training questions with source entities $s_1, ..., s_m$ and answer sets $\mathcal{T}_1, ..., \mathcal{T}_m$, we collect the pattern's F_1 scores $F_1(\mathcal{P}(s_1), \mathcal{T}_1), ..., F_1(\mathcal{P}(s_m), \mathcal{T}_m)$ in an m-dimensional vector $\mathbf{p}$. This vector carries the information on which questions the pattern tends to work well, and is used as an embedding for the pattern. Note that $\mathbf{p}$ is usually sparse, since an individual pattern answers a specific type of question.

Question Representation: We feed the question q into a BERT encoder [17]. Following common practice, we define the question representation $\mathbf{q}$ by prepending a special classifier token at position zero and using this token's output embedding:

$$\mathbf{q} = BERT(q)_0$$

Also, to obtain a textual representation $\mathbf{s}$ of the source entity s, we average those tokens $t_i, ..., t_j$ that cover s's mention in the question:

$$\mathbf{s} = \frac{1}{j - i + 1} \cdot \sum_{k=i}^{j} BERT(q)_k$$

Both embeddings $\mathbf{q}$ and $\mathbf{s}$ feature $d = 768$ dimensions.

Regressor: The regression model combines a question's text representations $\mathbf{q}, \mathbf{s}$ with n graph pattern representations $\mathbf{p}_1, ..., \mathbf{p}_n$: We first densify each pattern's representation $\mathbf{p}_i$ by mapping it to the same dimensionality as the text embeddings:

$$\mathbf{p}_i^{dense} = tanh(W \cdot \mathbf{p}_i)$$

where $W \in \mathbb{R}^{2d \times m}$ is a learned matrix. We then collect all patterns' dense embeddings in a matrix $\mathbf{P}^{dense} \in \mathbb{R}^{2d \times n}$ and predict all patterns' F_1 scores as:

$$\hat{F}_1(\mathbf{q}, \mathbf{s}, \mathbf{P}^{dense}) = \sigma\Big(\big((\mathbf{q} \circ \mathbf{s}) \cdot \mathbf{P}^{dense}\big) \cdot W'\Big)$$

where $\circ$ denotes vector concatenation, σ is the sigmoid function, and $W' \in \mathbb{R}^{n \times n}$ is another learned matrix. This results in n scores that approximate the F_1 predictions for the n graph patterns.

Training: We train the regressor (including all BERT parameters, W and W') in a supervised fashion on a set of target questions: Given a set of m questions and n graph patterns, we obtain $m \times n$ training samples, since the answer sets $\mathcal{T}$ of training questions are known, and we can compute patterns' true F_1 scores as ground truth. We start training from a pre-trained BERT encoder, and minimize a weighted mean squared error loss, whereas we assign a 10 times higher weight to nonzero F_1 scores (see details on hyperparameters in Section 5.1).

3.3. Scoring

Scoring is given a question q with source entity s and a set of graph patterns whose F_1 scores have been estimated by FRED as described in the last section. We denote these estimates with $\hat{F}_1^1, ..., \hat{F}_1^n$.

We apply each graph pattern to the source entity, obtaining sets of candidate entities $\mathcal{P}_1(s), ..., \mathcal{P}_n(s)$. The overall set of candidate entities is defined as $\mathcal{C} := \cup_{i=1}^n \mathcal{P}_i(s)$. The score of a candidate $c \in \mathcal{C}$ is then defined as:

$$score(c) = \sum_{i=1}^{n} \hat{F}_1^i \ \cdot \ F_1(\{c\}, \mathcal{P}_i(s)) \tag{1}$$

i.e. a candidate will get a high score if it appears as one of few results in patterns that supposedly fit the question well. We rank candidates by their descending score.

4. Datasets

We use the datasets WebQuestionsSP [1] and SimpleQuestions [2] for our experiments. Both are based on the popular knowledge graph *Freebase*[2] [25]. WebQuestionsSP is a collection of 4 737 English questions where each question includes a precise SPARQL query, Freebase IDs for source- and answer entities and the source entity mention in the question. SimpleQuestions is a collection of 108 442 simple, English questions, each with Freebase IDs for the single source entity, the predicate and one answer entity (for questions with > 1 plausible answers, one has been selected randomly).

For both datasets, we build custom splits by shuffling the entire dataset and running it through a filtering pipeline to ensure data integrity and to ensure compatibility with our approach. Further, some measures are taken to limit compute load with the graph pattern learning and subsequent processing of generated patterns.

- **Data Quality**: We remove erroneous instances (such as duplicates or questions with missing source entities), and drop questions with *value* answers, since these cannot be covered by the GPL. For WebQuestionsSP, we only keep questions with a maximum number of answers of 1 and 5 respectively, resulting in two different versions of the

[2] We use a dump from 8th September 2015.

dataset. This simultaneously eliminates outlier questions with a large number of answers (up to 3 688) and also tests the approach's applicability to answer questions with multiple answers.

- **Data Balance**: As discussed in Section 3.1, we partition the training set of source-target pairs into semantically coherent chunks. To do so, the *relation ID* is introduced: For WebQuestionsSP this refers to the *inferential chain*, a set of predicates used in a question's SPARQL query. For SimpleQuestions it refers to the given predicate. To preserve dataset balance, we only keep relation IDs with at least 10/5 and at most 200/200 questions for SimpleQuestions/WebQuestionsSP, and randomly select at most 30 source-target pairs per source entity.
- **Downscaling**: To limit the GPL's execution time, we also downsample the unique source entities to a total number of 2000.

For each setting, we use the remaining data to randomly sample 4 versions of each dataset, over which we report averaged results. We split into training data (90%), on which we train the GPL and FRED, validation data (5%) which we use in GPL training and for early stopping, and test data (5%) on which we report results. During splitting, we ensure that questions with the same source entity are assigned to the same split.

Finally, for the WebQuestionsSP based datasets we utilize the *expert-generated patterns* that come with the dataset (we remove the `FILTER` statements to generalize the patterns, as we found these statements to usually refer to question specific additional entities or values). Since we will use these expert patterns as drop-in replacements for the learned GPL patterns, we will use only those from the training splits. The metrics of the grouped datasets and the number of expert patterns for WebQuestionsSP based datasets are shown in Table 1.

Dataset group	#Questions	#Pairs	#Expert Patterns
WQSP1	2757 (2000, 379, 378)	2165 (1567, 312, 286)	1274
WQSP5	4353 (3182, 587, 584)	5163 (3745, 726, 692)	1773
SQ	8998 (7961, 520, 517)	9006 (7982, 512, 512)	–

Table 1
Overview of the final datasets used in our evaluation. The columns *Questions* and *Pairs* show totals as well as (training, eval, test) splits. Column *Expert* shows the number of generated expert patterns (training split only).

5. Experimental Results

This section presents our setup of the training and evaluating our model setup (Section 5.1), and discusses quantitative results: Control runs give an upper bound on performance based on the generated GPL graph patterns (Section 5.2), a comparison to question agnostic scoring approaches evaluates the added value of FRED's incorporation of text semantics into the answer candidate ranking (Section 5.3), and an ablation study evaluates to what extent FRED relies on the source entity mention during inference (Section 5.4). A comparison between learned and expert patterns evaluates which of both sets of patterns generalize better on test questions, and also evaluates the overlap in both sets (Section 5.5). Finally, the performance improvement achieved when combining both learned and expert patterns is evaluated and compared to the performance of each set of patterns alone (Section 5.6).

5.1. Setup

To evaluate how our FRED approach performs overall, we use the well-known quality measures MAP (when comparing with a reference model that does yield ranked results) and F_1 with a fixed cut-off rank (for reference models that do not).

GPL trainings are performed using the GPL's default parameters[3]. We set the number of chunks such that each contains at most 20 different relation IDs, and subsampled each chunk to max. $\#\mathcal{GT}_k = 800$ samples. FRED trainings are performed using Adamax [26] with an initial learning rate of $\lambda = 5e{-}5$, batch size 4, and apply a dropout [27] rate of 10% before multiplying with W'. We trained for a maximum of 200 epochs, whereas early stopping is applied using FRED's MAP on the validation split. We start our training from the pre-trained BERT checkpoint *bert-base-uncased*.

5.2. Control Runs

A first interesting question is what maximum performance could be achieved with our learned graph patterns, assuming a perfect matching between question and patterns. To estimate this upper bound, the results of FRED scoring at different cutoff values are compared against a simulated *oracle classifier*, which selects the *best fitting* pattern for a question, i.e. the one achieving the highest F1 score, and yields this pattern's result set. The results in Table 2 show that the highest scored entity by FRED comes close to the results of the best fitting GPL pattern. FRED attains near best possible performance on WQSP1 and SQ, whereas the difference to the upper bound placed by the GPL is greater for WQSP5.

Dataset	GPL Oracle	$\mathbf{FRED_1}$	$\mathbf{FRED_2}$	$\mathbf{FRED_3}$	$\mathbf{FRED_4}$	$\mathbf{FRED_5}$
WQSP5	0.79 ± 0.02	0.63 ± 0.03	0.58 ± 0.02	0.55 ± 0.02	0.51 ± 0.02	0.49 ± 0.03
WQSP1	0.79 ± 0.06	0.73 ± 0.06	0.58 ± 0.05	0.50 ± 0.03	0.46 ± 0.03	0.44 ± 0.02
SQ	0.53 ± 0.12	0.47 ± 0.11	0.39 ± 0.09	0.35 ± 0.08	0.34 ± 0.08	0.32 ± 0.07

Table 2
Results of the control runs evaluation, reported as the mean and standard deviation of the F1 score achieved on the test split over 4 runs. *GPL Oracle* shows an optimal matching between question and graph patterns. The columns $FRED_k$ show results for the top-k results of FRED scoring.

5.3. Question Agnostic Scoring

The graph pattern learner itself comes with several approaches for ranking its target candidates [23]. These approaches prioritize graph patterns based on their precision. Obviously, these approaches offer an alternative to our FRED scoring. However, while FRED incorporates

[3] https://github.com/RDFLib/graph-pattern-learner

information from the input question, the GPL's own scoring approaches only utilize information from the GPL training and the resulting graph patterns to score a target candidate.

To evaluate the added value of our model compared to such question-agnostic scoring, FRED (see Equation 1) is compared against three GPL-internal scoring mechanisms *Precisions*, *GP Precisions* and *Logistic Regression*. The results are shown in Table 3. We can see that FRED provides a significant improvement over these question-agnostic scoring approaches.

Dataset	FRED	Precisions	GP Precisions	Logistic Regression
WQSP5	0.69 ± 0.04	0.26 ± 0.02	0.32 ± 0.03	0.40 ± 0.02
WQSP1	0.73 ± 0.06	0.28 ± 0.05	0.31 ± 0.04	0.45 ± 0.07
SQ	0.46 ± 0.10	0.40 ± 0.12	0.42 ± 0.10	0.43 ± 0.09

Table 3

Results of the evaluation of alternate scoring approaches. Results are measured as the MAP achieved on the test split and reported as the mean and standard deviation over all 4 runs on the respective dataset version. FRED's incorporation of the question provides a performance gain compared to the other, question-agnostic scoring approaches.

5.4. Ablation Study

To evaluate to which extent the FRED model can be simplified without impacting performance, an ablation study is performed, where the model omits the source mention $\mathbf{s}$ embedding and instead relies solely on the question embedding $\mathbf{q}$ (see Section 3.2). The forward pass becomes

$$\hat{F}_1(\mathbf{q}, \mathbf{P}^{dense}) = \sigma\Big(\Big(\mathbf{q} \cdot \mathbf{P}^{dense}\Big) \cdot W'\Big)$$

where the size of W is adjusted to $d \times m$ dimensions, and consequently $\mathbf{P}^{dense} \in \mathbb{R}^{d \times n}$, i.e. the dense pattern embeddings now lie in a d-dimensional space instead of a $2d$-dimensional one.

The results of the evaluation are presented in Table 4 and show that the source mention embedding can be omitted without a significant impact on the model's performance. Our interpretation is that the question embedding already captures enough semantics of a question, such that the model is not required to interpret a question's source mention embedding.

Dataset	$\mathbf{FRED}$	$\mathbf{FRED}_{\overline{\mathbf{s}}}$
WQSP5	0.69 ± 0.04	0.68 ± 0.05
WQSP1	0.73 ± 0.06	0.73 ± 0.06
SQ	0.46 ± 0.10	0.46 ± 0.10

Table 4

Results of the ablation study: $FRED_{\overline{\mathbf{s}}}$ does not use the source mention embedding $\mathbf{s}$, as described above. Results are given as the mean and standard deviation of the MAP on achieved the test split over all 4 runs on each respective dataset. The table shows that the omission of source mention embedding does not impact results significantly.

5.5. Comparison between Learned and Expert Patterns

An example for three questions (**Q**) from WQSP1, their corresponding expert patterns (**EP**) and the best matching GPL pattern (**GPL**) is given in the following. $\boldsymbol{\Delta}$ denotes the cosine distance between expert- and GPL pattern in the embedding space spanned by the graph patterns' representations $\mathbf{p}$ (see Section 3.2).

- **Q**: *who was gerald ford vp?*
  EP:
  ```
  SELECT DISTINCT ?source ?target WHERE {
  ?source ns:government.us_president.vice_president ?target .}
  ```
  GPL:
  ```
  SELECT ?source ?target ?vcb0 WHERE {
  ?target key:wikipedia.lv ?vcb0 .
  ?target ns:government.us_vice_president.to_president ?source .}
  ```
  $\boldsymbol{\Delta}$: ≈ 0.0
- **Q**: *what language did the ancient babylonians speak?*
  EP:
  ```
  SELECT DISTINCT ?source ?target WHERE {
  ?source ns:location.country.languages_spoken ?target .}
  ```
  GPL:
  ```
  SELECT ?source ?target ?vcb0 WHERE { ?target
  ns:language.human_language.countries_spoken_in ?source .
  ?target ns:language.human_language.writing_system ?vcb0 .}
  ```
  $\boldsymbol{\Delta}$: 0.0065
- **Q**: *what capital city of brazil?*
  EP:
  ```
  SELECT DISTINCT ?source ?target WHERE {
  ?source ns:location.country.capital ?target .}
  ```
  GPL:
  ```
  SELECT ?source ?target ?vcb0 WHERE {
  ?source ns:location.country.capital ?target .
  ?target ns:travel.travel_destination.tourist_attractions ?vcb0 . }
  ```
  $\boldsymbol{\Delta}$: 0.2929

We see that in the first two cases, the learned graph patterns and the expert patterns are similar both in terms of the pattern's graph structure, and in terms of the entities the patterns retrieve (as expressed by $\boldsymbol{\Delta}$). In the third case, while the GPL found the predicate from the expert pattern, it added another restrictive predicate which lead to a greater difference in the result set. Overall, the examples above show that the GPL is able to learn patterns similar but not necessarily identical to the expert patterns delivered with WebQuestionsSP.

We investigate this question further quantitatively: For each expert pattern in the training set, we find the closest GPL-learned pattern in terms of the distance $\boldsymbol{\Delta}$ between the pattern representations. We collect the distribution of this nearest-neighbor distance in Figure 2 in a histogram, where 0 (left) corresponds to a low distance, i.e. expert patterns for which very similar GPL patterns can be found, and 100 (right) expert patterns that are dissimilar from any learned graph pattern. The plot indicates that for about 18% of all expert patterns a very similar GPL pattern was learned ($\boldsymbol{\Delta} = 0$), while at the same time for 18% no similar GPL pattern is found at all ($\boldsymbol{\Delta} = 100$).

Finally, we investigate the question how well both expert and learned patterns *cover* the targeted questions. We utilize the respective pattern populations drawn from / learned on the

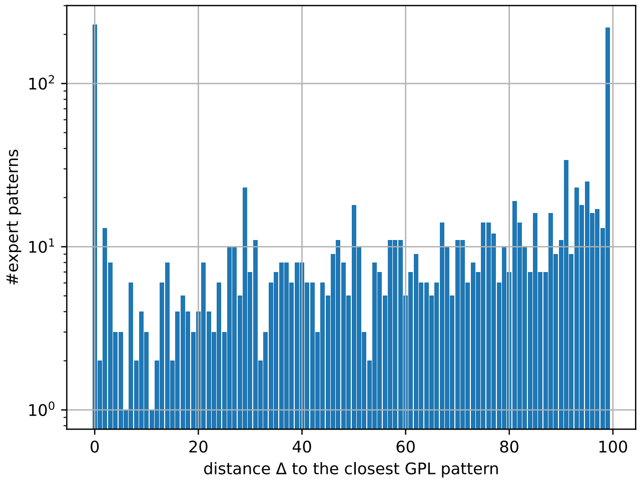

Figure 2: Histogram of the number of nearest GPL pattern neighbors for each expert pattern. The x-axis shows the cosine distance, where 100 denotes maximum distance. The y-axis shows the number of GPL patterns with less or equal distance. The cosine distance is measured in the embedding space spanned by the train-questions coverage of the graph patterns. It shows, that for approximately half of all expert patterns, an identical GPL pattern was generated.

training set, and count for each pattern the number of training/test questions that are "answered" by the pattern (indicated by an F1 score > 0). This gives an indicator about the generality of both sets of patterns. The results for the WebQuestionsSP datasets are shown in Table 5: For example, the first row indicates that a GPL-learned pattern covers 45.41 questions from the training set on average. We observe that the counts on the training set are much higher than on the test sets (which is mostly because the training set is 19 times larger). More importantly, however, we observe that GPL-learned patterns cover significantly more (≈ 4 times as many) questions than the expert-defined patterns, both on the training and test split. Correspondingly, the GPL patterns are more "general" than the expert patterns.

		# matched questions	
Dataset	**Split**	**GPL patterns**	**expert patterns**
WQSP1	train	45.41	10.45
WQSP1	test	2.15	0.48
WQSP5	train	58.38	12.60
WQSP5	test	2.98	0.63

Table 5
Metrics about the number of fitting questions per pattern. The columns *# matched questions* show the average number of questions for which a F1 score greater 0 was achieved on average per pattern. The higher numbers for the GPL patterns show that these are more general, covering more questions on average.

5.6. Complementing Expert Patterns

We compare the accuracy obtained by FRED when using a population of (a) expert-defined patterns, (b) GPL-learned patterns, and (c) the union of both sets. Results on the test splits are illustrated in Table 6: The expert patterns have been derived from the precise SPARQL queries delivered with WebQuestionsSP and – as expected – are more accurate than the GPL generated patterns. While they do in fact achieve a higher average F1, the combination of both GPL- and expert patterns performs better than each set of pattern alone. This indicates that the GPL-learned patterns can offer a high-quality, low-cost alternative to manually defining patterns.

Dataset	Mean F1	Average result len
WQSP1 GPL	0.79 ± 0.06	10.83 ± 16.81
WQSP1 Expert	0.81 ± 0.06	4.18 ± 2.85
WQSP1 GPL+Expert	$\mathbf{0.86 \pm 0.05}$	12.34 ± 15.73
WQSP5 GPL	0.79 ± 0.02	4.75 ± 2.56
WQSP5 Expert	0.82 ± 0.03	5.59 ± 3.93
WQSP5 GPL+Expert	$\mathbf{0.86 \pm 0.02}$	4.89 ± 2.43

Table 6
Results are measured as the F1 score achieved on the test split and presented as the mean and standard deviation over all datasets in the respective group. It is shown, that the combination of both expert and GPL patterns achieves the best results.

6. Concluding Discussion

This paper has presented FRED, an approach towards KBQA which combines an evolutionary learning of graph patterns with a transformer-based matching of questions to patterns. Our results indicate the general feasibility of the model, and demonstrate that our matching yields an accuracy close to optimal pattern-based control runs. We have also compared the patterns yielded by our approach to expert-generated patterns, and have demonstrated that both types of patterns complement each other well.

Some limitations of our approach are of a rather technical nature: The current implementation of the graph pattern learner assumes a single source entity per question to be given, and does not cope with value answers (such as dates). Second, our current strategy of partitioning the GPL training set is based on relation IDs (which are unlikely to be available in practice). An interesting direction of future work will be to investigate different clustering strategies of questions, likely based on text embeddings. Also, note that similar to other KBQA approaches, our current model requires the source entity and its mention in the question to be known (though we achieved comparable results without the latter in an ablation study). Investigating a more extended pipeline that includes entity localization and linking might be of interest here. Another future direction is a comparison against other query generating KBQA approaches, like SPBERT and QUINT. These were ommited due to the focus on a different KB (DBpedia [28]) in the first case and the absence of an available reference implementation in the latter.

Most significantly, however, weighs the fact that our approach considers graph patterns to be atomic, i.e. there are no combinations of generalizations of patterns at inference time. For example, a pattern cannot generalize from finding an actor's spouse to a singer's spouse unless the GPL has learned other individuals as part of its population. Finding strategies for more adaptive patterns is certainly an interesting direction for future work.

References

[1] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, J. Suh, The Value of Semantic Parse Labeling for Knowledge Base Question Answering, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2016, pp. 201–206.

[2] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale Simple Question Answering with Memory Networks (2015).

[3] OpenAI ChatGPT website, https://chat.openai.com/, ???? Accessed: 2023-05-09.

[4] Google Bard website, https://bard.google.com/, ???? Accessed: 2023-05-09.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, LLaMA: Open and Efficient Foundation Language Models, arXiv preprint, arXiv:2302.13971 (2023).

[6] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, T. B. Hashimoto, Stanford Alpaca: An Instruction-following LLaMA model, https://github.com/tatsu-lab/stanford_alpaca, 2023.

[7] Dolly Model Card on Hugging Face, https://huggingface.co/databricks/dolly-v2-12b, ???? Accessed: 2023-05-09.

[8] A. Creswell, M. Shanahan, Faithful Reasoning Using Large Language Models, 2022. arXiv:2208.14271.

[9] X. Huang, J. Zhang, D. Li, P. Li, Knowledge Graph Embedding Based Question Answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, p. 105–113.

[10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 26, 2013.

[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, p. 2181–2187.

[12] C. Mavromatis, G. Karypis, ReaRev: Adaptive Reasoning for Question Answering over Knowledge Graphs, arXiv preprint arXiv:2210.13650 (2022).

[13] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic Parsing on Freebase from Question-Answer Pairs, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1533–1544.

[14] A. Talmor, J. Berant, The Web as a Knowledge-Base for Answering Complex Questions, in: Proceedings of the 2018 Conference of the North American Chapter of the Association

for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 641–651.

[15] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, L. Song, Variational Reasoning for Question Answering with Knowledge Graph, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, 2018.

[16] H. Tran, L. Phan, J. Anibal, B. T. Nguyen, T.-S. Nguyen, SPBERT: An Efficient Pre-training BERT on SPARQL Queries for Question Answering over Knowledge Graphs, 2021.

[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pp. 4171–4186.

[18] R. Usbeck, R. H. Gusmita, A. N. Ngomo, M. Saleem, 9th Challenge on Question Answering over Linked Data (QALD-9) (invited paper), in: 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018., 2018, pp. 58–64.

[19] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, LC-QuAD: A corpus for complex question answering over knowledge graphs, in: International Semantic Web Conference, Springer, 2017, pp. 210–218.

[20] T. Soru, E. Marx, D. Moussallem, G. Publio, A. Valdestilhas, D. Esteves, C. B. Neto, SPARQL as a Foreign Language, in: Proceedings of the 13th International Conference on Semantic Systems - SEMANTiCS2017 Posters and Demos, 2017.

[21] A. Abujabal, M. Yahya, M. Riedewald, G. Weikum, Automated Template Generation for Question Answering over Knowledge Graphs, in: Proceedings of the 26th International Conference on World Wide Web, 2017, p. 1191–1200.

[22] Q. Cai, A. Yates, Large-scale Semantic Parsing via Schema Matching and Lexicon Extension, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013.

[23] J. Hees, Simulating Human Associations with Linked Data, Ph.D. thesis, Technical University Kaiserslautern, 2018.

[24] J. Hees, R. Bauer, J. Folz, D. Borth, A. Dengel, An Evolutionary Algorithm to Learn SPARQL Queries for Source-Target-Pairs, in: E. Blomqvist, P. Ciancarini, F. Poggi, F. Vitali (Eds.), Knowledge Engineering and Knowledge Management, 2016, pp. 337–352.

[25] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, p. 1247–1250.

[26] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2014.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.

[28] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, C. Bizer, DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia, Semantic Web Journal 6 (2014).