Explainable Conversational Question Answering over Heterogeneous Sources via Iterative Graph Neural Networks

Philipp Christmann
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbruecken, Germany
pchristm@mpi-inf.mpg.de

Rishiraj Saha Roy
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbruecken, Germany
rishiraj@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbruecken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

In conversational question answering, users express their information needs through a series of utterances with incomplete context. Typical ConvQA methods rely on a single source (a knowledge base (KB), *or* a text corpus, *or* a set of tables), thus being unable to benefit from increased answer coverage and redundancy of multiple sources. Our method EXPLAIGNN overcomes these limitations by integrating information from a mixture of sources with user-comprehensible explanations for answers. It constructs a heterogeneous graph from entities and evidence snippets retrieved from a KB, a text corpus, web tables, *and* infoboxes. This large graph is then iteratively reduced via graph neural networks that incorporate question-level attention, until the best answers and their explanations are distilled. Experiments show that EXPLAIGNN improves performance over state-of-the-art baselines. A user study demonstrates that derived answers are understandable by end users.

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

Question Answering, Explainability, Graph Neural Networks

1 INTRODUCTION

Motivation. In conversational question answering (ConvQA), users issue a sequence of questions, and the ConvQA system computes crisp answers [36, 42, 44]. The main challenge in ConvQA systems is that inferring answers requires understanding the current context, since incomplete, ungrammatical and informal *follow-up questions* make sense only when considering the conversation history so far.

Existing ConvQA models mostly focused on using either (i) a curated knowledge base (KB) [20–22, 24, 29, 48], or (ii) a text corpus [4, 15, 35, 36, 38], or (iii) a set of web tables [16, 31] as source to compute answers. These methods are not geared for tapping into multiple sources jointly, which is often crucial as one source could compensate for gaps in others. Consider the conversation:

q^1: *Who wrote the book Angels and Demons?*
　a^1: `Dan Brown`
q^2: *the main character in his books?*
　a^2: `Robert Langdon`
q^3: *who played him in the films?*
　a^3: `Tom Hanks`
q^4: *to which headquarters was robert flown in the book?*
　a^4: `CERN`
q^5: *how long is the novel?*
　a^5: `768 pages`
q^6: *what about the movie?*
　a^6: `2 h 18 min`

Some of these questions can be conveniently answered using a KB (q^1, q^3), tables (q^5, q^6), or infoboxes (q^1, q^3) as they ask about salient attributes of entities, and some via text sources (q^2, q^3, q^4) as they are more likely to be contained in book contents and discussion. However, none of these individual sources represents the whole information required to answer *all questions* of this conversation.

Recently, there has been preliminary work on ConvQA over a mixture of input sources [8, 9]. This improves the recall for the QA system with higher *answer coverage*, and the partial *answer redundancy* across sources helps improve precision.

Limitations of state-of-the-art methods. Existing methods for ConvQA over heterogeneous sources rely on neural sequence-to-sequence models to compute answers [8, 9]. However, this has two significant limitations: (i) sequence-to-sequence models are *not explainable*, as they only generate strings as outputs, making it infeasible for users to decide whether to trust the answer; (ii) sequence-to-sequence models require inputs to be cast into token sequences first. This loses insightful information on relationships between evidences [61]. Such inter-evidence connections can be helpful in separating relevant information from noise.

Approach. We introduce EXPLAIGNN[1] (EXPLAinable Conversational Question Answering over Heterogeneous Graphs via Iterative Graph Neural Networks), a flexible pipeline that can be configured for optimizing *performance, efficiency, and explainability* for ConvQA systems over heterogeneous sources. The proposed method operates in three stages:

---

[1]Code, data, extended version, and demo at https://explaignn.mpi-inf.mpg.de.
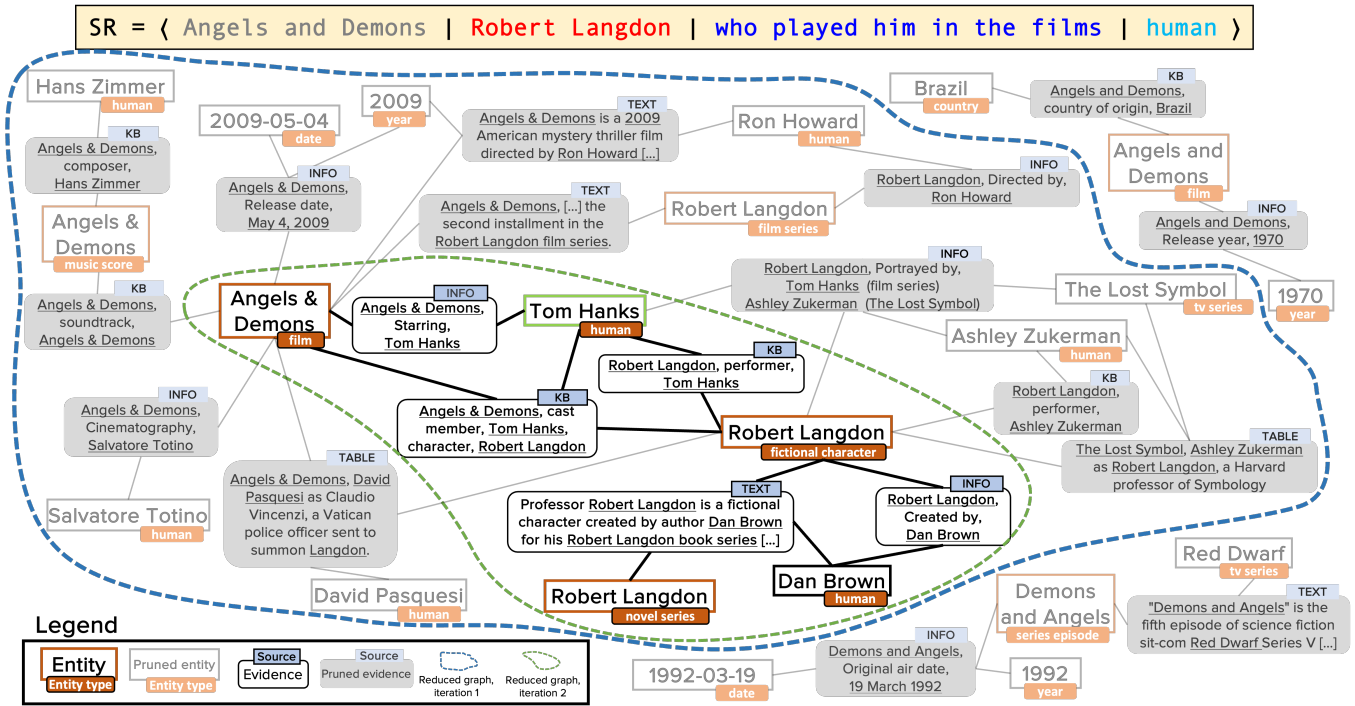
Figure 1: Toy heterogeneous graph for answering q^3, showing two pruning iterations. The graph is iteratively reduced by GNN inference to identify the key evidences. The subgraph surrounded by the blue dotted line is the result of the first iteration, while the green line indicates the graph after the second. From this smaller subgraph, the final answer (Tom Hanks) is inferred.

(i) Derivation of a self-contained *structured representation (SR)* of the user's information need (or intent) from the potentially incomplete input utterance and the conversational context, making the entities, relation, and expected answer type explicit.

(ii) Retrieval of relevant evidences and answer candidates from heterogeneous information sources: a curated KB, a text corpus, a collection of web tables, and infoboxes.

(iii) Construction of a graph from these evidences, as the basis for applying graph neural networks (GNNs). The GNNs are iteratively applied for computing the best answers and supporting evidences in a small number of steps.

A key novelty is that each iteration reduces the graph in size, and only the final iteration yields the answer and a small user-comprehensible set of explanatory evidences. Our overarching goal is to provide end-user *explain*ability to *GNN* inference by *iteratively* reducing the graph size, so that the final answers can indeed be claimed to be causal w.r.t. the remaining evidences, hence the name EXPLAIGNN. A toy example of such GNN-based reduction is in Fig. 1.

Contributions. We make the following salient contributions:

- Proposing a new method for ConvQA over heterogeneous sources, with a focus on computing explainable answers.
- Devising a mechanism for iterative application of GNN inference to reduce such graphs until the best answers and their explanatory evidences are obtained.
- Developing an attention mechanism which ensures that during message passing only question-relevant information is spread over the local neighborhoods.

2 CONCEPTS AND NOTATION

Question. A *question q* asks about *factoid* information, like *Who wrote the book Angels and Demons?* (intent is explicit), or *How long is the novel?* (intent is implicit).

Answer. An *answer a* to *q* can be an entity (like Tom Hanks), or a literal (like 768 pages).

Conversation. A *conversation* is a sequence of questions and answers $\langle q^1, a^1, q^2, a^2, \dots \rangle$. The initial question q^1 is complete, i.e. makes the information need intent-explicit. The follow-up questions q^t (t>1) may be incomplete, building upon the ongoing conversational history, therefore leaving context information implicit.

Turn. A conversation turn t comprises a $\langle q^t, a^t \rangle$ pair.

Knowledge base. A curated *knowledge base* is defined as a set of facts. Each fact consists of a subject, a predicate, an object, and an optional series of ⟨qualifier-predicate, qualifier-object⟩ pairs: $\langle s, p, o; qp_1, qo_1; qp_2, qo_2; \dots \rangle$. An example fact is: ⟨Angels and Demons, cast member, Tom Hanks; character, Robert Langdon⟩.

Text corpus. A *text corpus* consists of a set of text documents.

Table. A *table* is a structured form of representing information, and is typically organized into a grid of rows and columns. Individual rows usually record information corresponding to specific entities, while columns refer to specific attributes for these entities. The row-header, column-header, and the cell hold the entity name, attribute name, and the attribute value, respectively.

Infobox. An *infobox* consists of several entries that are ⟨attribute name, attribute value⟩ pairs, and provide salient information on a certain entity. An infobox can be perceived as a special instantiation
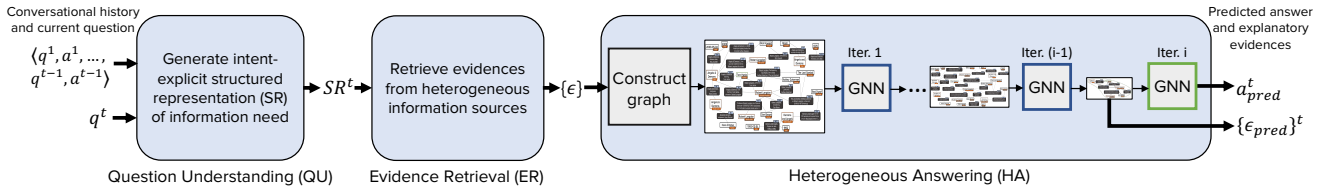
Figure 2: An overview of the Explaignn pipeline, illustrating the three main stages of the approach.

of a table, recording information on a single entity, and consisting of exactly two columns and a variable number of rows.

Evidence. An *evidence* ϵ is a short text snippet expressing factual information, and can be retrieved from a KB, a text corpus, a table, and an infobox. To be specific, *evidences* are verbalized KB-facts, text-sentences, table-records, or infobox-entries.

Structured representation. The *structured representation SR* [8] for q is an intent-explicit version of the question. The SR represents the current question using four slots: (i) *context entity*, (ii) *question entity*, (iii) *relation*, (iv) *expected answer type*. This intent-explicit representation can be represented in linear form as a single string, using delimiters to separate slots ('|' in our case). The SR for q^3 is:

⟨ Angels and Demons | Robert Langdon |
who played him in the films | human ⟩

In this example, Angels and Demons is the context entity, Robert Langdon the question entity, who played him in the films the relation, and human the expected answer type. We consider a relaxed notion of relations, in the sense of not being tied to KB terminology that canonicalizes textual relations to predicates. This allows for softer matching in evidences, and answering questions for which the information cannot easily be represented by such predicates.

3 OVERVIEW

The architecture of Explaignn (Fig. 2) follows the pipeline of Convinse [8]: (i) an intent-explicit structured representation of the information need is generated, (ii) evidences are retrieved from heterogeneous sources, and (iii) this set of relevant evidences is used for answering the question and providing explanatory evidences.

3.1 Question understanding

We use [8] and generate a structured representation (SR) capturing the complete intent in the current question and the conversational context. For generating SRs, we leverage a fine-tuned auto-regressive sequence-to-sequence model (BART [25]).

Preventing hallucination. We propose a novel mechanism to avoid *hallucinations* [30] in SRs. For q^3 of the running example, the trained model could generate Robert de Niro as the (topically unrelated) question entity of the output SR ⟨ Dan Brown | Robert de Niro | who played him in the films | human ⟩. This would lead the entire QA system astray, and could therefore confuse end users.

The SR is supposed to represent the information need on the *surface level*, and therefore expected to use the vocabulary present in the input (the conversational history and current question). This makes it possible to identify hallucinations: output words that are absent from the entire conversation so far, indicate such a situation. To fix this, we generate the top-k SRs (k=10 in experiments), and

choose the highest-ranked SR that does not include any hallucinated words. Note that the expected answer type is an exception here: it may, by design, not be present in the input. So we remove this slot before performing the hallucination check. Answer types are often not made explicit but substantially help the QA system [44].

3.2 Evidence retrieval

KB evidences and entity disambiguations are obtained via running Clocq [6] on the SR (without delimiters). *Text, table and infobox evidences* are obtained by mapping the disambiguated KB-entities to Wikipedia pages, which are then parsed for extracting text-sentences, table-records, and infobox-entries corresponding to the respective entities. Evidences from KB, web tables, or infoboxes, being natively in (semi-)structured form, are then *verbalized* [28, 33] into token sequences (as in [8]). Examples can be seen inside Fig. 1, where each evidence is tagged with its source.

Use of SR slot labels. Convinse considers all entities in the SR during retrieval, regardless of the slot in which they appear. In contrast, Explaignn *restricts* the entities by retaining only those mentioned within the *context entity* or *question entity* slots. Evidences are then only retrieved for this restricted set of entities. This prunes noisy disambiguations in the relation and type slots.

4 HETEROGENEOUS ANSWERING

We first describe heterogeneous answering graph construction (Sec. 4.1). Next, we present the proposed question-aware GNN architecture, consisting of the encoder (Sec. 4.2), the message passing procedure (Sec. 4.3), the answer candidate scoring (Sec. 4.4), and the multi-task learning mechanism for GNN training (Sec. 4.5).

4.1 Graph construction

Given the evidences retrieved in the previous stage, we construct a *heterogeneous answering graph* that has two types of nodes: entities and evidences. The graph contains *textual information* as entity labels and evidence texts, as well as the *connections* between these two kinds of nodes. Specifically, an entity node e is connected to an evidence node ϵ, if e is mentioned in ϵ. There are no direct edges between pairs of entities, or pairs of evidences. An example heterogeneous graph is shown in Fig. 1.

Inducing connections between retrieved evidences. Shared entities are the key elements that induce *relationships* between the initial plain *set* of retrieved evidences. So one requires entity markup on the verbalized evidences coming from the different sources, that are grounded to a KB for *canonicalization*. Note that during evidence verbalization, original formats are not discarded. Thus, for KB-facts, entity mappings are already known. For text, table, and infobox evidences from Wikipedia, we link anchor texts

to their referenced entity pages. These are then mapped to their corresponding KB-entities. In absence of anchor texts, named entity recognition and disambiguation systems can be used [14, 26, 55]. Dates and years that appear in evidences are detected using regular expressions, and are added as entities to the graph as well. In Fig. 1, entity mentions are underlined within evidences.

4.2 Node encodings

GNNs incrementally update node encodings within local neighborhoods, leveraging message passing algorithms. However, these node encodings have to be initialized first using an encoder.

Evidence encodings. For the initial encoding of the nodes, we make use of cross-encodings [43] (originally proposed for sentence pair classification tasks in [10]). The evidence text, concatenated with the SR, is fed into a pre-trained language model. By using SR-specific cross-encodings, we ensure that the node encodings capture the information relevant for the current question, as represented by the SR. The encodings obtained for the individual tokens are averaged, yielding the initial evidence encoding $\epsilon^0 \in \mathbb{R}^d$.

Entity encodings. The entity encodings are derived analogously, using cross-encodings with the SR. We further append the KB-type of an entity to the entity label with a separator, before feeding the respective token sequence into the language model. Including entity types is beneficial, and often crucial:

(i) The cross-encoding can leverage the attention between the *expected answer type* in the SR and the *entity type*, which can be viewed as a soft-matching between the two.

(ii) When facing multiple entities with the same label, the entity type may be a discriminative factor (e.g., there are three entities of different types named *"Robert Langdon"* in Fig. 1).

(iii) For long-tail entities, the entity type can add decisive informative value to the plain entity label.

Analogous to evidence nodes, the encodings of individual tokens are averaged to obtain the entity encoding $e^0 \in \mathbb{R}^d$.

SR encoding. The SR is also encoded via the language model, averaging the token encodings to obtain the SR encoding $SR \in \mathbb{R}^d$.

Note that the same language model is used for the initial encodings of evidences, entities and the SR. The parameters of this language model are updated during GNN training, to ensure that the encoder adapts to the syntactic structure of the SR.

4.3 Message passing

The core part of the GNN is the *message passing* [53, 60] procedure. In this step, information is propagated among neighboring nodes, leveraging the graph structure. Given our graph design, in each message passing step information is shared between evidences and the connected entities. Again, we aim to focus on question-relevant information [12], as captured by the SR, instead of spreading general information within the graph. Therefore, we propose to weight the messages of neighboring entities using a novel *attention mechanism*, that re-weights the messages by their question, or equivalently, their SR relevance. This *SR-attention* is computed by $\alpha^l_{\epsilon,e}$ ($\in \mathbb{R}$):

$$\alpha^l_{\epsilon,e} = \operatorname*{softmax}_{\mathcal{N}(\epsilon)}\left(\mathrm{lin}^l_{\alpha_\epsilon}(e^{l-1}) \cdot SR\right) = \frac{\mathrm{lin}^l_{\alpha_\epsilon}(e^{l-1}) \cdot SR}{\sum\limits_{e_i \in \mathcal{N}(\epsilon)} \mathrm{lin}^l_{\alpha_\epsilon}(e_i^{l-1}) \cdot SR} \quad (1)$$

where we first project the entity encodings using a linear transformation ($\mathrm{lin}^l_{\alpha_\epsilon} : \mathbb{R}^d \to \mathbb{R}^d$), and then multiply with the SR encoding to obtain a score. The softmax function is then applied over all entities neighboring a respective evidence ($e_i \in \mathcal{N}(\epsilon)$). Thus, an entity can obtain different SR-attention scores for each evidence, depending on the scores of other neighboring entities.

The messages passed to ϵ are then aggregated, weighted by the respective SR-attention, and projected using another linear layer:

$$m^l_\epsilon = \mathrm{lin}^l_{m_\epsilon}\left(\sum_{e \in \mathcal{N}(\epsilon)} \alpha^l_{\epsilon,e} \cdot e^{l-1}\right) \quad (2)$$

where $\mathrm{lin}^l_{m_\epsilon}$ is the linear layer ($\mathrm{lin}^l_{m_\epsilon} : \mathbb{R}^d \to \mathbb{R}^d$).

The updated evidence encoding is then given by adding the evidence encoding from the previous layer (ϵ^{l-1}), and the messages passed from the neighbors (m^l_ϵ), activated by a ReLU function:

$$\epsilon^l = \mathrm{ReLU}(m^l_\epsilon + \epsilon^{l-1}) \quad (3)$$

The intuition here is that in each evidence update, the question-relevant information held by neighboring entities is passed on to an evidence, and then incorporated in its encoding.

The process for updating the entity encodings is analogous, but makes use of different linear transformation functions. The SR-attention $\alpha^l_{e,\epsilon}$ of evidences for an entity e is obtained as follows:

$$\alpha^l_{e,\epsilon} = \operatorname*{softmax}_{\mathcal{N}(e)}\left(\mathrm{lin}^l_{\alpha_e}(\epsilon^{l-1}) \cdot SR\right) \quad (4)$$

where $\mathrm{lin}^l_{\alpha_e}$ ($\mathbb{R}^d \to \mathbb{R}^d$) is the linear transformation function. Here, the softmax function is applied over all evidences surrounding the respective entity (i.e. $\epsilon_i \in \mathcal{N}(e)$). Again, the messages passed to an entity e are weighted by the respective SR-attention, and projected using a linear layer ($\mathrm{lin}^l_{m_e} : \mathbb{R}^d \to \mathbb{R}^d$):

$$m^l_e = \mathrm{lin}^l_{m_e}\left(\sum_{\epsilon \in \mathcal{N}(e)} \alpha^l_{e,\epsilon} \cdot \epsilon^{l-1}\right) \quad (5)$$

The updated entity encoding is then given by:

$$e^l = \mathrm{ReLU}(m^l_e + e^{l-1}) \quad (6)$$

These message passing steps are repeated L times, i.e. the GNN has L layers. Within these layers the question-relevant information is spread over the graph. Basically, nodes in the graph learn about their question relevance, based on the surrounding nodes and their relevance, and capture this information in their node encodings.

4.4 Answer score prediction

Scoring answer candidates makes use of the node encodings obtained after L message passing steps. We model the answer prediction as a *node classification* task [19, 51, 60], by computing an answer score for each entity node with consideration of their question relevance as captured within the node encodings. The computation of the answer score s_e is similar to the technique used for computing the SR-attention of an entity:

$$s_e = \operatorname*{softmax}_E\left(\mathrm{lin}_e(e^L) \cdot SR\right) \quad (7)$$

We project the entity encoding using a linear layer ($\mathrm{lin}_e : \mathbb{R}^d \to \mathbb{R}^d$), and multiply the projected encoding with the encoding of the SR. The softmax function is applied over all entity nodes (i.e. $e_i \in E$).
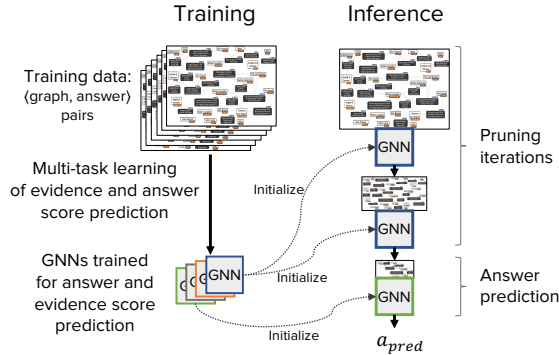
Figure 3: Training of and inference with iterative GNNs.

4.5 Multi-task learning

Our training data for the GNN consists of ⟨graph, answer⟩ pairs. The gold answer is always an entity or a small set of entities. Consequently, the positive training data is sparse: there can be hundreds of entities in the graph but only one gold answer.

To better use our training data, we propose a *multi-task learning* (MTL) [23, 48] approach. Given a GNN, we pose two complementary node classification tasks: (i) the answer prediction, and (ii) the prediction of evidence relevance. Evidences connected to gold answers are viewed as relevant, and others as irrelevant. Our method learns to predict a relevance score s_ϵ for each evidence node ϵ, analogous to the answer score prediction:

$$s_\epsilon = \underset{\epsilon \in \mathcal{E}}{\text{softmax}}\Big(\text{lin}_\epsilon(\epsilon^L) \cdot SR\Big) \quad (8)$$

where $\mathcal{E}$ is the set of all evidence nodes (and $\text{lin}_\epsilon : \mathbb{R}^d \to \mathbb{R}^d$).

For both tasks, answer prediction and evidence prediction, we use binary-cross-entropy over the predicted scores as loss functions: $\mathcal{L}_e$ and $\mathcal{L}_\epsilon$, respectively. The final loss used for training the GNN is then defined as a weighted sum:

$$\mathcal{L} = w_e \cdot \mathcal{L}_e + w_\epsilon \cdot \mathcal{L}_\epsilon \quad (9)$$

where w_e and w_ϵ are hyper-parameters to control the multi-task learning, and are chosen such that $w_e + w_\epsilon = 1$.

The described GNN architecture can then be trained for predicting scores of answer candidates and evidences, and used for inference on the whole input graphs in *one shot*.

5 ITERATIVE GRAPH NEURAL NETWORKS

We now outline how we use trained GNNs for iteratively reducing the graph at inference time. Specifically, we comment on the benefits that such iterative GNNs have for robustness, explainability and efficiency.

Drawbacks of a one-shot prediction. There are several drawbacks of predicting the answer from the full graph *at inference*:

(i) Directly predicting the answer from hundreds of answer candidates is non-trivial, and node classification may struggle to manifest fine-grained differences between such candidate answers in their encodings. This can negatively impact the *robustness* of the method on unseen data.

(ii) Further, if the answer is predicted from the whole input graph at inference time, this means that all nodes in the graph contribute towards the answer prediction. Showing the whole graph, consisting

of hundreds of nodes, to explain how the answer was derived is not practical. The SR-attention scores could be an indicator as to which nodes were more relevant, but attention is not always sufficient as an explanation [18]. Hence, *answer explainability* would be limited. (iii) Finally, obtaining cross-encodings for hundreds of nodes (entities and evidences) can be computationally expensive, affecting the *runtime efficiency* of the system. A large fraction of these initial graph nodes might be rather irrelevant, which can often be identified using a more light-weight (i.e. more efficient) encoder.

Iterative inference of GNNs. To overcome these drawbacks, we propose *iterative GNNs*: instead of predicting the answer in one shot, we *iteratively apply trained GNNs* of the outlined architecture *during inference*. The key idea is to shrink the graph after each iteration. This can be done via the evidence scores s_ϵ predicted by the GNNs, to identify the most relevant evidences. These evidences, and the connected entities, are used to initiate the graph given as input to the next iteration. In the final iteration, the answer is predicted from the reduced graph only.

Fig. 3 illustrates the training and inference with the iterative GNNs. Note that these GNNs are still trained on the full graphs in the training data, and are run iteratively only at inference time. This iterative procedure is feasible as the proposed GNN architecture is inherently independent of the input graph size: the same GNN trained on 500 evidences can be applied on a graph with 100, 20, or 5 evidences. The GNNs essentially learn to spread question-relevant information within local neighborhoods, which is not only required for large graphs with hundreds of evidences, but also for smaller graphs with a handful of nodes. Further, this iterative procedure is tractable only with the flexibility of scoring both entities and evidences in the graph, using the same GNN architecture.

Enhancing robustness. Within each iteration, the task complexity is decreased compared to the task complexity the original GNN was trained on. For example, the initial GNN was trained for predicting small answer sets and relevant evidences from several hundreds, but may only need to identify the top-100 evidences during inference. Thus, the current GNN has to be less discriminative at inference time than during training. This can help improve *robustness*.

Facilitating explainability. A primary benefit of the iterative mechanism is that the intermediate graphs can be used to better understand how answer prediction works via a GNN. Showing all the information contained in the original input graph with hundreds of nodes to the user is not practical: we can iteratively derive a small set of evidences (say five), from which the answer is predicted. These can be shown instead to the end user, enhancing *user explainability*.

Improving efficiency. To facilitate the runtime *efficiency* of the answering process, we refrain from encoding entities via cross-encodings in the *shrinking* (or *pruning*) iterations. Instead, we initiate the entity encodings, using a sum of the surrounding evidences, weighted by their question-relevance (i.e. their SR-attention):

$$e^0 = \sum_{\epsilon \in \mathcal{N}(e)} \alpha_{e,\epsilon} \cdot \epsilon^0 \quad (10)$$

where the SR-attention $\alpha_{e,\epsilon}$ is computed as in Eq. 1, employing a different linear projection. This can be perceived as obtaining *alternating encodings* of entities: the initial evidence encodings are used to initialize entity encodings (inspired by [2]). The message passing would then proceed as outlined in Sec. 4.3.

Instantiation. We train several one-shot GNNs of the architecture outlined above, using different weights on the answer prediction and evidence relevance prediction tasks (w_e and w_ϵ respectively) in the MTL setup. Further, we train GNNs using either cross-encodings or alternating encodings for entities. Training is conducted on the full input graphs present in the training set. We then simply instantiate all *pruning iterations* with the GNN that obtains the best evidence prediction performance on the graphs in the development (dev) set. Similarly, we use the trained GNN that obtained the best answering performance on the dev set to initiate the final *answering iteration*. Finally, the answer predicted by the system is given by:

$$a_{pred} = \arg\max_{e \in E} s_e \qquad (11)$$

This is shown to the end user, together with the explanatory evidences $\{\epsilon_{pred}\}$ (see outputs in Fig. 2), that are simply the set of evidences used in the answer prediction step.

6 EXPERIMENTS

6.1 Experimental setup

Dataset. We train and evaluate EXPLAIGNN on the ConvMix [8] benchmark, which was designed for ConvQA over heterogeneous information sources. The dataset has 16,000 questions (train: 8,400 questions, dev: 2,800 questions, test: 4,800 questions), within 3,000 conversations of five (2,800) or ten turns (200, only used for testing).
Metrics. For accessing the answer performance, we use *precision at 1* (**P@1**), as proposed in the ConvMix paper. To investigate the ranking capabilities of different methods in more detail, we also measure the *mean reciprocal rank* (**MRR**), and *hit at 5* (**Hit@5**). The *answer presence* (**Ans. pres.**) is the fraction of questions for which a gold answer is present in a given set of evidences.
Baselines. We compare EXPLAIGNN with the state-of-the-art method on the ConvMix dataset, CONVINSE [8]. CONVINSE leverages a Fusion-in-Decoder (FiD) [17] model for obtaining the top answer, which is designed to generate a single answer string. In [8], the ranked entity answers are then derived by collecting the top-k answer candidates with the highest surface-form match w.r.t. Levenshtein distance with the generated answer string. This procedure is somewhat limiting when measuring metrics beyond the first rank (i.e. MRR or Hit@5). Therefore, we enhanced the FiD model to directly generate top-k answer strings, and then consider the answer candidate with the highest surface-form match for each such generated string (**top-k FiD**), for fair comparison. We further compare with baselines proposed in [8] for question completion and resolution, and use the values reported in [8] for consistency.
Configurations. The QU and ER stages were initialized using the CONVINSE [8] code and data: we used Wikidata as the KB, and made use of the same version (2022-01-31) as earlier work. The Wikipedia evidences were taken from here[2], whenever applicable, and retrieved on-the-fly otherwise. This ensures that results are comparable with the results provided in earlier work.

The GNNs were implemented from scratch via PyTorch. We used DistilRoBERTa provided by Hugging Face[3] as encoder, which we found to perform slightly better than DistilBERT [45] (the distillation procedure is the same). DistilRoBERTa has an embedding

Table 1: Comparison of answering performance on the ConvMix [8] test set, using *gold answers* $\{a_{gold}\}$ in the history.

Method ↓	P@1	MRR	Hit@5
Q. Resolution [56] + BM25 + FiD [17]	0.282	0.289	0.297
Q. Rewriting [41] + BM25 + FiD [17]	0.271	0.278	0.285
CONVINSE [8] (original)	0.342	0.365	0.386
CONVINSE [8] (top-k FiD)	0.343	0.378	0.431
EXPLAIGNN (proposed)	0.406*	0.471*	0.561*

Table 2: Comparison of answering performance on the ConvMix test set, using *predicted answers* $\{a_{pred}\}$ in the history.

Method ↓	P@1	MRR	Hit@5
Q. Resolution [56]+ BM25 + FiD [17]	0.243	0.250	0.257
Q. Rewriting [41]+ BM25 + FiD [17]	0.221	0.227	0.235
CONVINSE [8] (original)	0.278	0.286	0.294
CONVINSE [8] (top-k FiD)	0.279	0.308	0.351
EXPLAIGNN (proposed)	0.339*	0.398*	0.477*

dimension of d=768. The GNNs were trained for 5 epochs. We chose an epoch-wise evaluation strategy, and kept the model that achieved the best performance on the dev set. We found three layer GNNs (L=3) most effective. AdamW was used as optimizer, using a learning rate of 10^{-5}, batch size of 1, and a weight decay of 0.01.

We also used the dev set for choosing the number of GNN iterations, and MTL weights for pruning and answering. The number of iterations was set to i=3. The GNN that maintains the highest answer presence among the top-5 evidences was chosen for instantiating the pruning iterations (alternating encodings for entities, w_e=0.3, w_ϵ=0.7), and the GNN obtaining the highest P@1 for the answer prediction (cross-encodings for entities, w_e=0.5, w_ϵ=0.5). In case there are more than 500 evidences, we retain only the top-500 obtained via BM25 scoring as input to the answering stage. A single GPU (NVIDIA Quadro RTX 8000, 48 GB GDDR6) was used to train and evaluate the models.

6.2 Key findings

This section will present the main experimental results on the ConvMix test set. All provided metrics are averaged over all *questions* in the dataset. The best method for each column is shown in **bold**. Statistical significance over the best baseline is indicated by an asterisk (*), and is measured via paired t-tests for MRR, and McNemar's test for binary variables (P@1 or Hit@5), with $p < 0.05$ in both cases.
EXPLAIGNN improves the answering performance. The main results in Table 1 demonstrate the performance benefits of EXPLAIGNN over the baselines. EXPLAIGNN significantly outperforms the best baseline on all metrics, illustrating the success of using *iterative graph neural networks* in ConvQA. As is clear from the method descriptions in Table 1, all baselines crucially rely on the generative reader model of FiD. FiD can ingest multiple evidences to produce the answer, but fails to capture their relationships explicitly, that is unique to our graph-based pipeline. Our adaptation of using top-k FiD instead of the default top-1 improved the ranking capabilities (MRR, Hit@5) of the strongest baseline CONVINSE. However, EXPLAIGNN still substantially improved over top-k FiD.
EXPLAIGNN is robust to wrong predictions in earlier turns. Unlike many existing works, we also evaluated the methods in a

Table 3: Effect of varying source combinations at inference time, on test set. EXPLAIGNN is still *trained* on all sources.

Method ↓	P@1	MRR	Hit@5	Ans. pres.
KB	0.363	0.427	0.511	0.617
Text	0.233	0.300	0.380	0.530
Tables	0.064	0.084	0.108	0.155
Infoboxes	0.256	0.302	0.362	0.409
KB+Text	0.399	0.464	0.549	0.672
KB+Tables	0.363	0.429	0.515	0.629
KB+Infoboxes	0.376	0.443	0.532	0.640
Text+Tables	0.235	0.305	0.392	0.540
Text+Infoboxes	0.309	0.369	0.445	0.572
Tables+Infoboxes	0.263	0.312	0.374	0.453
All sources	**0.406**	**0.471**	**0.561**	**0.683**

Table 4: Effect of varying the multi-task learning weights *when training* the one-shot GNN modules (on the dev set).

	P@1	MRR	Hit@5	Ans. pres.	HA runtime
EXPLAIGNN (i=1: 500→a_{pred}; **cross-encodings** for entities)					
w_e=1.0, w_ϵ=0.0	0.440	0.501	0.578	0.229	1, 018 ms
w_e=0.7, w_ϵ=0.3	0.439	0.499	0.573	0.573	1, 029 ms
w_e=0.5, w_ϵ=0.5	0.442	0.502	0.581	0.583	1, 017 ms
w_e=0.3, w_ϵ=0.7	0.431	0.495	0.572	0.586	1, 013 ms
w_e=0.0, w_ϵ=1.0	0.033	0.041	0.044	0.579	1, 008 ms
EXPLAIGNN (i=1: 500→a_{pred}; **alternating encodings** for entities)					
w_e=1.0, w_ϵ=0.0	0.417	0.485	0.573	0.508	443 ms
w_e=0.7, w_ϵ=0.3	0.410	0.470	0.545	0.568	444 ms
w_e=0.5, w_ϵ=0.5	0.404	0.472	0.555	0.569	447 ms
w_e=0.3, w_ϵ=0.7	0.405	0.472	0.552	0.589	442 ms
w_e=0.0, w_ϵ=1.0	0.117	0.169	0.221	0.581	449 ms

realistic scenario, in which the predicted answers a_{pred} are used as (noisy) input for the conversational history instead of the standard yet impractical choice of inserting gold answers from the benchmark. Results are shown in Table 2 (cf. Table 1 results, that show an evaluation with gold answers). While the performance of all methods drops in this more challenging setting, the trends are very similar, indicating that EXPLAIGNN can successfully overcome failures (i.e. incorrect answer predictions) in earlier turns. EXPLAIGNN outperforms all baselines significantly, including a P@1 jump from 0.279 for the strongest baseline to 0.339.

Heterogeneous sources improve performance. Analysis of source combinations is in Table 3. The first takeaway is that the answering performance was the best when the full spectrum of sources was used. EXPLAIGNN can make use of the enhanced answer presence in this case, and answer more questions correctly. Next, the results indicate that adding an information source is always beneficial: the performance of combinations of two sources is in all cases better than for the two sources individually.

6.3 Analysis

Multi-task learning enables flexibility. A systematic analysis of the effect of different multi-task learning weights is shown in Table 4. This analysis is conducted on the dev set, for choosing the best GNN for pruning and answering, respectively. Entities

are either encoded via cross-encodings, or alternating encodings (see Eq. 10). The results indicate the runtime benefits of using alternating encodings for entities. Further, when optimized for evidence relevance prediction (w_ϵ=0.7 or w_ϵ=1.0), it can maintain a high answer presence (measured within top-5 evidences), indicating that light-weight encoders are indeed sufficient for the pruning iterations. Further, we found putting equal weights on the answer and evidence relevance prediction to be beneficial for answering.

Iterative GNNs do not compromise runtimes. Table 5 reports results of varying the number of iterations $i \in \{1, 2, 3, 4\}$, and the graph size in the number of evidences the answer is predicted from ($|\mathcal{E}| \in \{500, 100, 50, 20, 5\}$). For each row, the reduction in graph size in terms of the number of evidences considered was kept roughly constant for consistency. By our smart use of alternating encodings of entities in the pruning iterations, *runtimes remain immune* to the number of pruning iterations (times for i=4 are not necessarily higher than those for i=3, and so on). Rather, the runtime is primarily influenced by the size of the graph (no. of evidences) given to the final answer prediction step (compare runtimes within each iteration group). Recall that this graph size for answer prediction can impact *explainability* to end users, if it is not small enough. Notably, *performance* remains reasonably stable in most cases. A key takeaway from these results is that the trained GNN models generalize well to graphs of different sizes. Concretely, while all of these models are trained on graphs established from 500 evidences, they can be applied to score nodes in graphs of variable sizes.

EXPLAIGNN can be applied out-of-the-box. For testing the generalizability of EXPLAIGNN, we applied the pipeline trained on the CONVMIX dataset out-of-the-box, without any training or fine-tuning, on the CONVQUESTIONS [7] dataset. CONVQUESTIONS is a competitive benchmark for ConvQA methods operating over KBs. We test the (same) EXPLAIGNN pipeline in two different modes: (i) using only facts from the KB, and (ii) using evidences from all information sources. Table 6 shows the results ([†] and [‡] indicate statistical significance over the leaderboard toppers KRR [22] and PRALINE [20], respectively). In the KB-only setting, EXPLAIGNN obtains state-of-the-art performance, reaching the highest MRR score. Also, we found that integrating heterogeneous sources can improve the answer performance substantially, even though CONVQUESTIONS was collected primarily for KBs.

Iterative GNNs improve robustness. In Sec. 4, we argued that iterative GNNs enhance the pipeline's robustness over a single GNN applied on the full graph in one shot. While the performance of the one-shot GNN on the CONVMIX dev set is comparable (Table 5), we found that it cannot generalize as well to a different dataset: when applied on CONVQUESTIONS, the respective performance is significantly lower than for EXPLAIGNN, in both KB-only (P@1: 0.330 vs. 0.281) and heterogeneous (P@1: 0.363 vs. 0.318) settings.

SR-attention, cross-encoding and entity types are crucial. Table 7 shows results ([†] indicates significant performance drop) of our ablation study. We found that each of these mechanisms helps the pipeline to improve QA performance. The most decisive factor is the SR-attention (Sec. 4.3), which ensures that only the question-relevant information is spread within the local neighborhoods: without this component, the performance drops substantially (P@1= 0.442 to 0.062). Similarly, the cross-encodings (Sec. 4.2)

Table 5: Varying the no. of iterations, pruning factors, and evidences the answer is predicted from *during inference* (dev set).

Method ↓	P@1	MRR	Hit@5	Ans. pres. and no. of evidences after pruning iteration			HA runtime
EXPLAIGNN (i=1: 500→a_{pred})	**0.442**	0.502	0.581	–	–	–	1,017 ms
EXPLAIGNN (i=2: 500→100→a_{pred})	0.441	0.504	0.587	–	–	0.687 | 100	744 ms
EXPLAIGNN (i=2: 500→50→a_{pred})	0.440	0.504	0.588	–	–	0.675 | 50	591 ms
EXPLAIGNN (i=2: 500→20→a_{pred})	0.438	0.504	**0.591**	–	–	0.655 | 20	515 ms
EXPLAIGNN (i=2: 500→5→a_{pred})	0.422	0.480	0.560	–	–	0.589 | 5	459 ms
EXPLAIGNN (i=3: 500→200→100→a_{pred})	0.441	0.504	0.585	–	0.694 | 200	0.685 | 100	995 ms
EXPLAIGNN (i=3: 500→150→50→a_{pred})	0.441	**0.505**	0.586	–	0.693 | 150	0.678 | 50	741 ms
EXPLAIGNN (i=3: 500→100→20→a_{pred}; **proposed**)	**0.442**	**0.505**	0.589	–	0.687 | 100	0.654 | 20	601 ms
EXPLAIGNN (i=3: 500→50→5→a_{pred})	0.419	0.475	0.556	–	0.675 | 50	0.579 | 5	511 ms
EXPLAIGNN (i=4: 500→300→150→100→a_{pred})	0.441	0.504	0.587	0.696 | 300	0.691 | 150	0.686 | 100	1,232 ms
EXPLAIGNN (i=4: 500→200→100→50→a_{pred})	0.440	0.504	0.585	0.694 | 200	0.685 | 100	0.677 | 50	945 ms
EXPLAIGNN (i=4: 500→200→50→20→a_{pred})	0.436	0.500	0.584	0.694 | 200	0.677 | 50	0.652 | 20	769 ms
EXPLAIGNN (i=4: 500→100→20→5→a_{pred})	0.422	0.476	0.553	0.687 | 100	0.654 | 20	0.575 | 5	577 ms

Table 6: Out-of-the-box EXPLAIGNN, without further training or fine-tuning, on the CONVQUESTIONS [7] benchmark.

Method ↓	P@1	MRR	Hit@5
CONVEX [7]	0.184	0.200	0.219
FOCAL ENTITY [24]	0.248	0.248	0.248
OAT [29]	0.166	0.175	–
OAT [29] (gold seed entity)	0.250	0.260	–
CONQUER [21]	0.240	0.279	0.329
PRALINE [20]	0.294	0.373	0.464
KRR [22] (gold seed entity)	**0.397**	0.397	0.397
EXPLAIGNN (KB-only)	0.330[‡]	**0.399**[‡]	**0.480**[†‡]
EXPLAIGNN	0.363[‡]	0.447[†‡]	0.546[†‡]

Table 7: Ablation study of the EXPLAIGNN pipeline.

Method ↓	P@1	MRR	Hit@5
EXPLAIGNN	**0.442**	**0.505**	**0.589**
w/o SR-attention	0.062[†]	0.137[†]	0.178[†]
w/o cross-encoder	0.352[†]	0.431[†]	0.529[†]
w/o entity type	0.420[†]	0.492[†]	0.584
w/o hallucination prevention	0.436	0.502	0.588
w/o use of SR slots in retrieval	0.427[†]	0.493[†]	0.578[†]

initialize the nodes with question-relevant encodings. Also notable is the crucial role of entity types (Sec. 4.2), that help identify irrelevant answer candidates with mismatched answer types.

Error analysis. We identified three key sources of error: (i) the answer is not present in the initial graph (53.9% of error cases), which can be mitigated by improving the QU and ER stages of the pipeline, (ii) the answer is dropped when shrinking the graph (8.1%), and (iii) the answer is present in the final graph but not identified as the correct answer (38.0%). The graph shrinking procedure is responsible for only a few errors (2.2% in the first iteration, 5.9% in second), demonstrating the viability of our iterative approach.

7 EXPLAINABILITY

We evaluate the explainability of our pipeline by a user study on Amazon Mechanical Turk (AMT). The typical use case scrutinized is that a user has an information need, obtains the answer predicted

Figure 4: A representative example from our user study, illustrating a typical output of EXPLAIGNN.

by the system, and is unsure whether to trust the provided answer. Thus, the key objective of the explanations here is to help the user decide whether to trust the answer. If the user is able to make this decision diligently, the explanations serve their purpose.

User study design. For a predicted answer to a conversational question, the user is given the conversation history, the current question, the SR, and the explaining evidences. An example of the input presented to the user is shown in Fig. 4 (we used five explanatory evidences). The user then has to decide whether the provided answer is correct. We randomly sample 1200 instances on which we measure user accuracy. One of our main considerations during sampling was that one half (600) was correctly and the other half incorrectly answered by EXPLAIGNN.

Table 8: Confusion matrix for the user study, showing the probabilities of user correctness against user certainty.

	User correct	User incorrect	
User certain	0.632	0.166	0.798
User uncertain	0.129	0.073	0.202
	0.761	0.239	

User study interface. For each instance, we then ask Turkers the following questions: (i) *"Do you think that the provided answer is correct?"* (**User correctness**), (ii) *"Are you certain about your decision?"* (**User certainty**), and (iii) *"Why are you certain about the decision?"* or *"Why are you uncertain about the decision?"*. The first two questions can be answered by either *"Yes"*, or *"No"*. Depending on the answer to the second question, the third question asks for reasons for their (un)certainty. The user can select multiple provided options. If the user is certain, these options are either *good explanation*, *prior knowledge*, and *common sense*. If the user is uncertain, then she must choose between *bad explanation* or *question/conversation unclear*. The idea is to remove confounding cases in which users make a decision regardless of the provided explanation, since we cannot infer much from these. Note that Turkers were not allowed to access external sources like web search.

Quality control. For quality control, we restrict the participation in our user study to Master Turkers with an approval rate of ≥95%. Further, we added honeypot questions, for which the answer and provided information are clearly irrelevant w.r.t. the question (domain and answer type mismatch), We excluded submissions from workers who gave incorrect responses to the honeypots.

Explaignn provides explainable answers. Findings are presented as a confusion matrix in Table 8 (values computed after removing confounding assessments). In the 771 of the 1200 observations that remain, we found that the user can accurately decide the correctness of the system answer (76.1%) and is certain about their assessment (79.8%) most of the time. This proves that our explanations are indeed useful to end users. If the user is certain about their assessment, then we observed that the accuracy in deciding the correctness was higher: P(User correct | User certain) = 0.792.

8 RELATED WORK

Conversational question answering. There has been extensive research on ConvQA [42, 44] in recent years, which can largely be divided into methods using a KB [22, 24, 29], methods using a text corpus [35–37], and methods integrating tables [16, 31].

In **ConvQA over KBs**, the (potentially completed) question is often mapped to logical forms that are run over the KB to obtain the answer [13, 22, 29, 34, 48]. A different type of approach is to search the answer in the local neighborhoods of an ongoing context graph, which captures the relevant entities of the conversation [7, 21, 24]. Early **ConvQA systems over textual sources** assumed the relevant information (i.e. text passage or document) to be given [5, 15, 42], and modeled the problem as a machine reading comprehension (MRC) task [40]. This assumption was challenged in [36], which proposed ORConvQA including a retrieval stage. Recent works follow similar ideas, and mostly rely on *question rewriting* [41, 54] or *question resolution* [56], and then employ a MRC model. In related work on **ConvQA over tables**, the answer

is either derived via logical forms [16], or via pointer-networks operating on graph-encodings of the tables [31].

All of these methods rely on a single information source for answering questions, inherently limiting their answer coverage. Recently, there has been preliminary work on ConvQA using a mixture of the sources outlined above [8, 9]. The method proposed in [9] appends incoming questions to the conversational history, and then generates a program code to derive the answer from a table using a sequence-to-sequence model. In Convinse [8], evidences from heterogeneous sources are concatenated and fed into a sequence-to-sequence model to generate the answer. Both methods heavily rely on sequence-to-sequence models where the generated outputs are not explainable, and may even be hallucinated.

QA over heterogeneous sources. In addition to work on ConvQA over heterogeneous sources, there is a long line of work on answering *one-off questions* using such mixtures of sources [3, 11, 46, 50, 51, 57, 58]. More recently, UniK-QA [33] proposed the verbalization of evidences, and then applied FiD [17] for the answering task. UDT-QA [28] improved over UniK-QA by implementing more sophisticated mechanisms for evidence verbalization, and use T5 [39] to generate the answer. Similarly, Shen et al. [49] propose a dataset and method for answering questions on products from heterogeneous sources, leveraging BART [25]. These approaches, being sequence-to-sequence models at their core, face similar problems as mentioned before. HeteroQA [12] explore heterogeneity in the context of community QA where retrieval sources could be posts, comments or even other questions and others.

Explainable QA. Existing work is mostly on single-turn methods operating over on a single source, with template [1, 47] and graph-based derivation sequences [19, 27] as mechanisms for ensuring explainability. Works on text-QA provide end users with actual passages and reasoning paths used for answering [32, 59]. Post-hoc explainability for QA over KBs and text is investigated in [52]. These methods cannot be easily adapted to a conversational setting with incomplete follow-up questions.

Explainability in GNNs. Note that explainability for GNNs is an active field of research [62], devising general techniques to identify important graph node features, or provide model-level explanations. Such approaches are mostly designed for developers, and therefore hardly applicable in our scenario. Through our iterative model, we propose a QA-specific method for deriving explanations of GNN predictions that can be understood by average web users.

9 CONCLUSION

There are three main takeaways for a broader audience from this work. First, at a time when large language models (LLMs) like Chat-GPT are used as a one-stop shop for most NLP tasks including ConvQA, our method Explaignn stands out by providing traceable provenance of its answer predictions. Next, explainability for graph neural networks is an unsolved concern: we propose an iterative model that sequentially reduces the graph size as a medium for offering causal insights into the prediction process. Finally, for several systems in IR and NLP, performance, efficiency, and explainability are seen as trade-offs. Through our highly configurable solution, we show that in certain use cases, it is actually possible to find configurations that lie at the sweet spots of all the three factors.

REFERENCES

[1] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. QUINT: Interpretable Question Answering over Knowledge Bases. In *EMNLP*.

[2] Shubham Chatterjee and Laura Dietz. 2022. BERT-ER: Query-specific BERT Entity Representations for Entity Ranking. In *SIGIR*.

[3] Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2021. Open Question Answering over Tables and Text. In *ICLR*.

[4] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension. In *IJCAI*.

[5] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *EMNLP*.

[6] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2022. Beyond NED: Fast and Effective Search Space Reduction for Complex Question Answering over Knowledge Bases. In *WSDM*.

[7] Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *CIKM*.

[8] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2022. Conversational Question Answering on Heterogeneous Sources. In *SIGIR*.

[9] Yang Deng, Wenqiang Lei, Wenxuan Zhang, Wai Lam, and Tat-Seng Chua. 2022. PACIFIC: Towards Proactive Conversational Question Answering over Tabular and Textual Data in Finance. In *EMNLP*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[11] David A. Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development* (2012).

[12] Shen Gao, Yuchi Zhang, Yongliang Wang, Yang Dong, Xiuying Chen, Dongyan Zhao, and Rui Yan. 2022. HeteroQA: Learning towards question-and-answering through multiple information sources via heterogeneous graph modeling. In *WSDM*.

[13] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *NeurIPS*.

[14] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

[15] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2018. FlowQA: Grasping Flow in History for Conversational Machine Comprehension. In *ICLR*.

[16] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

[17] Gautier Izacard and Édouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *EACL*.

[18] Sarthak Jain and Byron C Wallace. 2019. Attention is not Explanation. In *NAACL-HLT*.

[19] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex Temporal Question Answering on Knowledge Graphs. In *CIKM*.

[20] Endri Kacupaj, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. 2022. Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs. In *CIKM*.

[21] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Reinforcement Learning from Reformulations in Conversational Question Answering over Knowledge Graphs. In *SIGIR*.

[22] Xirui Ke, Jing Zhang, Xin Lv, Yiqi Xu, Shulin Cao, Cuiping Li, Hong Chen, and Juanzi Li. 2022. Knowledge-augmented Self-training of A Question Rewriter for Conversational Knowledge Base Question Answering. In *EMNLP*.

[23] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang-Dong Yoo. 2019. Gaining Extra Supervision via Multi-task learning for Multi-Modal Video Question Answering. In *IJCNN*.

[24] Yunshi Lan and Jing Jiang. 2021. Modeling Transitions of Focal Entities for Conversational Knowledge Base Question Answering. In *ACL*.

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.

[26] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient One-Pass End-to-End Entity Linking for Questions. In *EMNLP*.

[27] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. 2019. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *SIGIR*.

[28] Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open Domain Question Answering with A Unified Knowledge Interface. In *ACL*.

[29] Pierre Marion, Paweł Krzysztof Nowak, and Francesco Piccinno. 2021. Structured Context and High-Coverage Grammar for Conversational Question Answering

[30] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. In *ACL*.

[31] Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering Conversational Questions on Structured Data without Logical Forms. In *EMNLP-IJCNLP*.

[32] Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction. In *ACL*.

[33] Barlas Oğuz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified Representations of Structured and Unstructured Knowledge for Open-Domain Question Answering. In *NAACL-HLT*.

[34] Laura Perez-Beltrachini, Parag Jain, Emilio Monti, and Mirella Lapata. 2023. Semantic Parsing for Conversational Question Answering over Knowledge Graphs. In *EACL*.

[35] Minghui Qiu, Xinjing Huang, Cen Chen, Feng Ji, Chen Qu, Wei Wei, Jun Huang, and Yin Zhang. 2021. Reinforced History Backtracking for Conversational Question Answering. In *AAAI*.

[36] Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. In *SIGIR*.

[37] Chen Qu, Liu Yang, Minghui Qiu, W Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019. BERT with history answer embedding for conversational question answering. In *SIGIR*.

[38] Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. 2019. Attentive history selection for conversational question answering. In *CIKM*.

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* (2020).

[40] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

[41] Gonçalo Raposo, Rui Ribeiro, Bruno Martins, and Luísa Coheur. 2022. Question rewriting? Assessing its importance for conversational question answering. In *ECIR*.

[42] Siva Reddy, Danqi Chen, and Christopher Manning. 2019. CoQA: A conversational question answering challenge. *TACL* (2019).

[43] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

[44] Rishiraj Saha Roy and Avishek Anand. 2022. *Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections*. Springer.

[45] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *EMC2@NeurIPS*.

[46] Denis Savenkov and Eugene Agichtein. 2016. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *SIGIR*.

[47] Saeedeh Shekarpour, Abhishek Nadgeri, and Kuldeep Singh. 2020. QA2Explanation: Generating and Evaluating Explanations for Question Answering Systems over Knowledge Graph. In *IntEx-SemPar@EMNLP*.

[48] Tao Shen, Xiubo Geng, QIN Tao, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base. In *EMNLP-IJCNLP*.

[49] Xiaoyu Shen, Gianni Barlacchi, Marco Del Tredici, Weiwei Cheng, Bill Byrne, and Adrià de Gispert. 2022. Product Answer Generation from Heterogeneous Sources: A New Benchmark and Best Practices. In *ECNLP@ACL*.

[50] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text. In *EMNLP-IJCNLP*.

[51] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *EMNLP*.

[52] Alona Sydorova, Nina Poerner, and Benjamin Roth. 2019. Interpretable Question Answering on Knowledge Bases and Text. In *ACL*.

[53] Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message Passing for Complex Question Answering over Knowledge Graphs. In *CIKM*.

[54] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question rewriting for conversational question answering. In *WSDM*.

[55] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. REL: An entity linker standing on the shoulders of giants. In *SIGIR*.

[56] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas, and Maarten de Rijke. 2020. Query resolution for conversational search with limited supervision. In *SIGIR*.

[57] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *COLING*.

[58] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *ACL*.

[59] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*.

[60] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge

Graphs for Question Answering. In *NAACL*.

[61] Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. KG-FiD: Infusing Knowledge Graph in Fusion-in-Decoder for Open-Domain Question Answering. In *ACL*.

[62] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *PAMI* (2022).