
Question Answering over Heterogeneous Sources

A dissertation submitted towards the degree

Doctor of Engineering (Dr.-Ing.)

of the Faculty of Mathematics and Computer Science
of Saarland University

by **Philipp Christmann**



Saarbrücken, 2025



Date of the Colloquium: 06.02.2026
Dean of the Faculty: Prof. Dr. Roland Speicher
Chair of the Committee: Prof. Dr. Vera Demberg
Reviewers: Prof. Dr. Gerhard Weikum
Dr. Rishiraj Saha Roy
Prof. Dr. Gerard de Melo
Dr. Johanne Trippas
Academic Assistant: Dr. Soumi Das



Declaration of Original Authorship

I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

gez. Philipp Christmann

Saarbrücken, 12. August 2025

Abstract

Question answering (QA) systems provide crisp answers to questions posed by end users. Most existing QA systems cover solely one information source for answering: either a curated knowledge base (KB), or a text corpus, or a set of web tables. This inherently limits their answer coverage as individual sources are incomplete in capturing the wide variety of information targeted by users. Further, these prior systems are typically not designed for:

- (i) *Traceable answers* which help end users to investigate the answer correctness.
- (ii) *Cost-efficient inference*, instead relying on billion-scale generative language models.
- (iii) *Complex information needs* which can only be understood in an ongoing conversational context, require modeling of temporal constraints, or involve advanced operations such as aggregations, grouping and joins.

This dissertation addresses these issues and makes the following salient contributions:

- (i) Proposing a general 3-stage architecture for answering questions over *heterogeneous sources*, including a question understanding stage to model the information need, a stage for evidence retrieval from a set of heterogeneous sources, and an answering stage that treats this evidence in a unified manner for deriving a traceable answer.
- (ii) Devising *question understanding mechanisms* for conversational questions with incomplete intent, temporal questions with implicit or explicit time constraints, and complex questions that involve aggregation, grouping and joining of information from different sources.
- (iii) Developing end-to-end QA systems for conversational QA (CONVINSE and EXPLAIGNN), temporal QA (FAITH), complex QA (QUASAR) and QA over personal data (REQAP), that *integrate heterogeneous sources* for enhanced answer coverage.
- (iv) Constructing *large-scale benchmarks* for conversational QA (CONVMIX), temporal QA (TIQ), complex QA (COMPMIX), and QA over personal data (PERQA), with realistic questions that *target heterogeneous sources*.

Answers by our methodologies can be *traced back to the underlying evidence* by design. Also, computational efficiency is a primary design point – our methods build upon *small-scale models* that are trained via smart use of distant supervision and in-context learning.

Kurzfassung

Frage-Antwort-Systeme adressieren Informationsbedürfnisse von Endnutzern durch kurze und prägnante Antworten. Die meisten Systeme verwenden hierzu lediglich Informationen aus einer Quelle: entweder aus einem Wissensgraphen oder einem Textkorpus oder einer Ansammlung an Tabellen. Dies schränkt die Antwort-Verfügbarkeit stark ein, da die individuellen Quellen nicht die komplette Breite an relevanten Informationen bereitstellen. Zusätzliche Einschränkungen existierender Systeme sind:

- (i) Der Prozess zur Beantwortung von Fragen ist oft *nicht nachvollziehbar* für Endnutzer.
- (ii) Die Verwendung von Sprachmodellen mit Milliarden Parametern führt zu *hohen Inferenz-Kosten*.
- (iii) Die *Nicht-Berücksichtigung von Fragen*, welche (a) die Einbindung des Kontextes einer Konversation, (b) das Modellieren temporaler Bedingungen, oder (c) die Verwendung von Operatoren zur Verknüpfung, Gruppierung oder Aggregation von Informationen erfordern.

Diese Dissertation befasst sich mit den zuvor genannten Schwachstellen und präsentiert die folgenden Forschungsbeiträge:

- (i) Wir stellen eine generelle System-Architektur zur Frage-Beantwortung unter Berücksichtigung von *heterogenen Quellen* vor, die drei Phasen beinhaltet: (a) eine Phase zum Frage-Verständnis, (b) eine Phase zum Abruf relevanter Informationen aus heterogenen Quellen, und (c) eine Phase zur Generierung einer nachvollziehbaren Antwort, welche Informationen aus unterschiedlichen Quellen vereinheitlicht.
- (ii) Wir präsentieren *Mechanismen zum Frage-Verständnis* von (a) unvollständig formulierten Informationsbedürfnissen innerhalb einer Konversation, (b) Fragen mit impliziten oder expliziten temporalen Bedingungen, sowie (c) komplexen Fragen unter Verwendung von Operatoren zur Verknüpfung, Gruppierung, oder Aggregation von Informationen aus verschiedenen Quellen.
- (iii) Wir entwickeln vollständige Frage-Antwort-Systeme zur Beantwortung von Fragen innerhalb einer Konversation (CONVINSE und EXPLAINNN), Fragen mit temporalen Bedingungen (FAITH), komplexen Fragen (QUASAR), und Fragen zu persönlichen Daten (REQAP), basierend auf *Informationen aus heterogenen Quellen*.
- (iv) Wir erstellen umfangreiche Datensätze mit Fragen innerhalb von Konversationen (CONVMIX), Fragen mit temporalen Bedingungen (TIQ), Fragen komplexer Natur (COMPMIX), und Fragen zu persönlichen Daten (PERQA), die Wissen aus *heterogenen Quellen erfordern*.

Unsere Methoden sind so konzipiert, dass Antworten nachvollziehbar sind und zu den ursprünglichen Quellen zurückverfolgt werden können. Außerdem ist die Recheneffizienz ein wesentlicher Entwurfsaspekt – unsere Methoden ermöglichen die Verwendung von kompakten neuronalen Modellen. Diese Modelle werden mithilfe von Techniken für *distant supervision* (automatische Annotation von Trainingsdaten) oder *in-context learning* (kontextbasiertes Lernen) trainiert.

Acknowledgements

I am grateful to my supervisors, Gerhard Weikum and Rishiraj Saha Roy, for their guidance throughout the years of my Bachelor's, Master's and PhD thesis.

Gerhard, thank you for your trust and for offering me a PhD position under your supervision. It has been a privilege to work so closely with you, with almost daily discussions towards the end. I sincerely value the knowledge and insights I gained through all these interactions.

Rishi, thank you for your invaluable mentorship, knowledge and all of our discussions. You provided me with all the necessary skills for composing this dissertation and greatly shaped my development as a researcher.

I would like to thank Zhen for our fruitful collaboration, and Svitlana, Adrià, Ionut, Rexhina, and Bill for the great internship experience.

Many thanks to Johanne and Gerard for taking the time to review this dissertation.

I am also grateful to my friends at D5 for insightful discussions, amazing conference experiences, and countless “kicker” breaks.

Finally, I would like to thank my friends and family, especially my parents, Rita and Roger, and my partner, Laura. Thank you for always being there for me!

Contents

1	Introduction	1
1.1	Motivation	1
1.2	State of the Art and its Limitations	3
1.3	Desiderata	3
1.4	Contributions	4
1.5	Publications	5
1.6	Outline	7
2	Background	9
2.1	Concepts and Notation	9
2.2	Related Work	14
3	Conversational Question Answering	19
3.1	Introduction	20
3.2	The CONVINSE Methodology	22
3.3	The CONVMIX Benchmark	27
3.4	Experimental Setup	29
3.5	Experimental Results and Insights	31
3.6	Demonstration	36
3.7	Related Work	36
3.8	Conclusion	38
4	Explainable Question Answering	39
4.1	Introduction	40
4.2	The EXPLAIGNN Methodology	41
4.3	Experimental Setup	48
4.4	Experimental Results and Insights	49
4.5	User Study on Explainability	55
4.6	Demonstration	58
4.7	Related Work	58
4.8	Conclusion	60
5	Temporal Question Answering	61
5.1	Introduction	62
5.2	Concepts and Notation	64
5.3	The FAITH Methodology	64
5.4	The TIQ Benchmark	69

Contents

5.5	Experimental Setup	73
5.6	Experimental Results and Insights	75
5.7	Related Work	81
5.8	Conclusion	82
6	Complex Question Answering	85
6.1	Introduction	86
6.2	The QUASAR Methodology	87
6.3	The COMPMIX Benchmark	90
6.4	Experimental Setup	94
6.5	Experimental Results and Insights	96
6.6	Related Work	100
6.7	Discussion	101
7	Personal Question Answering	103
7.1	Introduction	105
7.2	Data Model	107
7.3	The REQAP Methodology	107
7.4	The PERQA Benchmark	113
7.5	Experimental Setup	116
7.6	Experimental Results and Insights	119
7.7	User Study for Evaluation	123
7.8	Demonstration	125
7.9	Related Work	128
7.10	Conclusion	129
8	Conclusion	133
8.1	Summary	133
8.2	Lessons Learned	135
8.3	Limitations	138
8.4	Outlook	139
	List of Resources	143
	List of Figures	145
	List of Tables	147
	Bibliography	149

1

Introduction

Contents

1.1	Motivation	1
1.2	State of the Art and its Limitations	3
1.3	Desiderata	3
1.4	Contributions	4
1.5	Publications	5
1.6	Outline	7

1.1 Motivation

There has been extensive research on *question answering* (QA) with the goal of automatically providing crisp and direct answers to questions posed in natural language by lay users [175]. This work can largely be divided into:

- (i) methods that use a curated knowledge base (KB) as answer source (KB-QA) (e.g., [1, 8]),
- (ii) methods that retrieve information from a text corpus (Text-QA) (e.g., [85, 256]), and
- (iii) methods that tap into a collection of tables (Table-QA) (e.g., [76, 88]).

Typically, the QA system solely taps into a single type of information source: either a KB *or* a text corpus *or* a collection of tables. This greatly limits the *answer coverage* of the QA system, as these individual sources are inherently *incomplete*, and may not cover the knowledge required for answering a specific user question.

Example. Consider the questions in the following example conversation:

Conversation C¹

*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

1 Introduction

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 : *is Deception Point longer?*

a^6 : No

Questions q^1 and q^3 can be answered conveniently using a KB like Wikidata [220], YAGO [195], or DBpedia [3], that store factual world knowledge in compact RDF triples. However, questions q^2 , q^4 or q^5 are not easily answerable by such KBs. Question q^2 entails understanding the significance of a character in a story, which is typically not captured in KBs. Similarly, the knowledge required for answering question q^4 is found in the novel itself or related summaries and discussions, but not in structured KBs or tables. Numerical information, as necessary for targeting questions q^5 and q^6 , is typically captured in tables only.

Further, even single questions often require access to information from multiple information sources. Consider the *follow-up questions* q^7 and q^8 :

...

q^7 : *New Langdon books after the iPhone 4 release?*

a^7 : *Inferno* (2013), *Origin* (2017), *The Secret of Secrets* (2025)

q^8 : *How much money have I spent on his books?*

a^8 : 83.66 EUR

For q^7 , while the release date of the iPhone 4 is stored in structured form in KBs, the bibliography of Dan Brown, including all releases of the Robert Langdon series, is provided on a Wikipedia page. Identifying this subset of Dan Brown books from a KB would require non-trivial reasoning, even if the necessary information was present in the KB (which is not the case).

Finally, question q^8 is of a slightly different nature, asking about personal data of the specific user. Again, the relevant information for answering can be distributed among different sources of personal data. For example, information about the user's book purchases can be found in structured data from an Amazon Marketplace export, in structured data from Apple books, and in mails from online retail providers with unstructured content.

Challenges. Ideally, the QA system should be able to tap into all these kinds of information repositories, to improve its answer recall, and to boost answer confidence by leveraging multiple sources of evidence. Orthogonal to the necessity of incorporating such *heterogeneous* sources, the example conversation poses further challenges for QA systems:

- (i) The individual questions in this conversation are *incomplete*, and build upon the ongoing context. None of the questions, except for q^1 (and potentially q^7), are understandable when considered in isolation. This setting is referred to as *conversational QA* (ConvQA).
- (ii) Temporal questions, such as q^7 , can express time constraints explicitly or, in this case, even *implicitly*. For answering such questions *faithfully*, these temporal constraints need to be modeled explicitly by the QA system.
- (iii) Answering more complex questions (e.g., q^6 and q^8) involving comparisons, aggregations or joins, entails fine-grained interpretation of the user's information need. Also, information from unstructured text (as in mails) may first need to be transformed into structured form, to enable such complex operations over large-scale input data.

The goal of this dissertation is to seamlessly integrate heterogeneous sources for answering (i) conversational questions with implicit context, (ii) temporal questions with time constraints, and (iii) questions involving complex operations that can be of analytical nature.

1.2 State of the Art and its Limitations

QA over mixed sources. There has been a long line of research on QA over a single type of information source, namely over KBs [1, 8, 11], text corpora [23, 85, 256], or table collections [20, 76, 88]. Being tied to single sources only, these methods yield limited answer coverage.

As a result, QA over combinations of sources evolved as a promising direction [24, 196, 197]. Recently, *verbalization* emerged as the prevalent approach for QA over heterogeneous sources, linearizing information from structured sources into textual form [154]. For example, each table-row and each fact in the KB is transformed into a text sentence. Unstructured text is split into snippets of manageable size (typically sentence-level). This evidence is then uniformly processed using natural language processing (NLP) techniques.

These works are *limited to simple questions* and unable to deal with incomplete follow-up questions, questions with temporal constraints, or questions with other complexities. Specifically, these methods cannot cope with the challenges posed by questions in our running example.

Parametric knowledge. With the emergence of *generative language models* (GLMs) [143], QA systems are often built by relying solely on the neural parameters of the GLM, the so-called *parametric knowledge*. These GLM parameters are optimized for generating web text, including factual knowledge, during their *pre-training* stage, and their *instruction tuning* stage [155] enables them to provide fluent responses to user questions. However, GLMs *do not provide traceable provenance*, and *fall short on long-tail knowledge*, as such information is scarcely present in their training data.

1.3 Desiderata

The driving theme of this dissertation is to enable QA systems to seamlessly *integrate heterogeneous sources* for answering questions. Besides this primary objective, there are three other desiderata that are considered in the context of this dissertation:

- *Precise question understanding*
- *Answer traceability*
- *Computational efficiency*

Integrating heterogeneous sources. By integrating the full range of structured and unstructured information sources, we aim to *enhance the answer coverage* (e.g., for being able to answer all of $q^1 - q^8$). At the same time, since the relevant information can be expressed in multiple sources simultaneously, tapping into all heterogeneous sources (KB, text *and* tables) can *boost the answer accuracy* (e.g., q^1 and q^3). Finally, *more complex information needs* often require access to evidence from multiple knowledge repositories, with subsequent aggregation of this information, to answer a single information need (e.g., q^7 and q^8).

Precise question understanding. User questions are often incomplete, implicitly referring to context introduced in an ongoing conversation. Such *conversational questions* require precise understanding of the user’s intent by the QA system. Notably, the QA system’s representation of the information need has to be agnostic to the specific information repository used for answering, and support matching against heterogeneous sources. For example, KB subgraphs, as utilized by previous work on conversational QA [32], are not a suitable representation in this case, being limited to the KB only.

Orthogonally, processing complex questions, with temporal constraints or aggregations, requires fine-grained understanding of the user’s intent. Ideally, the corresponding representation

goes beyond the question itself, towards an actual *answering plan* while still remaining flexible regarding the information sources.

Answer traceability. By design, QA systems are used when the correct answer is unknown to the user. Hence, providing only the answer would typically prevent the user from verifying the *answer correctness*. For a more trustworthy user experience, the QA system should outline the steps towards deriving the answer – to enable end users in making sense of *answer traces* with *supporting evidence*.

Computational efficiency. Finally, we aim to avoid using billion-scale language models at inference time, whenever possible, to *reduce the computational cost and energy consumption*. We aim to deploy the QA system on hardware with moderate computational resources, ideally directly on end user devices.

1.4 Contributions

This dissertation makes the following salient contributions:

- We propose CONVINSE [33], the first method for ConvQA that integrates heterogeneous sources to enhance the answer coverage for this task. CONVINSE derives an intent-explicit and source-agnostic *structured representation* from the incomplete question, *verbalizes* all relevant information and generates the answer autoregressively. For experiments, we construct the CONVMIX dataset, the first benchmark for ConvQA over heterogeneous sources, with 14K conversations obtained via crowdsourcing.
- CONVINSE and previous works on heterogeneous QA [154] retrieve top- k (typically 100) evidence snippets, and then generate the answer, without considering the relationships between these information pieces. We develop EXPLAIGNN [34] which constructs an *evidence graph* based on shared entities of evidence snippets, seamlessly integrating information from heterogeneous sources into a graph that models the interplay of evidence. This graph is then efficiently reduced in size via *iterative inference of graph neural networks* (GNNs).
- Questions with time constraints can be answered by general-purpose QA systems, but often with the caveat that the temporal conditions are neglected or even entirely disregarded. We propose FAITH [91], a QA system that transforms implicit temporal constraints into explicit conditions. These conditions are used for pruning out evidence that violates the time constraints. FAITH is the first temporal QA system that taps into heterogeneous information sources, and provides faithful answers with explanatory evidence. For experiments, we construct TIQ, a dataset with 10K questions that express implicit temporal constraints, based on heterogeneous sources.
- For answering complex questions, we present QUASAR [37], a flexible framework for QA over heterogeneous sources. QUASAR unifies ideas from CONVINSE and EXPLAIGNN: a structured representation is generated to capture the information need, and an evidence graph is constructed from relevant information. Different from EXPLAIGNN, the few most relevant evidence pieces, obtained via iterative GNN inference, are used as context for GLM-based answer generation. This generation step enables coping with more complex questions. Further, we collate the COMPMIX [35] benchmark with 9K questions that exhibit complex phenomena such as comparisons and joins, and require heterogeneous sources for answering.

Table 1.1: Overview of this dissertation’s contributions.

Project	Contributions	Publications
CONVINSE (Chapter 3)	<ul style="list-style-type: none"> • CONVINSE, the first end-to-end method for conversational QA over heterogeneous sources. • CONVMIX, the first benchmark for conversational QA over heterogeneous sources. 	In <i>SIGIR 2022</i> [33].
EXPLAIGNN (Chapter 4)	<ul style="list-style-type: none"> • EXPLAIGNN, an explainable method for conversational QA over heterogeneous sources that efficiently reduces an evidence graph based on iterative graph neural networks. 	In <i>SIGIR 2023</i> [34].
FAITH (Chapter 5)	<ul style="list-style-type: none"> • FAITH, the first QA system that taps into mixed sources for questions with temporal constraints. • TIQ, a benchmark consisting of 10K questions with implicit time constraints. 	In <i>WWW 2024</i> [91] and <i>TempWeb Workshop at WWW 2024</i> [92].
QUASAR (Chapter 6)	<ul style="list-style-type: none"> • QUASAR, a unified system architecture for complex QA over heterogeneous sources. • COMPMIX, a dataset with complex questions that inherently require access to heterogeneous sources. 	In <i>WWW 2024</i> [35] and <i>IEEE Data Engineering Bulletin 2024</i> [37].
REQAP (Chapter 7)	<ul style="list-style-type: none"> • REQAP, a novel approach for complex QA over structured and unstructured personal data. • PERQA, a large-scale benchmark for personal QA with realistic user data and questions. 	In <i>ACL 2025 Findings</i> [38] and <i>CIKM 2025</i> [39].

- Finally, we devise REQAP [38], a novel approach for QA over structured and unstructured personal data. Different from existing work on QA over heterogeneous sources, REQAP can answer questions that require complex operations over hundreds or thousands of relevant evidence pieces. To this end, an executable operator tree is created first, based on a recursive question understanding mechanism. Operators are designed to enable seamless integration of structured and unstructured sources, and the execution of the operator tree yields a traceable answer. For evaluation, we construct the PERQA dataset which synthesizes realistic data and questions based on personas.

Table 1.1 provides an overview of the salient contributions made with this dissertation.

1.5 Publications

Covered publications. The contributions of this dissertation are presented in the following publications with the author as main contributor, and lead author (in all but two cases):

- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
Conversational Question Answering on Heterogeneous Sources.
In *SIGIR 2022 (Long Research Paper)*.
- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
Explainable Conversational Question Answering Heterogeneous Sources via Iterative Graph Neural Networks.
In *SIGIR 2023 (Long Research Paper)*.
- *Zhen Jia, Philipp Christmann, and Gerhard Weikum.*
Faithful Temporal Question Answering over Heterogeneous Sources.
In *WWW 2024 (Long Research Paper)*.

- *Zhen Jia, Philipp Christmann, and Gerhard Weikum.*
TIQ: A Benchmark for Temporal Question Answering with Implicit Time Constraints.
In *TempWeb Workshop at WWW 2024 (Workshop Paper)*.
- *Philipp Christmann and Gerhard Weikum.*
RAG-based Question Answering over Heterogeneous Data and Text.
In *IEEE Data Engineering Bulletin 2024 (Journal Paper)*.
- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
CompMix: A Benchmark for Heterogeneous Question Answering.
In *WWW 2024 (Resource Paper)*.
- *Philipp Christmann and Gerhard Weikum.*
Recursive Question Understanding for Complex Question Answering over Heterogeneous Personal Data.
In *ACL 2025 Findings (Long Research Paper)*.
- *Philipp Christmann and Gerhard Weikum.*
The ReQAP System for Question Answering over Personal Information.
In *CIKM 2025 (Demonstration Paper)*.

Related publications. Further publications by the author that are topically related, but not covered by this dissertation, are:

- *Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum.*
Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion.
In *CIKM 2019 (Long Research Paper)*.
- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
Beyond NED: Fast and Effective Search Space Reduction for Complex Question Answering over Knowledge Bases.
In *WSDM 2022 (Long Research Paper)*.
- *Philipp Christmann.*
Explainable Conversational Question Answering over Heterogeneous Sources.
In *SIGIR 2022 (Doctoral Consortium Paper)*.
- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
Question Entity and Relation Linking to Knowledge Bases via CLOCQ.
In *SMART Challenge at ISWC 2022 (Long Research Paper)*.
- *Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum.*
CLOCQ: A Toolkit for Fast and Easy Access to Knowledge Bases.
In *BTW 2023 (Short Research Paper)*.
- *Philipp Christmann, Svitlana Vakulenko, Ionut-Teodor Sorodoc, Bill Byrne, and Adrià de Gispert.*
Retrieving Contextual Information for Long-Form Question Answering using Weak Supervision.
In *EMNLP 2024 Findings (Short Research Paper)*.

1.6 Outline

Chapter 2 introduces key concepts that are used in the remainder of this dissertation, and provides relevant background and related work. Chapter 3 and Chapter 4 present CONVINSE and EXPLAIGNN, our works on ConvQA. In Chapter 5, we devise FAITH for temporal QA, and present QUASAR in Chapter 6 for complex QA. REQAP, our project on QA over personal data, is presented in Chapter 7. The conclusion, in Chapter 8, discusses this dissertation’s key insights, limitations, and promising directions for future research.

2

Background

Contents

2.1 Concepts and Notation	9
2.1.1 Question Answering	9
2.1.2 Language Model	11
2.1.3 Retrieval and Ranking	12
2.1.4 Graph Neural Network	13
2.2 Related Work	14
2.2.1 Question Answering over a Single Source	14
2.2.2 Question Answering over Heterogeneous Sources	16

2.1 Concepts and Notation

This section introduces salient concepts and notation that are used throughout this dissertation.

2.1.1 Question Answering

Question. A *question* q asks about *factual* information (e.g., “Who wrote the book *Angels and Demons?*”) or *personal* data (e.g., “how often did I play football this year?”). Questions can be *incomplete* with implicit intent, such as “How long is the novel?” or “How much money have I spent on his books?”.

Answer. An *answer* a to a question is either an entity (e.g., Tom Hanks) or a literal such as a date (e.g., 4 May 2009), a year (e.g., 1975), a number (e.g., 3) or a crisp phrase (e.g., No). Sentence-level or long-form answers are not within the scope of this dissertation.

QA system. A *QA system* aims to provide an answer for a given question.

Conversation. A *conversation* C consists of a sequence of questions (q^1, q^2, \dots) and corresponding answers (a^1, a^2, \dots) . The first question q^1 in C is complete, while *follow-up questions* (q^2, q^3, \dots) are usually incomplete. The problem of answering such incomplete questions is referred to as *conversational QA* or *ConvQA*.

2 Background

An example conversation (subset of the conversation in Section 1.1) is shown below:

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

Turn. A *turn* t in a conversation consists of a specific $\langle q^t, a^t \rangle$ pair. For example, the second turn refers to $\langle q^2, a^2 \rangle$.

Knowledge base. A *knowledge base* (KB), also known as *knowledge graph*, is a set of facts that express real-world knowledge. Each *KB-fact* is a $\langle \text{subject}, \text{predicate}, \text{object} \rangle$ (SPO) triple, optionally augmented by $\langle \text{qualifier predicate}, \text{qualifier object} \rangle$ pairs which specify additional information for the main triple. An example is $\langle \text{Angels\&Demons}, \text{cast member}, \text{Tom Hanks}; \text{character role}, \text{Robert Langdon} \rangle$, which specifies that Tom Hanks played the character Robert Langdon in Angels&Demons. Subjects are entities (e.g., Tom Hanks or Angels&Demons), while objects can be entities, types (e.g., human or book) or literals. Predicates (e.g., cast member or character role) denote relationships. Prominent representatives of KBs are Wikidata [220], YAGO [195], DBpedia [3], and Google’s knowledge graph [193].

Named-entity recognition and disambiguation. Given a question, the goal of *named-entity recognition and disambiguation* (NERD) is to identify entity mentions and link them to the corresponding KB-entities. For example, “*Angels and Demons*” is an entity mention in the question “*Who wrote the book Angels and Demons?*”, which is then disambiguated to the book Angels&Demons in the KB. Note that exact matching of surface names is often not sufficient, as in “*GRRMs latest book?*”, where “*GRRM*” refers to the author George R. R. Martin. Further, there are often multiple valid candidates: there is a novel, movie, TV series episode, documentary, album and fashion collection matching with “*Angels and Demons*” in Wikidata. In such situations, top- k disambiguation can improve recall for a downstream QA system utilizing the *linked entities* [31].

Text corpus. A *text corpus*, or *text collection*, is a set of documents, with each document consisting of a sequence of sentences.

Table. A *table* is a structured relational construct consisting of cells organized into rows and columns, with optional row and column headers. Cell values are typically entities or literals, while headers are often predicates.

Infobox. An *infobox* is a special form of a table, capturing salient attribute-value pairs about an entity. A Wikipedia infobox appears on the top right corner of the entity’s Wikipedia page. Infobox entries resemble KB-facts, but they are not necessarily clean in terms of entity linkage (e.g., a birthplace could be given as a string with city, country or other regional variations).

Evidence. *Evidence* is given with an answer as explanatory support. The evidence consists of *information snippets* or *evidence pieces* that are retrieved from a KB, a text corpus, a table, or a Wikipedia infobox.

Answering evidence. By *answering evidence* we refer to the subset of information snippets that, given a specific question q , contains at least one correct answer a for q . It can either mention a as a string, or include a as an entity (in case of KB-facts).

2.1.2 Language Model

The goal of *language modeling* is to estimate the conditional probability of a word or token (or token sequence) occurring in a given context [9]. Within the scope of this dissertation, a *language model* (LM) is defined as a large-scale transformer that is pre-trained on vast amounts of textual data [143, 257]. For clarity, we further differentiate between different types of LMs: (i) encoder LMs, (ii) sequence-to-sequence LMs, and (iii) generative LMs.

Encoder language model. An *encoder language model* (ELM) is trained using the masked language modeling objective [42], and typically processes the input *bi-directionally*. The input is a sequence of text $s = s_1 \dots s_T$ with T tokens and the output is a sequence of T token embeddings with d dimensions each:

$$LM_{encoder}(s) = LM_{encoder}(s_1 \dots s_T) = \mathbf{s}_1 \dots \mathbf{s}_T \quad (2.1)$$

where $\mathbf{s}_t \in \mathbb{R}^d$ is the final hidden state of token t . To derive a single d -dimensional embedding \mathbf{s} , representing the entire input s , we utilize *mean pooling* ($mean : \mathbb{R}^{T \times d} \rightarrow \mathbb{R}^d$):

$$\mathbf{s} = mean(\mathbf{s}_1 \dots \mathbf{s}_T) = \frac{1}{T} \sum_{t=1}^T \mathbf{s}_t \quad (2.2)$$

Prominent examples of encoder language models are BERT [42] and ROBERTA [128].

Sequence-to-sequence language model. A *sequence-to-sequence language model*, or *Seq2seq LM* in short, is a model that first derives an encoding of the input sequence, again typically *bi-directional*, and then decodes the output sequence *autoregressively*. Formally, for an input sequence s with T tokens, the output sequence o with T' tokens is computed as:

$$\begin{aligned} o &= LM_{seq2seq}(s) = LM_{decoder}(LM_{encoder}(s)) \\ &\sim P(o \mid \mathbf{s}_1 \dots \mathbf{s}_T) \end{aligned} \quad (2.3)$$

where $LM_{decoder}$ denotes the autoregressive decoder, and P its probability distribution. This conditional probability of the output is computed as:

$$P(o \mid \mathbf{s}_1 \dots \mathbf{s}_T) = P(o_1 \dots o_{T'} \mid \mathbf{s}_1 \dots \mathbf{s}_T) = \prod_{t=1}^{T'} P(o_t \mid \mathbf{s}_1 \dots \mathbf{s}_T, o_1 \dots o_{t-1}) \quad (2.4)$$

where in each decoding step, the distribution of $P(o_t \mid \mathbf{s}_1 \dots \mathbf{s}_T, o_1 \dots o_{t-1})$ is given by a softmax over all tokens in the output vocabulary of $LM_{seq2seq}$ (which can potentially be different from the input vocabulary for tasks like translation).

Prominent examples of Seq2seq LMs are BART [115] and T5 [165].

Generative language model. The key difference of *generative language models* (GLMs) from Seq2seq models is that the input is processed only *uni-directionally*. Such *decoder-only* models do not derive a contextualized encoding of the input via a separate encoder. Instead, the model initiates its internal activations by processing the input sequence s , often referred to as *prompt*, and then *autoregressively* decodes the output o :

$$\begin{aligned} o &= LM_{generative}(s) \\ &\sim \prod_{t=1}^{T'} P(o_t \mid s_1 \dots s_T, o_1 \dots o_{t-1}) \end{aligned} \quad (2.5)$$

Prominent examples of generative language models are GPT [14] and LLAMA [209].

LM fine-tuning. To adapt an LM to a specific task, the LM’s parameters can be *fine-tuned*. This necessitates a task-specific dataset capturing the target data distribution. For Seq2seq LMs and GLMs this dataset consists of pairs of input sequences (or *prompts* for GLM) and output sequences. For encoder LMs, the format of the fine-tuning dataset highly depends on the downstream task [42] (e.g., sentence pairs and class labels for sentence pair classification, input sentence and per-token tags for sentence tagging, etc.).

In-context learning. Recent GLMs are capable of *in-context learning* (ICL): given a small set of input-output pairs, typically around 10, the GLM can generalize their relationship to unseen inputs [14]. ICL enables strong model performance for a specific task captured by the input-output pairs, even when no training data for fine-tuning is available. The disadvantage of ICL is the increased inference cost due to the GLM’s processing of the input-output pairs, and that strong performance typically requires GLMs with at least 7B parameters.

Sampling strategies. Note that both Seq2seq and generative LMs *sample* the output o based on the output probability distribution. There are different strategies for sampling, including basic approaches like greedy sampling (i.e., applying the argmax function at each decoding step) or beam search, and more sophisticated strategies like Nucleus sampling [80], optionally with temperature scaling. Unless stated otherwise, we utilize beam search for Seq2seq models and Nucleus sampling with temperature scaling for GLMs.

2.1.3 Retrieval and Ranking

For retrieving information from a text corpus, information retrieval and ranking techniques have been developed, often based on LMs [122]. We differentiate between (i) sparse retrieval and (ii) dense retrieval. Both these *first-stage retrieval* paradigms have the goal of identifying a pool of (the most) relevant information (e.g., top- k documents with $k=100$ or 500), given a textual query. The granularity of textual information can vary, from sentence-level to document-level. In the following we use *document* to refer to any such textual information, for readability.

Sparse retrieval. In *sparse retrieval*, each document and query is assigned a *sparse vector*, and *inverted indexes* are used for efficient computation of the retrieval result. A well-known approach for sparse retrieval is BM25 [172], which derives bags of words and ranks documents based on their matching with the query. For each query term, this ranking considers its frequency in the document (TF), and the corresponding *inverse document frequency* (IDF) in the text corpus.

BM25 is restricted to exact matching of query terms with documents, unable to account for semantic matches. This sparked work on *learned sparse retrieval* [5], where the vectors are obtained via encoder LMs enabling such *semantic matching*. The dimension of these learned vectors is typically the size of the LM vocabulary – intuitively, the learned sparse vector *expands* the query and documents with semantically related terms, by assigning relevant tokens non-zero weights. The output is a ranked list of documents. A prominent approach is SPLADE [58].

Dense retrieval. In *dense retrieval*, the query q and each document d^i is assigned a dense vector embedding, with typical dimensions of 512 or 768. These vectors are obtained independently via encoder LMs (*bi-encoders*), and enable semantic matching. The matching is computed via a scoring function f that considers the query embedding (\mathbf{q}) and document embedding (\mathbf{d}^i):

$$\text{score}(q, d^i) = f(\mathbf{q}, \mathbf{d}^i) \tag{2.6}$$

where f is usually implemented as the dot product. For efficiency, most implementations utilize *approximate nearest-neighbor search* [95] within the latent vector space, for deriving the top- k retrieval results.

Prominent examples of dense retrievers are DPR [105] and CONTRIEVER [84].

Cross-encoder. First-stage retrieval computes vector representations for the query and document independently, limiting their expressiveness. To leverage the attention-based interactions between the query and the document, *cross-encoders* (CEs) [122] are commonly used for re-ranking the top- k first-stage retrieval results. The query and document are concatenated (\circ) to form a single input sequence, which enables to compute their cross-attention in the $LM_{encoder}$:

$$score(q, d^i) = f_s (LM_{encoder} (q \circ d^i)) \quad (2.7)$$

The function f_s then computes a scalar after mean pooling from the sequence embeddings, which is used for re-ranking the top- k documents. Alternatively, to classify documents as relevant and irrelevant, f_s can also predict two scalars, one for each class.

Cross-encoders are typically fine-tuned, which often greatly improves their performance on a target data distribution. Fine-tuning requires a dataset of inputs ($q \circ d^i$) and their ground-truth relevance (for re-ranking) or class label (for classification). For re-ranking, providing challenging *negative* samples (i.e., inputs with negative relevance label) is decisive.

2.1.4 Graph Neural Network

Graph neural networks (GNNs) combine machine learning with structured graph representations [228]. GNNs are able to learn the relationships of nodes within a given graph, and utilize these for tasks such as *node classification* or *node ranking*.

Graph. A *graph* G is defined by a set of *vertices* or *nodes* V , and a set of edges E that run between nodes ($E \subset V \times V$).

Graph neighborhood. The *graph neighborhood* N of a node $v \in V$ is defined as $N(v) = \{u \mid (v, u) \in E\}$.

Node initialization. For applying a GNN to a graph G , each node in G is encoded as a vector first. In general, any encoding strategy can be used, including traditional feature engineering. Throughout this dissertation, we use encoder LMs to initialize the features of nodes. Assume that $label(v)$ provides a textual node label for v , then v is initialized as $\mathbf{v}^0 = LM_{encoder}(label(v))$.

Message passing. The core part of the GNN is the *message passing* which distributes information within the graph: encodings of nodes are updated based on the encodings of neighboring nodes. Each message passing step corresponds to a GNN *layer* l , and L denotes the total number of GNN layers. Given a node v , \mathbf{v}^l represents its encoding at layer l . Then the *message passing update* at layer l can be represented as a function f_m of v and its neighbors' encodings in the previous layer $l - 1$:

$$\mathbf{v}^l = f_m (\mathbf{v}^{l-1}, \{\mathbf{u}^{l-1} \mid u \in N(v)\}, X) \quad (2.8)$$

where X are optional graph-external factors, such as a given question. This function f_m is the decisive part of the GNN, and popular implementations utilize simple aggregation (e.g., GCN [110]) or inter-node attention (e.g., GAT [216]).

Node ranking. In the context of QA, nodes typically correspond to answer candidates. Given the node encoding \mathbf{v}^L after L message passing updates, v can be scored to derive a *node ranking*:

$$score(v) = f_r (\mathbf{v}^L) \quad (2.9)$$

where f_r computes a scalar, that induces the ranking among nodes.

2.2 Related Work

Question answering is a long-standing research topic, with early prototypes developed well before the year 2000 (e.g., BASEBALL [66] in 1961 or LUNAR [224] in 1973) – a comprehensive introduction to QA over web content is provided in [175].

This section first provides relevant background on QA over single sources (KBs, text *or* tables) (Section 2.2.1), and then dives into work on QA over heterogeneous sources (Section 2.2.2).

2.2.1 Question Answering over a Single Source

KB-QA. Given their structured representation of factual knowledge, KBs have been a valuable information source for QA [8, 213, 235]. The predominant approach is to map a natural-language question to a SPARQL query, that is then executed on the KB to derive an answer [8, 235]. Early work handcrafted *query templates* to cover specific input question structures, and then disambiguated mentions of entities, predicates and types to populate these templates [210]. DEANNA [235] proposed an integer linear program (ILP) for *joint disambiguation* of such mentions. SEMPRE [10] used *semantic parsing* to capture the information need in a logical form, that was then translated into a SPARQL query. AQQU [8] improved the answering performance by integrating a learning-based entity recognition module, and by utilizing learning-to-rank. QUINT [2] and NEQA [1] devised methods to automatically *learn* query templates, and expand this set of templates online as new user questions are observed. This initial work often applied techniques such as part-of-speech (POS) tagging or dependency parsing to process input questions, and progress was driven by benchmarks such as WEBQUESTIONS [10], FREE917 [17] and the QALD series [131, 211, 212]. The WDAQUA [44] system simultaneously queries multiple KBs for answering questions. More recent work utilizes Seq2seq models or generative LMs to generate SPARQL queries, enabling them to cope with more complex questions [7, 130]. Benchmarks such as COMPLEXWEBQUESTIONS [203] or LC-QUAD2.0 [48] have been released to track the progress of this line of research.

There has also been work on utilizing graph-based approaches, such as GNNs, for KB-QA [29, 93, 213, 243]. This line of work does not generate an explicit *SPARQL query*, but constructs an entity-centric KB subgraph, based on linked entities in the question. Specifically, KB-facts with disambiguated entities are retrieved and cast into a graph structure. This subgraph with question-relevant information is then used for computing an answer. Such graph-based methods have also been proposed for multi-hop QA [253] and conversational QA [32], with judicious expansion of the initial subgraph.

For both SPARQL-based and graph-based approaches, KB-QA heavily relies on the correct disambiguation of entities, predicates and types. Entity linking is often outsourced to external NERD tools [55, 79], and many incorrect answers by QA systems can actually be traced back to disambiguation errors. To alleviate this problem, we developed CLOCQ [31], which disambiguates all mentions of entities, predicates and types *jointly*, and returns top- k results for each mention, to account for such errors.

Despite big advances in information extraction, KBs are still inherently incomplete [223]. This limits the answer coverage of any QA system operating solely over a KB.

Text-QA. Early work on Text-QA utilized sparse retrieval to retrieve relevant passages, and applied simple techniques such as pattern matching [168], dependency parsing [189], or n -gram mining [49] to extract direct answers. Triggered by the SQUAD [166] benchmark released in 2016, there has been a long line of research on *machine reading comprehension* (MRC), where a question and passage are given, and the target answer is a segment in this input passage (e.g., [185, 246]).

The prevalent approach was *span prediction*, predicting the start and end token of the answer. Similar to KB-QA, MRC was greatly affected by the advent of LMs [42, 128, 165].

DRQA [23] was pivotal work, as it combined sparse retrieval and MRC, and established the *retrieve-and-read* paradigm. The corresponding end-to-end task of retrieval from a text corpus, and answering based on retrieval results, is often referred to as *open-domain QA* [260]. BERTSERINI [238] improves open-domain QA performance by building on the ANSERINI [236] retrieval toolkit. Recent advances include improvements of the retriever component (e.g., DPR [105] or CONTRIEVER [84]), novel generative approaches such as the Fusion-in-Decoder (FID) [85] architecture, or enabling end-to-end training for *retrieval-augmented generation* (RAG) [116]. Again, large-scale datasets, such as TRIVIAQA [96] or NATURALQUESTIONS [111], have been driving research on Text-QA.

Even though the answer coverage of textual data is typically higher than that of a KB, textual data is often harder to process, as the same information can be expressed in many different ways. Also, numerical information is scarcely represented in text. This limits the capabilities of QA systems for more complex questions that require aggregation of semantically-similar information.

Table-QA. In parallel to advances in KB-QA and Text-QA, there has been work on utilizing tables for QA – either *web tables* or *relational database tables*.

Table search aims to find relevant web tables for a keyword query (e.g., WEBTABLES [15] or TABLE [217]), but does not provide direct answers. Comparable to MRC but for tables, [157] devised an approach for detecting a cell that answers a question for a given web table, and released the WIKITABLEQUESTIONS dataset for this task. Their approach applies semantic parsing techniques to a KB derived from the provided table. TAPAS [77] utilizes BERT to predict an aggregation function and a subset of table cells for answering more complex questions, based on a given table. Sun et al. [198] developed a framework for cell search from a large *collection* of web tables. Similarly, TABLEILP [108] and FRETS [88] enable answering science questions over a set of relevant tables obtained via sparse retrieval from a table collection. Chakrabarti et al. [20] extends the capabilities of QA over table collections by integrating semantic matching, based on deep neural networks, for scoring table relevance. The method presented by Herzig et al. [76] follows a similar direction and extends TAPAS with an encoder LM for dense table retrieval.

The goal in *text-to-SQL* (Text2SQL) or *natural-language-to-SQL* (NL2SQL) is to translate a natural-language query to SQL code [106, 127] that can be executed over a *relational database* (DB). In 1996, CHILL [251] was proposed as a parsing framework to map natural language to a SQL query. This early work heavily relied on manual engineering, such as handcrafted predicates for an intermediate logical form and reliance on templates. Text2SQL was greatly advanced by the *sequence-to-sequence* (Seq2seq) architecture [200], that encodes an input natural-language query and the *database schema* and decodes SQL code [233], and the SPIDER dataset [248]. The DB schema includes information about the tables, column headers and column types of the underlying database, and is crucial for the performance of Text2SQL systems. To model the connections between the DB schema and the NL input, GNN encodings have been explored as a promising direction [18, 65]. Current state-of-the-art methods leverage the pre-training of generative LMs with code repositories (e.g., GitHub data), and further fine-tune with question-query pairs [53, 59, 118, 159].

Web tables and DB tables often store detailed numerical information that is absent in curated KBs or textual data. However, similar to KBs, tables are limited in answer coverage, and would not express certain kinds of information such as the content of a book. Further, tables often require context (e.g., page titles for Wikipedia tables) to be understood correctly.

2.2.2 Question Answering over Heterogeneous Sources

QA systems operating on a single source are inherently limited in their answer coverage, which sparked work on integrating multiple *heterogeneous* sources into the answering process.

Late fusion of information. Given that QA approaches operating over single sources are fairly different, depending on the specific source (see Section 2.2.1), combining KBs, text corpora and tables within a unified QA system is not straightforward. Hence, preliminary work on heterogeneous QA followed a *late fusion* paradigm, in which information is retrieved and answers are computed for each source independently. These independent predictions are then aggregated for deriving the final answer. The DEEPQA [56] architecture, well-known for its human expert level performance at the “*Jeopardy!*” quiz show, followed this paradigm. QA research on augmenting incomplete KBs with textual data adopted a similar framework [180, 181, 231, 232].

Such late fusion is limited in expressiveness, as the information within individual sources is only considered independently for deriving answer candidates. Ideally, evidence from multiple sources is combined within one unified approach for identifying answer candidates *and* the answer.

Graph-based fusion. GRAFT-NET [197] takes a different approach, and unifies KB and text in a graph structure, with *early fusion* of information. For a given question, a heterogeneous graph with KB-facts and text snippets is created. The answer is then derived via a GNN. PULL-NET [196] extended these ideas, and improved answer recall in the question graphs by expanding the graph iteratively, controlled by a classifier. HETEROQA [60] applies similar ideas, with a unified graph structure and GNN inference, for answering community questions. KG-FID [247] identifies KB-entities in text passages, and then uses the KB to impose a graph structure among these passages, for GNN-based re-ranking. While this approach leverages structural cues of the KB, no evidence in the form of KB-facts is integrated. UNIQORN [160] constructs a question-specific graph from a KB and text, but considers a finer node granularity. For textual data, a *quasi-KB* is constructed via on-the-fly information extraction, and then each entity and predicate in the KB or quasi-KB becomes a node in the graph. The answer is computed by identifying Group Steiner Trees (GSTs).

These works combine a KB with textual data, without considering tables. Our own work, EXPLAIGNN, is loosely inspired by GRAFT-NET and PULLNET, but also incorporates information from tables and infoboxes into its heterogeneous evidence graph.

Verbalization. The *verbalization* paradigm was introduced by UNIK-QA [154], which integrates the full range of KB, text corpus and web tables. Information from all sources is *verbalized* into a textual form, uniformly retrieved via a dense retriever, and given as context to a Seq2seq LM for generating the answer. During verbalization, KB-facts are linearized by concatenating the labels of its individual constituents. Text documents from a corpus are split into passages or sentences. Table rows are brought into a sentence-like structure by concatenating cell values with the respective column headers. Similarly, each infobox entry is verbalized line-by-line. For tables and infoboxes, additional context (such as the page title or the table caption) is often integrated. Illustrative examples of verbalized evidence for each source are shown in Table 2.1.

Note that there are differences in the granularity used for verbalized evidence pieces: UNIK-QA combines multiple KB-facts, splits text on a passage level, groups information per table, and summarizes each infobox within *one* evidence piece. When verbalizing in our work (in CONVINSE, EXPLAIGNN, FAITH and QUASAR) we apply a finer granularity: each KB-fact, text sentence, table row and infobox entry becomes one verbalized piece of evidence (similar to Table 2.1). This avoids distracting the QA system by the presence of semantically disjoint information within a single piece of evidence.

Table 2.1: Example of verbalized evidence from different input sources.

Source	Original evidence form	Verbalized evidence form											
KB	<code>< Angels&Demons, cast member, Tom Hanks; character role, Robert Langdon ></code>	<i>“Angels&Demons, cast member, Tom Hanks, character role, Robert Langdon”</i>											
Text	Professor Robert Langdon is flown to CERN to help investigate the murder of Leonardo Vetra.	<i>“Professor Robert Langdon is flown to CERN to help investigate the murder of Leonardo Vetra.”</i>											
Table	<i>The Lost Symbol (TV series) – Episodes</i>												
	<table border="1"> <thead> <tr> <th>No.</th> <th>Title</th> <th>Directed by</th> <th>Release date</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>As Above So Below</td> <td>Dan Trachtenberg</td> <td>September 16, 2021</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	No.	Title	Directed by	Release date	1	As Above So Below	Dan Trachtenberg	September 16, 2021
No.	Title	Directed by	Release date										
1	As Above So Below	Dan Trachtenberg	September 16, 2021										
...										
Infobox	<table border="1"> <thead> <tr> <th colspan="2"><i>Angels & Demons</i></th> </tr> </thead> <tbody> <tr> <td>Author</td> <td>Dan Brown</td> </tr> <tr> <td>Pages</td> <td>768</td> </tr> <tr> <td></td> <td>...</td> </tr> </tbody> </table>	<i>Angels & Demons</i>		Author	Dan Brown	Pages	768		...	<i>“Angels&Demons, Author, Dan Brown”</i> <i>“Angels&Demons, Pages, 768”</i>			
<i>Angels & Demons</i>													
Author	Dan Brown												
Pages	768												
	...												

UDT-QA [134] extends UNIK-QA and utilizes a Seq2seq LM during verbalization to generate fluent text passages for all input sources. While this approach slightly improves performance in some scenarios, the additional LM involvement might perturb the original source information. Also, this entails running a Seq2seq LM on *all* input sources. MATTER [114] improves the throughput of the UNIK-QA architecture, by integrating efficient encodings of verbalized evidence. RAGONITE [178] explores different strategies to *contextualize* during verbalization.

Verbalization demonstrates strong performance when few evidence snippets are sufficient for deriving an answer, and has been used in four projects presented in this dissertation. However, for questions that require a larger set of evidence pieces verbalization fails, as both Seq2seq LMs and GLMs are restricted by their (input) context window size. This problem is especially pronounced for questions that entail more complex operations, such as aggregation, joins, grouping, or ordering. Current LMs are not suitable for computing such operations precisely.

Heterogeneous retrieval. There has also been preliminary work on improving the retrieval stage for heterogeneous sources [142]. This work augments our COMPMIX [35] benchmark with annotations for retrieval, and proposes the UNIHGKR framework [142] for fine-tuning an encoder LM for dense retrieval from heterogeneous sources. Recently, SPAGHETTI [252] proposed to employ a source-specific retrieval strategy. Their pipeline derives evidence from a KB via a SPARQL query, but verbalizes tables and infoboxes. In addition, information generated by a GLM is added as evidence. A generative LM is also used to generate an answer based on these source-specific answers and evidence pieces.

Conversational QA. ConvQA over heterogeneous sources was first explored in our CONVINSE [33] project. Subsequently, Deng et al. [41] release the PACIFIC benchmark for ConvQA over heterogeneous financial data, and present UNIPCQA, a unified Seq2seq framework, for solving this task. EXPLAIGNN [34] improves upon CONVINSE by combining the verbalization paradigm with structured graph processing, utilizing the intra-relationships of verbalized evidence pieces. Recent work builds upon our framework and improves the performance on our CONVMIX benchmark, by utilizing GNNs to derive a self-sufficient question representation [119],

integrating a GLM [86] or optimizing sub-tasks via direct preference optimization (DPO) [101]. SUQL [125] explores conversational search over structured data and unstructured text by extending SQL with special user-defined functions (UDFs) that process free text.

Complex QA. There has been a large body of work on multi-hop QA [26, 183, 240] that aims to answer quiz-like questions such as “*what is the birth date of the author of Angels&Demons?*”. Such multi-hop questions involve joining multiple facts in a chain. Part of this work also considers text and tables as information sources, driven by the HYBRIDQA [26] benchmark and its extension OTTQA [24]. Typical approaches utilize GNNs for identifying an answering path in a subgraph [54] or rank such paths via LMs [135].

Apart from this work on multi-hop QA, answering complex questions via heterogeneous sources has been somewhat underexplored. Some of the graph-based approaches discussed before, including PULLNET and UNIQORN, are able to cope with questions that refer to multiple entities and relations, generalizing over such multi-hop QA. However, those approaches fall short on more complex questions that involve aggregation, grouping, or temporal constraints. Our FAITH [91] project targets questions with temporal constraints via heterogeneous sources, and is the first of its kind. REQAP [38] investigates answering complex questions over heterogeneous personal data, that entail complex operations such as aggregation, joins, or grouping, but also temporal constraints.

Domain-specific QA. Shen et al. [191] release a benchmark for product QA over heterogeneous sources, and compare different techniques with verbalization. Taffa et al. [202] explore answering scholarly questions over KBs and text, and also pursue a verbalization-based approach that employs entity-centric retrieval via SPARQL queries. The STARK [226] dataset is a collection of domain-specific benchmarks for product search, academic paper search, and medical QA. Each benchmark requires access to structured and unstructured data. Zhao et al. [258] utilize the verbalization paradigm for answering medical questions. FINQA [28] and TAT-QA [259] are datasets for the financial domain, and TsQA [120] covers geographical data.

Multi-modal QA. Recently, there has been interest in combining text with visual inputs (i.e., images or videos) for question answering [74, 204]. MANYMODALQA [74] was one of the first datasets in this research area, and necessitates identifying which single modality (tables, text or images) to utilize for a given question. The MULTIMODALQA [204] benchmark requires the combination of different modalities for individual questions. Again, construction of question-specific graphs and GNN inference is a common strategy for such multi-modal QA [75, 237]. Interestingly, UNIMMQA [133] demonstrates strong performance with a verbalization-based approach, that utilizes image captioning to integrate visual content. More recent work builds upon advances in vision language models to verbalize image contents [121, 126]. Visual inputs are not within the scope of this dissertation.

3

Conversational Question Answering

This chapter introduces CONVINSE, the first methodology for answering conversational questions with implicit user-intent, that integrates information from heterogeneous sources. We introduce a general 3-stage pipeline, extending the common retrieve-and-read [23] procedure with a question understanding stage. This initial stage generates an explicit structured representation of an incoming question and its conversational context, and can be matched against evidence from mixed sources. For experiments, we construct the CONVMIX dataset comprising 3,000 real-user conversations with 16,000 questions, along with ground-truth answer annotations.

All code, data and a demo are available at <https://convinse.mpi-inf.mpg.de>.

Contents

3.1	Introduction	20
3.2	The CONVINSE Methodology	22
3.2.1	Question Understanding (QU)	22
3.2.2	Evidence Retrieval and Scoring (ERS)	24
3.2.3	Heterogeneous Answering (HA)	25
3.2.4	Distantly Supervised Labeling	26
3.2.5	Training of the CONVINSE Methodology	26
3.3	The CONVMIX Benchmark	27
3.3.1	Existing Benchmarks and their Limitations	27
3.3.2	Construction of the CONVMIX Benchmark	27
3.4	Experimental Setup	29
3.4.1	Heterogeneous Sources	29
3.4.2	Baselines	30
3.4.3	Metrics	30
3.4.4	Configurations	31
3.5	Experimental Results and Insights	31
3.5.1	Key Findings	31
3.5.2	Analysis	34
3.6	Demonstration	36
3.7	Related Work	36
3.8	Conclusion	38

3.1 Introduction

Motivation. Conversational question answering (ConvQA) [30, 162, 169, 175] is a popular mode of communication with digital personal assistants like Alexa, Cortana, Siri, the Google Assistant, or more recently ChatGPT and Gemini, that are ubiquitous in today’s devices. In ConvQA, users pose questions to the system sequentially, over multiple turns. Follow-up questions in such conversations usually contain *implicit* context. The ConvQA system is expected to resolve such implicit information from the conversational history. Consider, for example, a typical ConvQA session on factual knowledge below:

*q*¹: Who played Jaime Lannister in *GoT*?
*a*¹: Nikolaj Coster-Waldau
*q*²: What about the dwarf?
*a*²: Peter Dinklage
*q*³: When was he born?
*a*³: 11 June 1969
*q*⁴: Release date of first season?
*a*⁴: 17 April 2011
*q*⁵: Duration of an episode?
*a*⁵: 50–82 minutes

Understanding these information needs within the conversational context is a major challenge. Importantly, the *topic* or relevant entities in a conversation can often *shift*, as in *q*⁴, which has no connection to the previous question *q*³, but refers to “*GoT*” in *q*¹. Hence, the ConvQA system needs to identify relevant parts of the conversational context for answering the current question.

Limitations of the State of the Art. State-of-the-art work on ConvQA makes use of a single information source only: either curated knowledge bases (KB) [32, 68, 97, 100, 112, 137, 158, 177, 190], or unstructured text collections [27, 82, 161, 163, 164], or web tables [83, 144], but only one of these. However, even questions within one conversation often require access to different types of information sources, as illustrated by our running example. While questions *q*¹ and *q*³ can be answered conveniently using KBs, answering *q*², *q*⁴ or *q*⁵ via a KB requires complex reasoning efforts, even if all information was present in the KB. In contrast, the answers to these three questions are much more easily spotted in content of text documents or web tables. In addition, there are obviously many information needs where answers are present only in unstructured textual form, as KBs and web tables have inherently limited coverage. An example would be a question like: “What did Brienne call Jaime?” (**Kingslayer**). A smart ConvQA system should, therefore, be able to tap into more than one kind of knowledge repository, to improve answer recall and to boost answer confidence by leveraging multiple kinds of evidence across sources (plausible for *a*¹ and *a*³, say).

However, integrating heterogeneous sources into existing ConvQA systems is non-trivial. When specializing on a single type of information source, methods often adopt source-specific design choices that do not generalize well [32, 68, 82]. For example, representations of the conversational context, like KB subgraphs or text passages, are often specifically modeled for the knowledge repository at hand, making these heterogeneous sources apparently incompatible.

An orthogonal direction is to augment the question itself for making the intent *explicit*. Simple heuristics add the previous turn or the initial turn (or both) as context, which can result in extremely long and noisy queries. Query resolution [219] appends relevant keywords from the conversational history, which leads to crisper queries but at the cost of losing structure. More sophisticated approaches generate a full-fledged and grammatically correct question from

Table 3.1: A comparison of question understanding approaches for ConvQA.

Original question (q^4)	<i>Release date of first season?</i>
Question resolution [219]	<i>Release date of first season? in GoT</i>
Question rewriting [167]	<i>What was the release date of the first season of GoT?</i>
SR (ours)	$\langle \text{GoT} \mid \text{first season} \mid \text{release date} \mid \text{date} \rangle$

scratch [167, 214]. This adds major complexity and may lose valuable cues from the conversation flow. Further, these methods also face evidence retrieval problems arising from long and potentially verbose questions [69].

Finally, none of the prior works on ConvQA produce human-interpretable structures that could assist end users in case of erroneous system responses.

Approach. To overcome these limitations, we propose CONVINSE (ConvQA with Intermediate Representations on Heterogeneous Sources for Explainability), an end-to-end framework for answering conversational questions on a mixture of sources. CONVINSE has three stages:

- (i) *question understanding* (QU),
- (ii) *evidence retrieval and scoring* (ERS),
- (iii) *heterogeneous answering* (HA).

The first stage, QU, is our primary contribution in this work. It addresses the challenges of incomplete user utterances introduced by the conversational setting. We derive an *intent-explicit structured representation* (SR) that captures the complete information need. Table 3.1 shows such an SR for q^4 of our running example. SRs are frame-like structures for a question that contain designated slots for open-vocabulary lexical representations of entities in the conversational context (marked gray) and the current question (**red**), relational predicates (**blue**), and expected answer types (**cyan**). SRs can be viewed as concise gists of user intents, intended to be in a form independent of any specific answering source. They are self-contained interpretable representations of the user’s information need, and are generated using fine-tuned Seq2seq LMs. For fine-tuning, we devise a mechanism that derives training data via distant supervision from plain sequences of QA pairs. We further propose a *conversational flow graph* (CFG), which can be inferred from the SR, and enhances the explainability of the answer derivation process. SRs are intentionally non-canonicalized and thus independent of any specific information source.

The second stage, ERS, exploits recent developments in entity-based retrieval [31] to judiciously retrieve question-relevant evidence from each information source (KB, text and tables). This evidence from heterogeneous sources is verbalized [154] on-the-fly and each evidence snippet is run through a scoring model. The top- k snippets are passed to the answering stage.

The third and final stage, HA, leverages FID [85], a state-of-the-art answering model for open-domain QA. The FID model is trained for generating a crisp answer from the top- k evidence snippets. This answer is returned to the end user, along with the SR and potentially the CFG.

Benchmark. Another contribution is the construction of CONVMIX, the first benchmark for ConvQA over heterogeneous sources. CONVMIX is a crowdsourced dataset that contains questions with answers emanating from the Wikidata KB, the full text of Wikipedia articles, and the collection of Wikipedia tables and infoboxes. CONVMIX contains 2,800 conversations with five turns (14K utterances), and 200 conversations with ten turns (2K utterances), their gold answers and respective knowledge sources for answering. Conversations are accompanied by metadata like entity annotations, completed questions, and paraphrases.

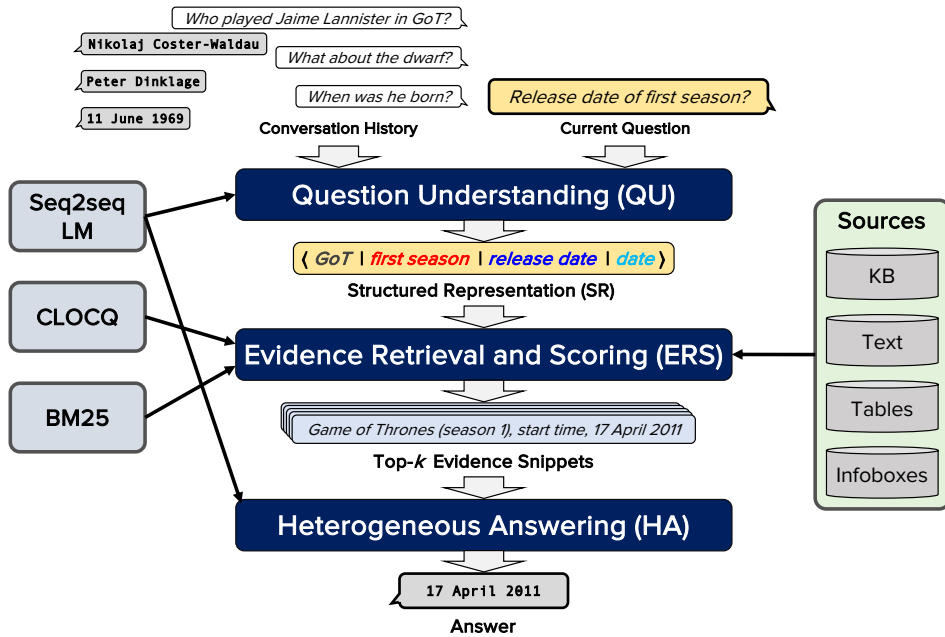


Figure 3.1: An overview of the CONVINSE methodology.

Contributions. The salient contributions of this work are the following:

- Proposing CONVINSE, the first end-to-end method for ConvQA over heterogeneous sources.
- Introducing structured representations to capture user intents in a structured and explainable manner, a key element for seamless answering over a mixture of heterogeneous sources.
- Presenting a distant supervision mechanism for automatically annotating conversations with structured representations.
- Collecting CONVMIX, the first benchmark for ConvQA over heterogeneous sources.

3.2 The CONVINSE Methodology

Figure 3.1 shows an overview of the CONVINSE pipeline. The following subsections discuss the three stages: question understanding, evidence retrieval and scoring, and heterogeneous answering.

3.2.1 Question Understanding (QU)

Follow-up questions in a conversation (q^2, q^3, \dots) usually contain implicit intent. A key challenge in ConvQA, therefore, is to understand the flow of the conversation, towards deriving intent-explicit structured representations (SRs) of the user’s information need. Instead of trying to generate a full-fledged question, we rather aim to capture the semantic structure, using a *syntax-agnostic* approach. This can be perceived as a “logical form” for heterogeneous QA, where no explicit grounding or canonicalization is possible. The representation is purely on the question-level, and thus agnostic to the information sources that are used during the answering process. However, it can readily be matched with different kinds of evidence pieces, which often take the form of keyword phrases (e.g., verbalized table records or KB-facts).

Specifically, we define an SR as a 4-tuple holding a slot for each of:

- Context entities (depicted in gray in Table 3.1),
- Question entities (in red),
- Question predicates (in blue), and,
- Expected answer types (in cyan).

Context and question entities. As an example, consider the gold SR for q^2 of the running example: $\langle GoT \mid the\ dwarf \mid who\ played \mid human \rangle$. The context entity (GoT in this case) is an entity mention from the conversational context. The question entity is the entity mention targeted in the current question (e.g., *the dwarf*). Here, the context entity makes the question entity explicit, indicating that the question is on the dwarf in Game of Thrones. Inferring the question entity may need to take the history into account (e.g., for q^4 in Table 3.1). The context entity and question entity can consist of multiple such mentions. This is required for questions such as “Where did Dany and Jon first meet?”, with the gold SR being $\langle GoT \mid Dany\ and\ Jon\ Snow \mid first\ meet \mid location \rangle$.

Question predicates. The question predicate is the counterpart to the relation or attribute of interest in a logical form. However, it is merely a surface phrase, without any normalization or mapping to a KB. This way, it is easy to match it against any kind of information source. For example, the question predicates *who played* or *first meet* can be matched with evidence from KB or text sentences from documents or table headers, alike.

Answer types. Expected answer types assist the answering model in detecting and eliminating spurious answer candidates [175, 261]. In general, multiple types could be inferred here. The question predicate *first meet* alone could imply the answer type to be either “date” or “location”. Stop words like “where” are often disregarded by downstream QA models [32]; in contrast, the SR answer type retains this information and would infer only the correct “location”. Further, this type can help in identifying the expected granularity of the answer. For the question “When is his birth date?”, one would expect a complete date with day, month and year as the answer, but for the question “When did they win their last world cup?” the corresponding year would be expected as answer. The fourth slot would be populated with “date” and “year”, respectively.

Specific slots in the SR can be left blank. For q^5 , the generated question entity GoT is already explicit, and thus no context entity is required: $\langle _ \mid GoT \mid duration\ of\ an\ episode \mid number \rangle$.

Implementation. The SR generation is implemented by fine-tuning a Seq2seq LM. We tried BART [115] and T5 [165] in preliminary experiments, and found BART to perform slightly better. BART is particularly effective when information is copied and manipulated from the input to generate the output autoregressively [115], which is exactly the setting here. The conversation history and the current question concatenated with a delimiter constitute the input, and the SR is the output. When encoding the history and the current question, the model considers cross-attention between turns, identifying relevant parts from the conversation history.

SRs and explainability. One of our primary goals in CONVINSE was to produce intermediate representations for end users as we proceed through the QA pipeline. Concretely, understanding the flow within the conversation is an essential problem in ConvQA [32, 82]. While the SR itself is human-readable, when presented only with the SR (or some rewritten/resolved question), certain decisions of the ConvQA system might not be immediately obvious to a real user. Here, we propose an intuitive mechanism to infer and present the conversational flow to a user: given the generated SR, we identify the source turn for each word, using exact match in the history, and consider such source turns as relevant for the current question. If there is no source turn, we

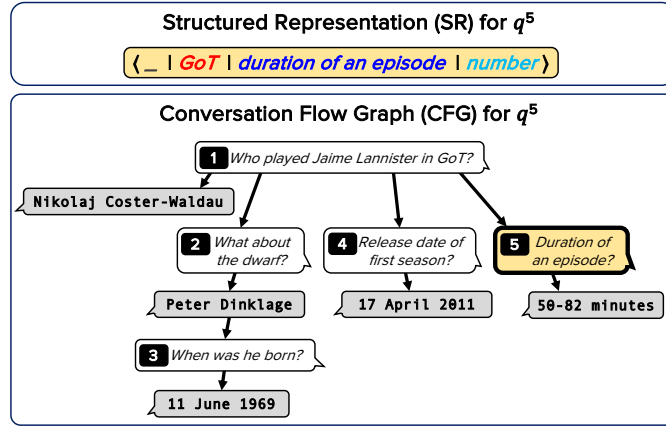


Figure 3.2: Conversational flow graph for our running example.

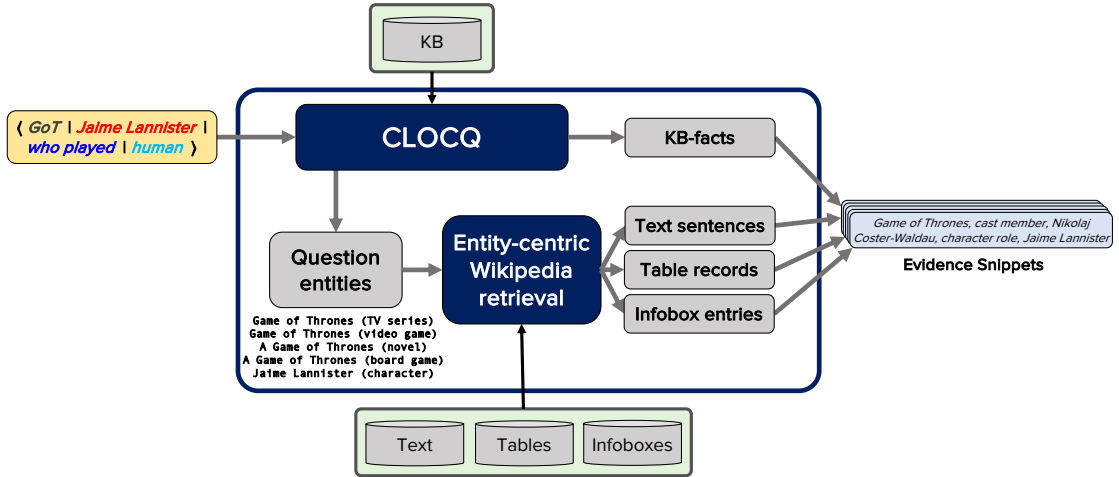


Figure 3.3: An overview of the evidence retrieval.

consider the question at hand as self-sufficient. A *conversational flow graph* (CFG) is established as follows: questions and answers are nodes, and an edge connects a question to its relevant history. Due to the potential dependence of a turn on multiple preceding ones, the CFG for a conversation may not strictly be a tree, but rather a directed acyclic graph (DAG). The CFG can be presented to the interested end user together with the SR, as depicted for q^5 in Figure 3.2, either for gaining confidence in final answers, or for scrutinizing error cases.

3.2.2 Evidence Retrieval and Scoring (ERS)

Evidence retrieval. At this stage, the goal is to retrieve relevant evidence given the generated SR. We then convert retrieved evidence snippets on-the-fly to verbalized NL forms [154]. Table 3.2 shows example evidence from different sources for the context entity *Game of Thrones*.

For evidence from the *KB*, we make use of our CLOCQ¹ methodology [31], which is a recent method for retrieving relevant KB-facts for answering a question. Since questions are treated

¹<https://clocq.mpi-inf.mpg.de>

Table 3.2: Verbalized evidence from different input sources.

KB	“ <i>Game of Thrones, cast member, Nikolaj Coster-Waldau, character role, Jaime Lannister</i> ”
Text	“ <i>Game of Thrones, The third and youngest Lannister sibling is the dwarf Tyrion (Peter Dinklage).</i> ”
Tables	“ <i>Game of Thrones, Season is Season 1, (...), First aired is April 17, 2011 (...).</i> ”
Infobox	“ <i>Game of Thrones, Running time, 50–82 minutes</i> ”

as keyword queries, the SR can directly be fed into CLOCQ (removing the separator ‘|’ during retrieval). KB-facts are verbalized [93, 154, 160], separating the individual constituents of a fact by a comma. The mapping between verbalized facts and the original KB-items are retained in-memory to help us later during evaluation (Section 3.4.3).

For *text*, we take an *entity-centric view* of Wikipedia: apart from relevant KB-facts, CLOCQ also provides top- k disambiguations for mentions of entities, predicates or types. For each disambiguated entity, we retrieve the corresponding Wikipedia page. For example, for the SR $\langle \textit{GoT} \mid \textit{Jaime Lannister} \mid \textit{who played} \mid \textit{human} \rangle$, we would consider the Wikipedia pages for **Jaime Lannister**, **Game of Thrones (TV series)**, **A Game of Thrones (novel)**, and so on (CLOCQ allows for multiple disambiguations per question token). The text within these Wikipedia pages is split into single sentences.

We extract *tables and infoboxes* from the same Wikipedia pages. Every table row is transformed to text individually, concatenating cell values with the respective column headers with an “*is*” in between, and separating such (cell, header) pairs with a comma [154]. For each infobox attribute (similar to predicates in a KB), we concatenate the corresponding lines, having again, a comma as separator.

The page title is prepended to all evidence pieces from Wikipedia as additional context. In addition, we exploit anchor links in evidence snippets to other Wikipedia pages for mapping the corresponding entity mentions (anchor texts) to KB-items. Wikipedia links are transformed to Wikidata IDs using a dictionary. Similarly, dates and years in evidence snippets are identified and normalized to the standard KB format using simple patterns. We keep such (entity mention, KB-item) pairs for all snippets in memory, as this helps us later in grounding answers to the KB during evaluation (Section 3.4.3). For example, for the text evidence in Table 3.2, “*Tyrion*” would link to the Wikipedia page of **Tyrion Lannister**: so we add the pair $\langle \textit{“Tyrion”}, \textit{Tyrion Lannister} \rangle$ as metadata to the corresponding evidence.

Evidence scoring. The set of evidence pieces compiled by the previous stage can be quite large (several thousands), which can affect the efficiency and effectiveness of the answering phase. Therefore, we first reduce this set, keeping only the most relevant information. Since all evidence is verbalized at this stage, each individual evidence snippet can be treated as a document, with the SR as the query. We then use the standard IR scoring model BM25 [172] for identifying the top- k relevant pieces of information.

3.2.3 Heterogeneous Answering (HA)

Given the top- k relevant evidence snippets per question, we make use of the state-of-the-art Fusion-in-Decoder (FID) [85] model. FID *generates* the answer, following a sequence-to-sequence approach. FID first pairs every evidence with the question, which is the SR in our case, and encodes these pairs, leveraging cross-attention to identify relevant information. The concatenation of these encodings is then fed into a decoder, which generates the answer autoregressively.

3.2.4 Distantly Supervised Labeling

Intuition. To train the QU phase of CONVINSE, a collection of ⟨history, question⟩-pairs with the corresponding gold SRs is required. Annotating conversations with such information is tricky with crowdworkers, and expensive too (cf. Section 3.3). Therefore, we devise a mechanism to automatically generate the gold SRs from pure conversations. Our technique is based on the following intuition: if a piece of information (e.g., an entity or relation phrase) in a previous turn is (i) essential for the understanding of the current incomplete question, and (ii) this information has been left implicit by the user, then it should be included in a completed version of the question. Consider this example:

q^1 : *Who played Jaime Lannister in GoT?*
 a^1 : *Nikolaj Coster-Waldau*
 q^2 : *What about the dwarf?*

It is unlikely that proper evidence (on Game of Thrones), can be found for the incomplete question q^2 . Clearly, important information from the conversational context is missing. However, once “GoT” is added to q^2 (e.g., “*What about the dwarf in GoT?*”), evidence with the correct answer can be retrieved. This suggests that the phrase “GoT” should feature in the SR for q^2 .

Implementation. Based on this idea, we create data for training the QU model as follows: starting with the complete question q^1 , we retrieve all evidence snippets (following the same procedure as outlined in Section 3.2.2). For *answering evidence*, that contains the annotated ground-truth answer, we identify the corresponding entity mention that caused the retrieval of this evidence (there can be multiple). In the example above, this would be the mention “GoT”, as answering evidence would be retrieved for the linked entity **Game of Thrones (TV series)**. Such entity mentions are then considered as relevant for the respective conversation turn. For the incomplete follow-up questions, we iteratively add such relevant entity mentions from previous turns (via simple concatenation), and retrieve evidence for the augmented question at hand. Whenever adding an entity mention results in answering evidence being retrieved, we consider the mention as relevant for the current turn. Similarly, entity mentions in the current turn are identified as relevant, if answering evidence is retrieved for them.

We then annotate SRs heuristically. Entity slots are populated as follows:

- (i) If there are relevant entities from the current turn, then these feature in the question entity slot, and relevant entities (if any) from previous turns become context entities in the SR.
- (ii) If there are only relevant entities from previous turns, then these become question entities, and the slot for context entities is left blank.

The remaining words in the current question (except for stop words) fill up the question predicate slot. The expected answer type is directly looked up from the KB, using the annotated ground-truth answer. Since the KB may have several types for the same entity, we take the most frequent one to avoid very specific types. For example, **Tyrion Lannister** has the types **fictional human** (more frequent) and **GoT character** (less frequent) in Wikidata. Only **fictional human** is considered in our SR. In case we do not find answering evidence for a question (e.g., the sources do not have the answer, or the retrieval fails), the instance is skipped during this training data collection step.

3.2.5 Training of the CONVINSE Methodology

We train the QU model first, using data generated as discussed in Section 3.2.4. The trained QU model is applied on the whole data (including training data), to generate SRs. In the remainder

Table 3.3: Basic statistics for the CONVMIX benchmark.

Title	Generate 5 conversations for question answering
Description	Choose entities of your choice from five domains, generate questions about them, and find answers from Wikidata and Wikipedia
Participants	32 unique Master Turkers
Time allotted (per HIT)	4 hours maximum
Time taken (per HIT)	1.5 hours on average
Payment (per HIT)	15 USD
Domains	Books, Movies, Music, TV series, Soccer
Conversations	3,000
Questions	16,000
Question length	8.78 words (initial), 5.19 (follow-ups), 5.87 (all)
Answer size	1.02 entities/strings on average
Entities covered	5,418 (long-tail: 2,511, with <50 KB-facts)
Heterogeneity	2,626 conversations (>1 source used by Turker)

of the approach, we utilize only the SRs generated by our model. When training the FID model, we skip instances for which the top- k evidence pieces do not have the ground-truth answer, since this could lead to hallucinations at inference time [174]. This way, the model is taught to predict the answer from the input. Since evidence from different sources is treated in a unified way, we train one FID model on the top- k evidence snippets after retrieval from all sources.

3.3 The CONVMIX Benchmark

3.3.1 Existing Benchmarks and their Limitations

Notable efforts at ConvQA benchmarking like QUAC [30] (text), CoQA [169] (text), SQA [83] (tables), CONVQUESTIONS [32] (KB), and CSQA [177] (KB) assume a single answering source. Rather than the easier option of artificially augmenting any of these with heterogeneous inputs, we believe that it is much more natural and worthwhile to build a new resource from scratch by users browsing through a mixture of sources, as they would do in a typical information seeking session on the web.

3.3.2 Construction of the CONVMIX Benchmark

Initiating conversations. How a user initiates a conversation is a key conceptual challenge that needs to be overcome in creating good ConvQA benchmarks. One could, for example, provide users with passages or documents, and ask them to create a sequence of questions from there [30, 169]. Alternatively, one could also provide annotators with some conversation from a benchmark so far, and request their continuation in some fashion [100]. Large-scale synthetic benchmarks would try to automate this as far as possible using rules and templates [177]. In keeping with our philosophy of natural conversations, we asked users to start with an entity of their choice (instead of spoonfeeding them with one, which could be counterproductive if the user has no interest or knowledge about the provided entity). Real conversations between humans, or several search sessions, often start when users have queries about such seed or topical entities. With the first question initiating the conversation, we collected four follow-up questions (total of five turns) that build upon the ongoing conversational context. To avoid a skew in conversations on head entities (like *Harry Potter* and *Lionel Messi*), an ongoing “blocklist” was maintained –

3 Conversational Question Answering

Table 3.4: Representative conversations from the CONVMIX benchmark. The types of sources which can be used for answering are given in brackets.

	Books	Movies	Music	TV series	Soccer
q^1	<i>Who wrote Slaughterhouse-Five?</i>	<i>Who played Ron in the Harry Potter movies?</i>	<i>What was the last album recorded by the Beatles?</i>	<i>Who is the actor of Rick Grimes in The Walking Dead?</i>	<i>Which national team does Kylian Mbappé play soccer for?</i>
a^1	Kurt Vonnegut [KB, Text, Info]	Rupert Grint [KB, Text]	Let It Be [KB, Text, Table]	Andrew Lincoln [KB, Text, Table]	France football team [KB, Text, Info, Table]
q^2	<i>Which war is discussed in the book?</i>	<i>Who played Dumbledore?</i>	<i>Where was their last paying concert held?</i>	<i>What about Daryl Dixon?</i>	<i>How many goals did he score for his home country in 2018?</i>
a^2	World War II [KB, Text]	R. Harris, M. Gambon [Text, Table]	Candlestick Park [Text]	Norman Reedus [KB, Text, Table]	9 [Table]
q^3	<i>What year was it's first film adaptation released?</i>	<i>What's the run time for all the movies combined?</i>	<i>What year did they break up?</i>	<i>did he also play in Saturday night live?</i>	<i>place of his birth?</i>
a^3	1972 [KB, Text, Table, Info]	1179 minutes [KB, Info]	1970 [KB, Text, Info]	Yes [Text]	Paris [KB, Text, Info]
q^4	<i>Who directed it?</i>	<i>Who was the production designer for the films?</i>	<i>Who was their manager?</i>	<i>whom did he play?</i>	<i>award he got in 2017?</i>
a^4	George Roy Hill [KB, Text, Table, Info]	Stuart Craig [KB, Text, Table]	Brian Epstein [KB, Text]	Daryl Dixon [Text]	Golden Boy [KB, Table]
q^5	<i>What was the final film that he made?</i>	<i>Which movie did he win an award for working on in 1980?</i>	<i>What was their nickname?</i>	<i>production company of the series?</i>	<i>Who is the award conferred by?</i>
a^5	Funny Farm [KB, Text, Table]	The Elephant Man [Text]	Fab Four [KB, Text]	NBC Studios [KB, Text, Info]	Tuttosport [KB, Text, Info]

whenever an annotator used an entity, this entity was added to the blacklist. The seed entity would then be unavailable to all future annotators. This also ensured that we had quite a few rare long-tail entities in our benchmark (see Table 3.3), as users were forced to think of alternatives.

Quality control. The study was conducted on the popular crowdsourcing platform Amazon Mechanical Turk (AMT), where we allowed only Master Workers to participate, for quality assurance. We also blocked single and sets of annotators who demonstrated evidence of excessive repetition or collusion in their annotations. Since the task is non-trivial for an average Turker (requires understanding of factoid questions, and familiarity of knowledge sources like Wikidata and Wikipedia, along with entities and literal answers), we also included quite a few internal checks and warnings that prompted users for unintentional mistakes before task submission. Workers with notably diverse and interesting conversations were awarded with a 5 USD bonus. Interestingly, several Turkers providing high-quality conversations found the task engaging (we provided a free-text feedback box), and submitted more than 20 HITs (human intelligence tasks; one HIT corresponds to the submission of five conversations). We conducted semi-automatic post-processing, validation and cleaning of the benchmark. Several issues were also resolved by meticulous manual inspection. For example, we ran CLOCQ [31] on the initial questions, and manually inspected cases for which no answering evidence was found, to identify cases in which the initial questions themselves were unanswerable. Such cases are specifically problematic, because the whole conversation might become unanswerable. Complete AMT guidelines are released as supplementary material².

Ensuring heterogeneity. Last but not the least, ensuring answer coverage over heterogeneous sources was a key concern. Here, we again kept it natural, and encouraged users not to forcibly stick to any particular source during their conversation. Interestingly, out of 3,000 conversations, only 374 used exactly one source. A majority (1,280) touched three sources, 572 touched four,

²<https://qa.mpi-inf.mpg.de/convinse/user-study.html>

while 774 used two inputs. Finally, note that this is only the source that the annotator used during her search process: it is quite possible that the answer can be located in other information sources (see the field [.] below answers in Table 3.4), thereby enabling future benchmark users to exploit answer redundancy.

Collecting longer conversations. We initially collected 2,800 conversations with five turns (referred to as CONVMIX-5T). However, there can also be cases in which users wish to dive deep into a specific topic, or other curiosities arise as the conversation continues. In such situations, conversations can easily go beyond five turns, making the understanding of the conversational flow even more challenging for the ConvQA system. Therefore, we collected 200 additional conversations with ten turns (denoted CONVMIX-10T), to test the generalizability of ConvQA systems over longer conversations. On manual investigation, we found that there are naturally more topic drifts within these conversations. These 2K (200×10) questions are only used for testing (and not for training/development) and serve as a robustness check. Thus, our complete benchmark CONVMIX (3,000 conversations in total) is made up of the two subsets CONVMIX-5T (2,800 conversations, 5 turns each) and CONVMIX-10T (200 conversations, 10 turns each).

Collected fields. We collected the following annotations from crowdworkers:

- (i) conversational questions,
- (ii) intent-explicit versions of follow-up questions,
- (iii) gold answers as plain texts and Wikidata URLs,
- (iv) question entities,
- (v) question paraphrases, and
- (vi) sources used for answer retrieval.

We believe that this additional metadata will make our resource useful beyond QA (in question rewriting and paraphrasing, for example). Most questions had exactly one correct answer, with the maximum being six. Table 3.3 summarizes notable properties of our study and benchmark, and Table 3.4 reports interesting representative examples. Note that HIT-specific entries, like the payment per HIT, are given for the collection of conversations with five turns. The respective numbers were doubled for the collection of conversations with ten turns.

3.4 Experimental Setup

We conduct all experiments on the CONVMIX benchmark. We split the part of CONVMIX with five turns, CONVMIX-5T, into train, development and test sets with the ratio 60:20:20. CONVMIX-10T is only used for testing purposes.

3.4.1 Heterogeneous Sources

CONVINSE and all baselines run on the same data collections. As our knowledge base, we take a complete NTriples dump of Wikidata (31 January 2022). It consists of about 17B triples, consuming about 2 TB disk space. We access the KB via our CLOCQ [31] interface³, that reduces the memory overhead and efficiently returns KB-facts for queried entities. The text collection is chosen to be the English Wikipedia (April 2022). The benchmark-relevant subset of Wikipedia is comprised of the pages of entities detected via CLOCQ. Documents are split into sentences using spaCy⁴. All tables and infoboxes originating from the retrieved Wikipedia

³A public API is available at <https://clocq.mpi-inf.mpg.de>.

⁴<https://spacy.io/api/sentencizer>

pages together constitute the respective answering sources. We parse Wikipedia tables using WikiTables⁵, and concatenate entries in the obtained JSON-dictionary for verbalization. This procedure also includes conversions for tables with nested structure. Infoboxes are detected using BeautifulSoup⁶.

3.4.2 Baselines

There are no prior works for ConvQA over heterogeneous sources. Thus, to compare the proposed CONVINSE pipeline with alternative choices, we adapt state-of-the-art question understanding (in this case, rewriting and resolution) methods from the IR and NLP literature. These serve as competitors for our SR generation phase. We then provide these baselines with exactly the same ERS and HA phases that CONVINSE has, to complete end-to-end QA pipelines.

Prepending history turns. Adding turns from the history to the beginning of the current question is still considered a simple yet tough-to-beat baseline in almost all ConvQA tasks [32, 100, 163, 214], and so we investigate the same here as well. Specifically, we consider four variants:

- (i) adding only the initial turn $\langle q^1, a^1 \rangle$, as it often establishes the topic of the conversation (**Prepend init**),
- (ii) adding only the previous turn $\langle q^{i-1}, a^{i-1} \rangle$, as it sets immediate context for the current information need (**Prepend prev**),
- (iii) adding both initial and previous turns (**Prepend init&prev**), and
- (iv) adding all turns $\{\langle q^t, a^t \rangle\}_{t=1}^i$ (**Prepend all**).

Question rewriting. We choose a T5-based rewriting model [167] (**Q. Rewriting**). The model is trained on the CANARD question rewriting benchmark [52], and we further fine-tune it on CONVMIX. For this, we use the $\langle \text{full history, current question} \rangle$ -pairs as input, and the respective completed questions (available in CONVMIX) as the gold label.

Question resolution. We use QURETEC [219] as baseline for question resolution (**Q. Resolution**), treating context disambiguation as a term classification problem. A BERT-encoder is augmented with a token-level classification head, and predicts for each history term whether the word should be added to the current question. The same distant supervision strategy (Section 3.2.4) as used by CONVINSE, is employed for deriving data to train QURETEC on CONVMIX.

3.4.3 Metrics

Measuring retrieval effectiveness. To evaluate retrieval quality, we use **answer presence** as our metric. It is a binary measure of whether one of the gold answers is present in the top- k evidence snippets ($k=100$ in all experiments).

Measuring answering effectiveness. We use one of the standard metrics for factoid QA [175], *precision at 1* (**P@1**), since FID generates a unique answer. FID generates plain strings as answers, and evaluation for such strings with exact match can often be problematic [192], since the correct answer could be expressed in different ways (e.g., $\{\text{“Eddard Stark”, “Ned Stark”}\}$, or $\{\text{“11 June 1969”, “11-06-1969”, “June 11, 1969”}\}$). Therefore, we try to normalize the answer to the KB, whenever possible, to allow for a fair comparison across systems. We search through $\langle \text{entity mention, KB-item} \rangle$ pairs coming from the evidence retrieval phase (Section 3.2.2). If

⁵<https://github.com/bcicen/wikitable>

⁶<https://beautiful-soup-4.readthedocs.io/en/latest/>

there is a perfect match between the entity mention and the predicted answer string, we return the corresponding KB-item as the answer. If there is no such perfect match, we compute the Levenshtein distance between the predicted answer and entity mentions from (entity mention, KB-item) pairs. The KB-item for the entity mention with the smallest edit distance is used as the answer in such cases. Note that such KB-items may also be normalized strings, dates, years, or numbers. These normalized KB-items are compared to the gold answers in the benchmark for the computation of P@1.

3.4.4 Configurations

CONVINSE uses a fine-tuned BART⁷ model for generating structured representations within its QU stage. The default hyperparameters from the Hugging Face library were used. The maximum sequence length was set to 20, and early stopping was enabled. Three epochs were used during training, with a batch size of ten. 500 warm-up steps with a weight decay of 0.01 turned out to be the most effective. CLOCQ, that was used for evidence retrieval inside CONVINSE and all baselines, has two parameters: k (number of disambiguations to consider for each question word), and p (a pruning threshold). In this work, we set k =Auto (CLOCQ dynamically sets the number of disambiguations), and p =1000, as these performed the best on our dev set. We used a Python implementation of BM25, with default parameters⁸. Code for FID is publicly available⁹. FID was trained on CONVMIX for its use in this work, with evidence from all sources as input. We use this same model for all source combinations (including single input sources), for demonstrating the robustness of CONVINSE. The number of input passages (k in this paper) was retained at 100 as in the original work [85]. The maximum token length of an answer was set to 10. A learning rate of 5×10^{-5} proved effective, with a weight decay of 0.01 and AdamW as the optimizer. All methods were trained on the CONVMIX train set, and all hyperparameters were tuned on the dev set. The code was written in Python, making use of the popular PyTorch library¹⁰. Whenever a neural model was used, code was run on a single GPU (NVIDIA Quadro RTX 8000, 48 GB GDDR6).

3.5 Experimental Results and Insights

We run CONVINSE and the baselines on the test set of CONVMIX (the test set of CONVMIX-5T, and CONVMIX-10T), and report results in Tables 3.5 and 3.6. We measure the performance at a question-level, and all metrics are micro-averaged. Throughout this section, best performing variants in columns are marked in **bold**. An asterisk (*) denotes statistical significance of CONVINSE over the nearest baseline. The McNemar’s test was performed for binary variables like P@1, and the paired t -test otherwise, with $p < 0.05$. All results are reported on the test set, except the ablation study, that was naturally conducted on the dev set. If not stated otherwise, we make use of *gold* answers for the previous turns in the conversation. For example, for answering q^4 we assume gold answers $a^0 - a^3$ to be known.

3.5.1 Key Findings

CONVINSE is viable for heterogeneous QA. The first and foremost takeaway is that our proposed pipeline is a viable approach for handling incomplete questions in conversations, given

⁷<https://huggingface.co/facebook/bart-base>

⁸<https://pypi.org/project/rank-bm25/>

⁹<https://github.com/facebookresearch/FiD>

¹⁰<https://pytorch.org>

3 Conversational Question Answering

Table 3.5: Comparison of answer presence within top-100 retrieved evidence snippets after QU + ERS on the CONVMIX test set.

QU + ERS ↓	KB	Text	Table	Info	KB + Text	KB + Table	KB + Info	Text + Table	Text + Info	Table + Info	All
Prepend init + BM25	0.380	0.298	0.120	0.331	0.415	0.386	0.406	0.297	0.329	0.331	0.419
Prepend prev + BM25	0.342	0.284	0.095	0.295	0.382	0.347	0.372	0.284	0.317	0.306	0.392
Prepend init&prev + BM25	0.440	0.366	0.137	0.420	0.486	0.443	0.479	0.359	0.407	0.409	0.495
Prepend all + BM25	0.431	0.367	0.148	0.430	0.476	0.437	0.468	0.361	0.411	0.419	0.482
Q. Resolution [219] + BM25 + FID	0.414	0.311	0.115	0.329	0.445	0.419	0.437	0.312	0.356	0.341	0.453
Q. Rewriting [167] + BM25	0.434	0.315	0.114	0.347	0.460	0.435	0.461	0.319	0.362	0.336	0.465
CONVINSE (Proposed)	0.475*	0.352	0.117	0.369	0.528*	0.486*	0.507*	0.353	0.408	0.381	0.542*

Table 3.6: Comparison of end-to-end (QU + ERS + HA) answering performance (P@1) on the CONVMIX test set.

ConvQA method ↓	KB	Text	Table	Info	KB + Text	KB + Table	KB + Info	Text + Table	Text + Info	Table + Info	All
Prepend init + BM25 + FID	0.211	0.174	0.065	0.200	0.246	0.211	0.240	0.174	0.203	0.195	0.254
Prepend prev + BM25 + FID	0.179	0.190	0.052	0.212	0.238	0.184	0.233	0.185	0.224	0.211	0.257
Prepend init&prev + BM25 + FID	0.234	0.233	0.074	0.276	0.290	0.238	0.292	0.229	0.274	0.272	0.312
Prepend all + BM25 + FID	0.230	0.234	0.074	0.282	0.290	0.238	0.282	0.224	0.267	0.265	0.300
Q. Resolution [219] + BM25 + FID	0.222	0.190	0.063	0.219	0.261	0.227	0.257	0.185	0.241	0.221	0.282
Q. Rewriting [167] + BM25 + FID	0.216	0.183	0.062	0.219	0.252	0.221	0.261	0.187	0.227	0.223	0.271
CONVINSE (Proposed)	0.251*	0.220	0.062	0.258	0.317*	0.257*	0.310*	0.220	0.276	0.253	0.342*

heterogeneous input sources. CONVINSE and most of the baselines consistently reach 40-50% on answer presence in their evidence pools after QU and ERS, with CONVINSE leading with 54% (see Table 3.5). These numbers set the upper bounds for end-to-end QA after the answering phase, which are in the ballpark of 25-34% (see Table 3.6).

It is noteworthy that even the basic prepending baselines, despite generating fairly verbose question formulations (15-26 words, Table 3.9) have very good answer presence within the top-100 evidence snippets. This is largely due to the CLOCQ entity-based retrieval module, which turned out to be quite robust even for long queries. Subsequently, BM25 scoring serves as a necessary filter, to prune the evidence sets. Sizes of these evidence sets varied from 2.3K (CONVINSE) to 7.5K (Prepend all), which would have posed efficiency challenges to the final answering stage had the BM25 filter not been applied.

CONVINSE outperforms baselines. We observe that the SRs in CONVINSE are significantly more effective than question rewriting/resolution and the prepend baselines. SRs provide the right balance between conciseness and coverage. Conciseness (SRs are just 6-7 words on average, Table 3.9) helps effective scoring with BM25 at the ERS stage, while expressive representation

Table 3.7: P@1 of ConvQA systems when using the *predicted answers* for the previous turns in the ongoing conversation.

Method ↓ (+ BM25 + FID)	CONVMix-5T (2800 questions)	CONVMix-10T (2000 questions)	CONVMix (4800 questions)
Prepend init	0.276	0.178	0.235
Prepend prev	0.190	0.123	0.162
Prepend init&prev	0.277	0.195	0.243
Prepend all	0.284	0.168	0.236
Q. Resolution [219]	0.283	0.188	0.243
Q. Rewriting [167]	0.258	0.168	0.221
CONVINSE	0.321*	0.217*	0.278*

of the conversational context is crucial, too. The expressivity in the SRs helps achieve the highest answer presence after QU and ERS (0.542 in Table 3.5), outperforming all baselines by substantial margins. This advantage is carried forward to the answering stage and results in the best end-to-end QA performance (0.342 in Table 3.6).

The simple prepend baselines sometimes come surprisingly close, though, and perform better than the sophisticated rewriting or resolution methods. Nevertheless, it is noteworthy that SRs and question rewriting/resolution have clear advantages in terms of interpretability. When a user wonders what went wrong upon receiving an incorrect (if not outright embarrassing) answer, the system could show the CFG, SRs or the inferred complete questions as informative explanations – helping the user to better understand and cope with the system’s limitations and strengths.

Altogether, the absolute P@1 numbers still leave huge room for improvement. This shows that CONVMix is indeed a challenging benchmark, with difficult user inputs where inferring the intent is hard, for example: “Which war is discussed in the book?” or “What was the final film he made?” (see Table 3.4 for more examples).

Combining heterogeneous sources helps. Another across-the-board observation is that combining knowledge sources is a strong asset for ConvQA. Consider the values in the “All” columns of Tables 3.5 and 3.6. These numbers are systematically and substantially higher than those in the columns for individual source types and even for pair-wise combinations.

To inspect whether these gains are not only from enhanced coverage, but also leverage *redundancy of information* across source types, we measured the average P@1 for all cases where the questions had 1, 2, 3, or 4 evidence snippets (among the top- k) containing the gold answer in the output of the ERS stage. The P@1 improves steadily as the answer can be found several times, i.e. as information becomes redundant, being 0.428 for one, 0.658 for two, 0.713 for three, and 0.763 across instances with four answering evidence snippets.

In general, tables prove to be the hardest to mine, within a unified framework, with relatively low answer presence and P@1 when used in separation (see Table 3.6).

CONVINSE excels in realistic setup with *predicted answers*. The experiments so far are conducted assuming the gold answers for the previous turns in the conversation to be given. However, in a realistic setup, the ConvQA system would not know these answers. Therefore, we conducted an additional experiment, in which we used the *predicted answers* by the system for the previous turns, when generating the outputs of the QU phase. The results of this experiment are shown in Table 3.7. CONVINSE outperforms all methods significantly on conversations of length both five and ten, even though it has never seen such 10-length conversations during

Table 3.8: P@1 of ConvQA systems over turns.

Method (+ BM25 + FID) ↓	CONVMIX-5T		CONVMIX-10T			
	1	2–5	1	2–4	5–7	8–10
Prepend init	0.388	0.271	0.320	0.235	0.185	0.128
Prepend prev	0.371	0.243	0.325	0.243	0.232	0.218
Prepend init&prev	0.370	0.315	0.335	0.310	0.308	0.245
Prepend all	0.375	0.323	0.305	0.278	0.270	0.195
Q. Resolution [219]	0.391	0.300	0.315	0.232	0.213	0.220
Q. Rewriting [167]	0.366	0.280	0.295	0.220	0.245	0.215
CONVINSE	0.395	0.368*	0.320	0.298	0.338*	0.252

Table 3.9: Average QU output length in words on test sets.

QU method ↓	CONVMIX-5T	CONVMIX-10T	CONVMIX
Original question	5.73	5.34	5.57
Prepend init	14.39	14.84	14.58
Prepend prev	12.23	12.17	12.21
Prepend init&prev	18.73	20.61	19.52
Prepend all	23.03	40.70	30.39
Q. Resolution [219]	8.24	8.54	8.36
Q. Rewriting [167]	9.07	9.04	9.06
CONVINSE	6.43*	6.53*	6.48*

training. Another observation is that the performance for the “*Prepend all*” baseline drops for CONVMIX-10T, which might be due to the much longer QU outputs generated as the conversation continues longer. In general, the performance of all methods drops a bit (0.057 P@1 on average, cf. Tables 3.6 and 3.7) when changing from gold answers to predicted answers.

3.5.2 Analysis

CONVINSE is stable over turns. One striking finding when drilling down into the results, is that the performance of CONVINSE stays fairly stable as the conversation continues – see Table 3.8. In contrast, as one would naturally expect, the baselines exhibit systematic degradation from turn to turn, as it becomes harder to capture implicit cues about the user intents with progressing conversation. This is most pronounced for the “*Prepend all*” model (per prepend baseline, a different FID model is trained leading to different performance even for turn 1). For most methods, there is a significant performance drop for the last three turns, indicating that further investigation of generalization to longer conversations might be worthwhile. This is a clear indication that naive models appear to be an easy choice overall, but quickly lose their effectiveness as the conversations become longer. This justifies more principled approaches as presented in CONVINSE.

SRs are compact. SRs are indeed succinct representations of user intents, as indicated by Table 3.9 on question lengths. Also, for interpretability, SRs are an easy-to-inspect gist that are likely to be comprehensible for end users and at the same time amenable for use with most standard IR models. Notably, for the more sophisticated models, the output length of the QU phase is almost stable on the two test sets with five turns and ten turns.

Table 3.10: Domain- and source-wise results of CONVINSE (P@1 metric).

Source ↓	Books	Movies	Music	TV Series	Soccer
KB	0.255	0.273	0.219	0.245	0.264
Text	0.226	0.229	0.244	0.234	0.165
Table	0.021	0.106	0.052	0.058	0.073
Info	0.282	0.272	0.226	0.265	0.243
All	0.329	0.357	0.353	0.338	0.333

Table 3.11: Ablation study of the CONVINSE SR.

Method ↓	Answer presence (ERS)	P@1 (HA)
CONVINSE	0.559	0.371
w/o Context entity slot	0.546	0.362
w/o Question entity slot	0.078	0.054
w/o Predicate slot	0.421	0.176
w/o Answer type slot	0.572	0.350
w/o Ordering	0.560	0.361

CONVINSE is stable over domains. Zooming into the results over the five thematic domains in CONVMIX, we find that performance is relatively stable (working best for the movies domain) – see Table 3.10 columns. The same holds when contrasting this with distributions over source types (Table 3.10 rows). Infoboxes consistently provide knowledge easily harnessed, while tables turn out to be the trickiest to handle, with proper verbalizations being a likely issue [154, 207].

Slots in SR are vital. A systematic ablation study shows that each of the SR slots plays an important role – see Table 3.11. We blanked out the contents of the respective slots during retrieval, and proceeded with these weaker SRs to the answering phase. Question entities clearly are the most pivotal; answer types do not help much at retrieval time, but justify their importance during answering (a shared insight with respect to the expected answer types in many QA models [175]). We also examined the effect of our proposed ordering of the SR slots, by randomly permuting the order of slots (e.g., predicates first). As expected, there is hardly an effect during ERS (both CLOCQ and BM25 are word-order-agnostic), but ordering proves beneficial when generating answers from evidence using sequence-aware models like FID.

Error analysis. CONVINSE cannot answer the question correctly 65.8% of the time, arising from three error cases:

- (i) the evidence retriever cannot retrieve answering evidence (42.4%), which can be due to the QU phase missing important cues in the conversation, failures within the evidence retriever, or the information sources not containing necessary information for answering, which could be targeted by considering additional information sources, or improving the retrieval within the ERS stage;
- (ii) the evidence retriever finds at least one answering evidence snippet but none is among the top- k after evidence scoring (27.2%), calling for more informed scoring mechanisms; or
- (iii) FID fails to generate the correct answer (30.4%), which indicates that more sophisticated reasoning over the set of evidence snippets might be required.

Anecdotal results. For a better understanding of how our SRs look like when contrasted with rewritten and resolved questions, we provide a few representative examples in Table 3.12.

Table 3.12: Anecdotal examples for which only CONVINSE answered correctly.

Domain	Books
Original	country of origin?
Q. resolution	country of origin? expanse leviathan
Q. rewriting	What country was it taken from?
SR (ours)	<code>< Leviathan Wakes <i>Expanse</i> country origin sovereign state ></code>
Domain	Movies
Original	What actor portrayed Magneto?
Q. resolution	What actor portrayed Magneto? x-men movie
Q. rewriting	Which actor played the character Magneto in the X-Men movie?
SR (ours)	<code>< X-Men <i>Magneto</i> actor portrayed human ></code>
Domain	Music
Original	Her date of birth?
Q. resolution	Her date of birth? shakira
Q. rewriting	When was Shakira born?
SR (ours)	<code>< Waka Waka (This Time for Africa) <i>Shakira</i> date birth date ></code>
Domain	TV Series
Original	What TV series did he first appear on?
Q. resolution	What TV series did he first appear on? tv show appeared the shows george
Q. rewriting	What TV series did George Grizzard first appear on?
SR (ours)	<code>< _ <i>George Grizzard</i> tv series first appear television series ></code>
Domain	Soccer
Original	Who won?
Q. resolution	Who won? shakira
Q. rewriting	Who won the FIFA FIFA World Cup in France?
SR (ours)	<code>< _ <i>2010 FIFA World Cup</i> won national association football team ></code>

3.6 Demonstration

An interactive demo for CONVINSE is publicly available at <https://convinse.mpi-inf.mpg.de>. Figure 3.4 shows a screenshot of the demo interface, which enables asking conversational questions, with an ongoing conversational context. For each answer derivation, the demo shows the generated structured representation, applying the same color coding for the individual slots as in this chapter.

3.7 Related Work

Conversational question understanding. Understanding the user intent of questions in a conversational context is a key challenge in ConvQA [250].

Some methods for *ConvQA over text* use a *hard history selection* to obtain a question-relevant subset of the conversation history [99, 161, 163]. However, the more prevalent approach is to construct a question-aware encoding with attention on the conversational history (*soft history selection*) [27, 82, 164]. This is then used by a neural reader for predicting the answer in the given passage. While this works well in ConvQA over text, the approach does not easily generalize to the heterogeneous setting, since this would require carefully designed mechanisms for representing inputs from different sources in a shared latent space. It may seem that hard history selection can be generalized to the heterogeneous setting more easily, but we found that such an approach hurts end-to-end answering performance in preliminary experiments. We employ attention over conversation history in the QU phase, which can be seen as a soft history selection.

Demo

The screenshot shows a conversational interface with three interactions. Each interaction consists of a question in a blue box, an answer in a grey box, and a corresponding graph representation in a white box with a black border. The graph representations use colored text to show the relationship between entities and predicates.

Interaction 1:
Question: capital of France?
Answer: Paris
Graph: (_ | France | capital of | sovereign state)

Interaction 2:
Question: who was the mayor in 2001?
Answer: Bertrand Delanoë
Graph: (_ | Paris | who was the mayor in 2001 | human)

Interaction 3:
Question: when was he born?
Answer: 30 May 1950
Graph: (_ | Bertrand Delanoë | when was he born | date)

At the bottom, there is a text input field with the placeholder "Please enter your question here". Below the input field are three buttons: "Answer" (dark blue), "Clear Last" (grey), and "Clear All" (grey).

Figure 3.4: Screenshot of the CONVINSE demo.

In *ConvQA over KBs*, the conversational flow is typically captured in a context graph, either explicitly [32] using iterative expansion, or implicitly [68, 100, 137] by maintaining a set of context entities and predicates. The conversational KB history can be incorporated using encodings [68, 137, 190] or heuristic subsets of the history [100], similar to ConvQA over text. All existing methods consider only one information source for answering conversations, which limits their answer coverage. Moreover, such works often adopt source-specific modeling of the conversational flow, like KB subgraphs, and cannot easily be extended to capture the conversational flow in a heterogeneous setting.

Question completion. One line of work that suggests a more general approach to the ConvQA problem aims to create a self-contained question from the incomplete utterance, that can be answered by standalone QA systems [52, 167, 214, 219, 234]. Such approaches can either take the form of question rewriting, which generates a complete question from scratch [167, 214], or question resolution, which adds relevant terms from the conversational history to the question [219]. Question rewriting can entail unnecessary complexity since recent QA systems may not require the question to have perfect syntax or grammar [85, 93].

Question resolution [219] operates on the surface level, without aiming for grammatically correct questions, creating a completed question similar to a keyword query. However, a potential downside is that there is no structure in the completed form.

We show in experiments (Section 3.5.1) that both completed forms via question rewriting

and question resolution perform worse than our generated structured representations, which can be perceived as logical forms for matching with heterogeneous sources.

3.8 Conclusion

We presented CONVINSE, the first end-to-end methodology for answering conversational questions over heterogeneous sources, covering knowledge bases (KBs), text corpora, web tables, and Wikipedia infoboxes in a single pipeline. Our approach combines ideas from different communities and their methodological paradigms. The learning of intent-explicit structured representations (SRs) is inspired by semantic frames in symbolic AI, but leaves its constituents in surface form without normalization. For retrieving relevant evidence from heterogeneous sources, we follow the line of IR thinking. The design of our SRs is vital here: the SRs provide enough structure to enhance effectiveness, but are lightweight and can be directly applied to KBs, text, tables, and infoboxes alike. Finally, the last stage for answering adopts the NLP paradigm of QA over text, utilizing a Seq2seq LM for autoregressive answer generation.

Our CONVMIX benchmark with thousands of real-user conversations was found useful in follow-up work on ConvQA (e.g., [86, 101, 119]).

4

Explainable Question Answering

CONVINSE constructs an SR that explicitly captures the user’s information need. This provides the user with human-comprehensible feedback on *what* the system *understood*. The answer generation, which is *how* the information need is *answered*, is still a black-box for end users. Further, the answer generation neglects structural cues between evidence pieces that are useful for detecting the correct answer. To this end, we propose EXPLAIGNN, an end-to-end method for explainable ConvQA over heterogeneous sources. The major contribution within EXPLAIGNN is the answering stage, which constructs an evidence graph based on information from heterogeneous sources. This graph is then iteratively reduced in size via GNNs. The answer is predicted from a small set of evidence pieces only, enabling the user to investigate the answer correctness.

All code, data and a demo are available at <https://explaignn.mpi-inf.mpg.de>.

Contents

4.1	Introduction	40
4.2	The EXPLAIGNN Methodology	41
4.2.1	Question Understanding (QU)	41
4.2.2	Evidence Retrieval and Scoring (ERS)	42
4.2.3	Heterogeneous Answering (HA)	42
4.2.4	Iterative Graph Neural Networks	46
4.3	Experimental Setup	48
4.3.1	Benchmarks	48
4.3.2	Baselines	48
4.3.3	Metrics	48
4.3.4	Configurations	48
4.4	Experimental Results and Insights	49
4.4.1	Key Findings	49
4.4.2	Analysis	50
4.4.3	Further Insights	53
4.5	User Study on Explainability	55
4.6	Demonstration	58
4.7	Related Work	58
4.8	Conclusion	60

4.1 Introduction

Motivation. Existing methods for QA over the full range of heterogeneous sources (KB, text, *and* tables) rely on Seq2seq LMs or GLMs, that are provided with a question representation and verbalized evidence snippets for generating the answer [33, 41]. Such models have been shown to achieve high performance on a variety of tasks and datasets [115, 165].

However, this approach has two significant limitations:

- (i) Seq2seq LMs/GLMs are *not explainable*, as they only generate strings as outputs, making it infeasible for users to decide whether to trust the answer;
- (ii) these models require inputs to be cast into plain token sequences first.

For example, within CONVINSE, all evidence is linearized into a textual sequence as input to an answer generator [85]. This loses insightful information on relationships between evidence snippets (e.g., shared entities) [247]. Such inter-evidence connections can be helpful in separating relevant information from noise.

Ideally, end users should be able to *trace back* answers to the evidence used by the QA system, enhancing *explainability*, such that users can verify the answer correctness. For computing the answer, the QA system should account for *structural cues* by explicitly modeling relationships between evidence pieces, for improving its answering performance. At the same time, following the main motivation of this dissertation, the QA system should enable seamless integration of *heterogeneous sources* into the answering process.

The following conversation will be our running example in this chapter:

*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**

Approach. We introduce EXPLAIGNN (Explainable Conversational Question Answering over Heterogeneous Graphs via Iterative Graph Neural Networks), a flexible pipeline for ConvQA that can be configured for optimizing *performance, efficiency, and explainability*, and integrates heterogeneous sources. EXPLAIGNN follows the general 3-stage architecture introduced in CONVINSE (as illustrated in Figure 3.1), but greatly improves the final answering stage:

- (i) Derivation of a self-contained *structured representation* (SR) of the user’s information need from the potentially incomplete input utterance and the conversational context, making the central entities, relation, and expected answer type explicit.
- (ii) Retrieval of relevant evidence and answer candidates from heterogeneous information sources: a curated KB, a text corpus, a collection of web tables, and infoboxes.
- (iii) Construction of an evidence graph from these heterogeneous information snippets, as the basis for applying graph neural networks (GNNs). The GNNs are iteratively applied for computing the best answers and supporting evidence in a small number of steps.

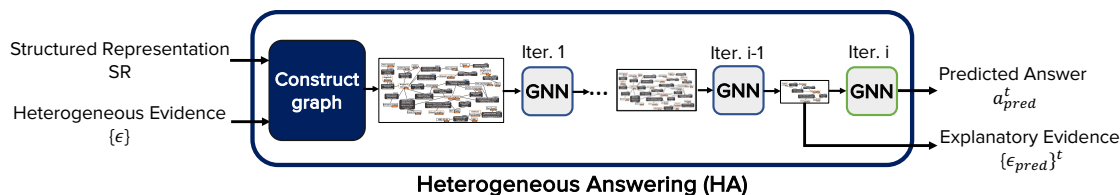


Figure 4.1: An overview of the answering stage of EXPLAIGNN, which constructs an evidence graph from heterogeneous sources, and then iteratively applies GNNs for computing the answer.

A key novelty is that each iteration reduces the evidence graph in size, and only the final iteration yields the answer and a small user-comprehensible set of explanatory evidence pieces. Our overarching goal is to provide end user *explainability* to *GNN* inference by *iteratively* reducing the graph size, so that the final answers can indeed be claimed to be causal with respect to the remaining evidence, hence the name EXPLAIGNN. The GNNs that instantiate the individual iterations leverage a novel attention mechanism, which ensures that only question-relevant information is spread within the local neighborhoods.

Finally, to investigate the explainability of the proposed pipeline, we conduct a user study. We find that users can tell the answer correctness with high accuracy, when provided only with the explanation computed by EXPLAIGNN, that is the SR and the evidence in the reduced graph.

Contributions. The salient contributions of this work are the following:

- Proposing EXPLAIGNN, a new method for ConvQA over heterogeneous sources, with a focus on computing answers that are explainable to end users.
- Devising a framework for iterative application of GNN inference to shrink evidence graphs until the answer and its explanatory evidence are obtained.
- Developing a novel attention mechanism which ensures that during message passing only question-relevant information is spread over the local neighborhoods.

4.2 The EXPLAIGNN Methodology

The architecture of EXPLAIGNN follows the pipeline of CONVINSE (cf. Figure 3.1):

- (i) the *question understanding* (QU) stage creates an intent-explicit SR,
- (ii) the *evidence retrieval and scoring* (ERS) stage retrieves evidence from heterogeneous sources,
- (iii) the final *heterogeneous answering* (HA) stage utilizes this set of relevant evidence pieces for answering the question and providing explanatory evidence.

The main contribution of this work is on the HA stage (iii), with targeted techniques to enhance the first two stages. The remainder of this section discusses the stages in more detail.

4.2.1 Question Understanding (QU)

Building upon our CONVINSE project, we aim to generate a structured representation (SR) capturing the complete intent of the current question in the conversational context. We leverage the same fine-tuned BART [115] model as in CONVINSE.

Preventing hallucination. We propose a mechanism to avoid *hallucinations* in SRs. Hallucination is a common problem of Seq2seq LMs (and also GLMs) [139]: occasionally, outputs are

generated that are completely unrelated to the given input. For q^3 of the running example, the trained model could generate *Robert Langer* as the (topically unrelated) question entity of the output SR $\langle \textit{Dan Brown} \mid \textit{Robert Langer} \mid \textit{who played him in the films} \mid \textit{human} \rangle$. This would lead the entire QA system astray, and could therefore confuse end users limiting their trust in the system’s predictions.

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 insight facilitates the detection of hallucinations: output words that are absent from the entire conversation so far, indicate such a situation. To fix such error cases, 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: answer types are often not made explicit (e.g., for questions starting with “*when...*” or “*where...*”), but can still substantially help the QA system [175]. So we remove this slot before performing the hallucination check.

4.2.2 Evidence Retrieval and Scoring (ERS)

For the ERS stage, we also follow the entity-centric retrieval pipeline of CONVINSE: evidence from the *KB* and entity disambiguations are obtained via running CLOCQ [31] on the SR. *Text, table and infobox evidence* is 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. All evidence is verbalized. Example evidence snippets can be seen within Figure 4.2, where each evidence is tagged with its source. BM25-scoring is used for identifying the top- k evidence snippets. Due to our efficient implementation of the answering stage, EXPLAIGNN allows for higher values for the k parameter than CONVINSE (500 or 1000 instead of 100).

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. Evidence is then only retrieved for this restricted set of entities. This prunes noisy disambiguations in the relation and type slots, which may otherwise appear due to CLOCQ’s unawareness of the SR structure.

4.2.3 Heterogeneous Answering (HA)

We first describe the construction of the heterogeneous evidence graph (Section 4.2.3.1). Next, we present our proposed question-aware GNN architecture, consisting of the encoder (Section 4.2.3.2), the message passing procedure (Section 4.2.3.3), the answer candidate scoring (Section 4.2.3.4), and the multi-task learning mechanism for GNN training (Section 4.2.3.5).

4.2.3.1 Graph Construction

Given the heterogeneous evidence retrieved in the previous stage, we construct an *evidence graph* that has two types of nodes: entity nodes and evidence nodes. 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 within ϵ . As a result, there are no direct edges between pairs of entity nodes, or pairs of evidence nodes. An example evidence graph is shown in Figure 4.2.

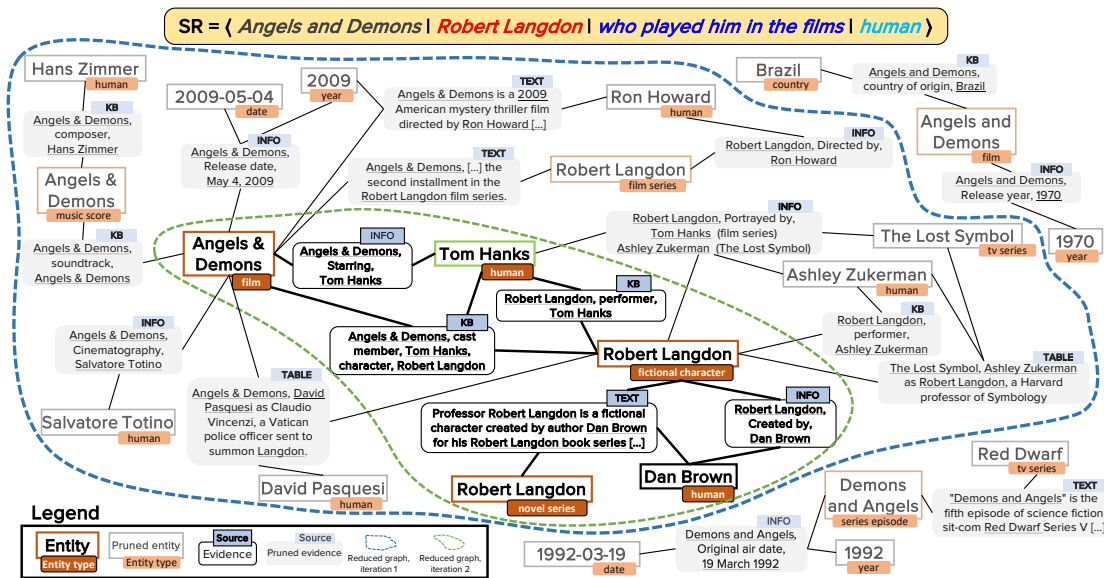


Figure 4.2: Toy evidence graph for answering q^3 , showing two pruning iterations. The graph is iteratively reduced by GNN inference to identify the key evidence snippets. The subgraph surrounded by the blue dotted line is the result of the first iteration, while the green dotted line indicates the graph after the second iteration. From this smaller evidence subgraph only, the final answer (Tom Hanks) is inferred.

Inducing connections between retrieved evidence pieces. Shared entities are the key elements that induce *relationships* between the initial plain *set* of relevant evidence snippets. So one requires entity markup on the verbalized evidence pieces coming from the different sources, with entities grounded to a KB for *canonicalization*. For this reason, during evidence verbalization, original formats are not discarded. Thus, for KB-facts, entity mappings are already known. For text sentences, table records, and infobox entries from Wikipedia, we link “href” anchor texts to their referenced entity pages. These are then mapped to their corresponding KB-entities. In absence of anchor texts, NERD [79, 117, 215] tools could be used. Dates and years that appear in evidence snippets are detected using regular expressions, and are added as entity nodes to the graph as well. In Figure 4.2, entity mentions are underlined within evidence nodes.

4.2.3.2 Node Encodings

GNNs incrementally update node encodings within local neighborhoods leveraging message passing. However, these node encodings have to be initialized first which is done using an ELM.

Evidence encodings. For the initial node encoding, we make use of cross-encodings [170]. The verbalized evidence snippet, concatenated with the SR, is fed into an encoder LM. For efficiency, we use a distilled ELM (such as DISTILBERT [179]). 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 node 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 Figure 4.2).
- (iii) For long-tail entities, the 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 $\mathbf{SR} \in \mathbb{R}^d$.

Note that the *same* ELM is used for the initial encodings of evidence nodes, entity nodes and the SR. The parameters of this shared language model are updated during GNN training, to ensure that the encoder adapts to the syntactic structure of the SR. In the remainder of this section, items in bold (e.g., \mathbf{SR}) indicate their encodings.

4.2.3.3 Message Passing

The core part of the GNN is the *message passing* [213, 243] 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 evidence nodes and the connected entities. Again, we aim to focus on question-relevant information, 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 represented by $\alpha_{\epsilon,e}^l \in \mathbb{R}$:

$$\alpha_{\epsilon,e}^l = \operatorname{softmax}_{\mathcal{N}(\epsilon)} \left(\operatorname{lin}_{\alpha_\epsilon}^l (e^{l-1}) \cdot \mathbf{SR} \right) = \frac{\operatorname{lin}_{\alpha_\epsilon}^l (e^{l-1}) \cdot \mathbf{SR}}{\sum_{e_i \in \mathcal{N}(\epsilon)} \operatorname{lin}_{\alpha_\epsilon}^l (e_i^{l-1}) \cdot \mathbf{SR}} \quad (4.1)$$

where we first project the entity encodings using a linear transformation ($\operatorname{lin}_{\alpha_\epsilon}^l : \mathbb{R}^d \rightarrow \mathbb{R}^d$), and then multiply with the SR encoding to obtain a score. The softmax function is applied over all entity nodes neighboring a respective evidence node ($e_i \in \mathcal{N}(\epsilon)$) for deriving scores in the range $[0, 1]$. Thus, an entity can obtain different SR-attention scores for each evidence node, 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:

$$\mathbf{m}_\epsilon^l = \operatorname{lin}_{m_\epsilon}^l \left(\sum_{e \in \mathcal{N}(\epsilon)} \alpha_{\epsilon,e}^l \cdot e^{l-1} \right) \quad (4.2)$$

where $\operatorname{lin}_{m_\epsilon}^l$ is the linear projection layer ($\operatorname{lin}_{m_\epsilon}^l : \mathbb{R}^d \rightarrow \mathbb{R}^d$).

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

$$\epsilon^l = \operatorname{ReLU}(\mathbf{m}_\epsilon^l + \epsilon^{l-1}) \quad (4.3)$$

The intuition here is that in each update of evidence nodes, the question-relevant information held by neighboring entity nodes is passed on to an evidence node, and then incorporated in its encoding. After multiple GNN layers, the information spreads further in the graph.

The process for updating the entity node encodings is analogous, but makes use of different weights in its linear transformation functions. The SR-attention $\alpha_{e,\epsilon}^l$ of evidence nodes for an entity node e is obtained as follows:

$$\alpha_{e,\epsilon}^l = \operatorname{softmax}_{\mathcal{N}(e)} \left(\operatorname{lin}_{\alpha_e}^l (\epsilon^{l-1}) \cdot \mathbf{SR} \right) \quad (4.4)$$

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

$$\mathbf{m}_e^l = \operatorname{lin}_{m_e}^l \left(\sum_{\epsilon \in \mathcal{N}(e)} \alpha_{e,\epsilon}^l \cdot \epsilon^{l-1} \right) \quad (4.5)$$

The updated entity node encoding is then given by:

$$\mathbf{e}^l = \operatorname{ReLU}(\mathbf{m}_e^l + \mathbf{e}^{l-1}) \quad (4.6)$$

These message passing steps are repeated L times, which implies that 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 own node encodings. Note that the shown equations are crafted for explanatory purposes. The implementation makes use of efficient matrix multiplications, leveraging the adjacency matrix instead of iterating through neighborhoods, and processes multiple instances within a batch.

4.2.3.4 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 [93, 197, 243], and compute an answer score for each entity node by considering their question relevance as captured within the node encodings. The computation of the answer score s_e is similar to our technique used for computing the SR-attention of an entity node:

$$s_e = \operatorname{softmax}_{V_e} \left(\operatorname{lin}_e (e^L) \cdot \mathbf{SR} \right) \quad (4.7)$$

We project the entity encoding using a linear layer ($\operatorname{lin}_e : \mathbb{R}^d \rightarrow \mathbb{R}^d$), and compute the dot product with the encoding of the SR. The softmax function is applied over all entity nodes in the graph (i.e., $e_i \in V_e$).

4.2.3.5 Multi-Task Learning

Our training data for the GNN consists of $\langle \text{graph}, \text{answer} \rangle$ 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 entity nodes in the graph but only few gold answers.

To better use our training data, we propose a *multi-task learning* (MTL) [109, 190] approach. Given a GNN, we pose two complementary node classification tasks: (i) the answer prediction, and (ii) the prediction of evidence snippet relevance.

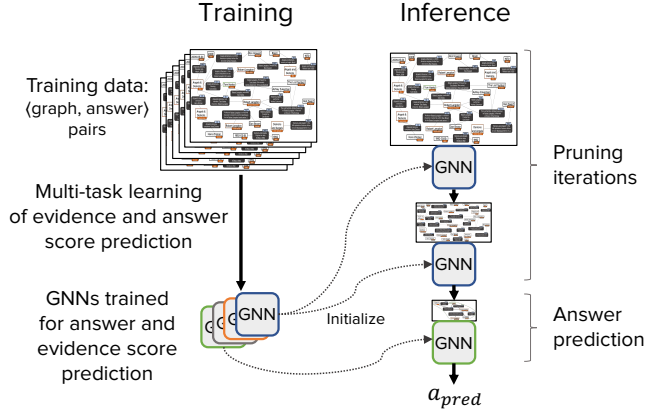


Figure 4.3: Training of and inference with our iterative GNNs.

For training, we utilize *distant supervision*: evidence pieces 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:

$$s_\epsilon = \text{softmax}_{\epsilon \in V_\epsilon} \left(\text{lin}_\epsilon(\epsilon^L) \cdot \mathbf{SR} \right) \quad (4.8)$$

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

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}_ϵ , respectively. The final loss used for training the GNN is then defined as a weighted sum:

$$\mathcal{L} = w_e \cdot \mathcal{L}_e + w_\epsilon \cdot \mathcal{L}_\epsilon \quad (4.9)$$

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

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

4.2.4 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 GNN inference has 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 [87]. Hence, *answer explainability* would be limited.

- (iii) Finally, obtaining cross-encodings for hundreds of nodes (entities and evidence pieces) can be computationally expensive, affecting the *run-time efficiency* of the system. A large fraction of these initial graph nodes might be rather irrelevant, which can often be identified using a more lightweight (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 scores s_e , to identify the most relevant evidence nodes. These evidence nodes, 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.

Figure 4.3 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 evidence nodes can be applied on a graph with 100, 20, or 5 evidence nodes (or even 1000). The GNNs essentially learn to spread question-relevant information within local neighborhoods, which is not only required for large graphs with hundreds of evidence nodes, but also for smaller graphs with a handful of nodes. Further, this iterative procedure is tractable only with the flexibility of scoring both entity nodes and evidence nodes 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 evidence from several hundreds of nodes, but may only need to identify the top-100 evidence nodes during our iterative 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 evidence graph with hundreds of nodes to the user is not practical: we can iteratively derive a small set of evidence nodes (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 *run-time 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 evidence nodes, weighted by their question-relevance (i.e., their SR-attention):

$$\mathbf{e}^0 = \sum_{\epsilon \in \mathcal{N}(\epsilon)} \alpha_{e,\epsilon}^0 \cdot \boldsymbol{\epsilon}^0 \quad (4.10)$$

where the SR-attention $\alpha_{e,\epsilon}^0$ is computed as in Equation 4.1, employing a different linear projection. This can be perceived as obtaining *alternating encodings* of entities: the initial evidence node encodings are used to initialize entity encodings (inspired by [22]). The message passing would then proceed as outlined in Section 4.2.3.3.

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 score 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:

$$a_{pred} = \arg \max_{e \in V_e} (s_e) \quad (4.11)$$

This is shown to the end user, together with the explanatory evidence $\{\epsilon_{pred}\}$ (see outputs in Figure 4.1), that are simply the set of evidence snippets used in the answer prediction step.

4.3 Experimental Setup

4.3.1 Benchmarks

We train and evaluate EXPLAIGNN on our CONVMIX (Section 3.3) benchmark. We also conduct a *robustness* test on our CONVQUESTIONS [32] benchmark, which is a popular dataset for ConvQA over KBs, and has 11,200 conversations with 5 turns each. In experiments on CONVQUESTIONS, we utilize EXPLAIGNN as trained on CONVMIX.

4.3.2 Baselines

We compare EXPLAIGNN against CONVINSE (Section 3.2). As additional comparison point, we enhanced the FID model within CONVINSE to generate top- k answer strings for improved metrics beyond the first rank. We also compare with other baselines proposed in the CONVINSE work, that are based on question completion and resolution.

4.3.3 Metrics

For accessing the answer performance, we use *precision at 1* (**P@1**), as in Section 3.4. 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** is the fraction of questions for which a gold answer is present in a given set of evidence snippets.

4.3.4 Configurations

The configuration of the QU and ERS stages follows the one for CONVINSE (Section 3.4).

The GNNs were implemented from scratch via PyTorch. We used DISTILROBERTA¹ as encoder, which we found to perform slightly better than DISTILBERT [179] in preliminary experiments. DISTILROBERTA has an embedding 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 evidence nodes was chosen for instantiating the pruning iterations (alternating encodings for entities, $w_e=0.3$, $w_e=0.7$), and the GNN obtaining

¹<https://huggingface.co/distilbert/distilroberta-base>

Table 4.1: Comparison of answering performance on the CONVMIX [33] test set, using *gold answers* $\{a_{gold}\}$ in the history.

Method ↓	P@1	MRR	Hit@5
Q. Resolution [219] + BM25 + FID [85]	0.282	0.289	0.297
Q. Rewriting [167] + BM25 + FID [85]	0.271	0.278	0.285
CONVINSE [33] (original)	0.342	0.365	0.386
CONVINSE [33] (top-k FID)	0.343	0.378	0.431
EXPLAIGNN (proposed)	0.406*	0.471*	0.561*

the highest P@1 for the answer prediction (cross-encodings for entities, $w_e=0.5$, $w_\epsilon=0.5$). In the ERS stage, we retain a maximum of $k=500$ evidence snippets after BM25 scoring, which means that our initial graph has 500 evidence nodes and the corresponding entity nodes. We restrict the number of entity nodes to 1500, which was sufficient in most cases, to enable batching. During distant supervision, when deriving relevance labels for evidence nodes, we prune out training instances, i.e. $\langle \text{graph}, \text{answer} \rangle$ -pairs, for which the distant supervision mechanism detects more than 10 positive evidence nodes (that are connected to the gold answer). Such cases often indicate spurious paths [100], which can have negative effects on the model training. The GNN for the pruning iterations was chosen by its answer presence among the top-5 evidence nodes, and the model trained for the final answer iteration was chosen by the P@1 metric.

A single GPU (NVIDIA Quadro RTX 8000, 48 GB GDDR6) was used to train and evaluate the models.

4.4 Experimental Results and Insights

4.4.1 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 4.1 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* for answer detection. As is clear from the method descriptions in Table 4.1, all baselines crucially rely on the generative reader model of FID. FID can ingest multiple evidence pieces to produce the answer, but fails to capture their relationships explicitly, that is unique to our graph-based pipeline. While our adaptation with top- k FID generations improved the ranking capabilities (MRR, Hit@5) of CONVINSE, EXPLAIGNN still substantially improves over this baseline on all metrics.

EXPLAIGNN is robust to wrong predictions in earlier turns. Unlike many existing works, we also evaluate the methods in a *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 4.2 (cf. Table 4.1 results, that show an evaluation with gold answers). While the performance of all methods drops in this

Table 4.2: 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 [219]+ BM25 + FiD [85]	0.243	0.250	0.257
Q. Rewriting [167]+ BM25 + FiD [85]	0.221	0.227	0.235
CONVINSE [33] (original)	0.278	0.286	0.294
CONVINSE [33] (top- k FiD)	0.279	0.308	0.351
EXPLAIGNN (proposed)	0.339*	0.398*	0.477*

Table 4.3: Effect of varying source combinations at inference time, on test set. EXPLAIGNN is still *trained* on all sources.

Sources ↓	P@1	MRR	Hit@5	Answer presence
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

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 4.3. The first takeaway is that the answering performance is the best when the full spectrum of sources is used. EXPLAIGNN can make use of the enhanced answer presence in this case, and answers 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.

4.4.2 Analysis

Multi-task learning enables flexibility. A systematic analysis of the effect of different multi-task learning weights is shown in Table 4.4. 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 Equation 4.10). The results indicate the run-time benefits of using alternating encodings for entities. Further, when optimized for evidence relevance prediction ($w_\epsilon=0.7$ or $w_\epsilon=1.0$), the GNN can maintain a high answer presence (measured within top-5 evidence nodes), indicating that lightweight 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 detecting the answer.

Table 4.4: Effect of varying the multi-task learning weights *when training* the one-shot GNN modules (measured on the CONVMIX dev set).

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

Table 4.5: Varying the number of iterations, pruning factors, and evidence nodes the answer is predicted from *during inference* (dev set).

EXPLAIGNN iterations ↓	P@1	MRR	Hit@5	Answer presence and number of evidence nodes after pruning iteration			Run-time (HA)
$i=1: 500 \rightarrow a_{pred}$	0.442	0.502	0.581	–	–	–	1,017 ms
$i=2: 500 \rightarrow 100 \rightarrow a_{pred}$	0.441	0.504	0.587	–	–	0.687 100	744 ms
$i=2: 500 \rightarrow 50 \rightarrow a_{pred}$	0.440	0.504	0.588	–	–	0.675 50	591 ms
$i=2: 500 \rightarrow 20 \rightarrow a_{pred}$	0.438	0.504	0.591	–	–	0.655 20	515 ms
$i=2: 500 \rightarrow 5 \rightarrow a_{pred}$	0.422	0.480	0.560	–	–	0.589 5	459 ms
$i=3: 500 \rightarrow 200 \rightarrow 100 \rightarrow a_{pred}$	0.441	0.504	0.585	–	0.694 200	0.685 100	995 ms
$i=3: 500 \rightarrow 150 \rightarrow 50 \rightarrow a_{pred}$	0.441	0.505	0.586	–	0.693 150	0.678 50	741 ms
$i=3: 500 \rightarrow 100 \rightarrow 20 \rightarrow a_{pred}$ (proposed)	0.442	0.505	0.589	–	0.687 100	0.654 20	601 ms
$i=3: 500 \rightarrow 50 \rightarrow 5 \rightarrow a_{pred}$	0.419	0.475	0.556	–	0.675 50	0.579 5	511 ms
$i=4: 500 \rightarrow 300 \rightarrow 150 \rightarrow 100 \rightarrow a_{pred}$	0.441	0.504	0.587	0.696 300	0.691 150	0.686 100	1,232 ms
$i=4: 500 \rightarrow 200 \rightarrow 100 \rightarrow 50 \rightarrow a_{pred}$	0.440	0.504	0.585	0.694 200	0.685 100	0.677 50	945 ms
$i=4: 500 \rightarrow 200 \rightarrow 50 \rightarrow 20 \rightarrow a_{pred}$	0.436	0.500	0.584	0.694 200	0.677 50	0.652 20	769 ms
$i=4: 500 \rightarrow 100 \rightarrow 20 \rightarrow 5 \rightarrow a_{pred}$	0.422	0.476	0.553	0.687 100	0.654 20	0.575 5	577 ms

Iterative GNNs do not compromise run-times. Table 4.5 reports results of varying the number of iterations $i \in \{1, 2, 3, 4\}$, and the graph size in the number of evidence nodes the answer is predicted from ($|\mathcal{V}_e| \in \{500, 100, 50, 20, 5\}$). For each row, the reduction in graph size, in terms of the number of evidence nodes considered, was kept roughly constant for consistency. First of all, by our smart use of alternating encodings for entities in the pruning iterations, *run-times remain robust* to the number of pruning iterations (run-times for $i=4$ are not necessarily higher than those for $i=3$, and so on). Rather, the run-time, measured in milliseconds (ms), is primarily influenced by the size of the graph (no. of evidence nodes) given to the final answer prediction step (compare run-times 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* of EXPLAIGNN 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 evidence nodes, they can be applied to score nodes in graphs of variable sizes.

Table 4.6: Results for out-of-the-box EXPLAIGNN, without further training or fine-tuning, on the CONVQUESTIONS [32] benchmark.

Method ↓	P@1	MRR	Hit@5
CONVEX [32]	0.184	0.200	0.219
FOCAL ENTITY [112]	0.248	0.248	0.248
OAT [137]	0.166	0.175	–
OAT [137] (utilizes gold entity annotations)	0.250	0.260	–
CONQUER [100]	0.240	0.279	0.329
PRALINE [98]	0.294	0.373	0.464
KRR [107] (utilizes gold entity annotations)	0.397	0.397	0.397
EXPLAIGNN (KB-only)	0.330 [‡]	0.399[‡]	0.480^{†‡}
EXPLAIGNN	0.363 [‡]	0.447^{†‡}	0.546^{†‡}

Table 4.7: 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 [†]

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 [32] dataset. CONVQUESTIONS is a competitive benchmark for ConvQA over KBs. We test EXPLAIGNN in two different modes: (i) using only facts from the KB (to match the setting of baselines), and (ii) using evidence from all information sources (for enhanced answer coverage). Table 4.6 shows the results ([†] and [‡] indicate statistical significance over the leaderboard toppers KRR [107] and PRALINE [98], 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 improves the answer performance substantially, even though CONVQUESTIONS was collected primarily for KBs.

Iterative GNNs improve robustness. In Section 4.2.3, 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 4.5), 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 KB-only mode (P@1: 0.330 versus 0.281) and with heterogeneous sources (P@1: 0.363 versus 0.318).

SR-attention, cross-encoding and entity types are crucial. Table 4.7 shows the 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 (Section 4.2.3.3), 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 (Section 4.2.3.2) that initialize the nodes

Table 4.8: Comparison of EXPLAIGNN variants run/trained on graphs of different sizes (measured by the number of evidence nodes), on the CONVMIX test set.

Method ↓	P@1	MRR	Hit@5	Run-time (HA)
EXPLAIGNN (train: 500, inference: 1000)	0.422	0.493	0.585	1,181 ms
EXPLAIGNN (train: 500, inference: 500)	0.417	0.489	0.582	676 ms
EXPLAIGNN (train: 500, inference: 100)	0.406	0.478	0.568	175 ms
EXPLAIGNN (train: 100, inference: 1000)	0.418	0.490	0.580	1,157 ms
EXPLAIGNN (train: 100, inference: 500)	0.417	0.488	0.579	669 ms
EXPLAIGNN (train: 100, inference: 100)	0.411	0.481	0.573	175 ms

with question-relevant encodings are vital. Also notable is the crucial role of entity types (Section 4.2.3.2), that help identify irrelevant answer candidates with mismatched answer types.

Our enhancements for the QU and ERS stages also contribute to the answering performance of EXPLAIGNN. The use of SR slots during retrieval (Section 4.2.2) has significant impact on all metrics. The hallucination prevention (Section 4.2.1) also helps to avoid simple mistakes that could reduce users’ trust in the system.

Error analysis. We identified three key sources of error:

- (i) the answer is not present in the initial graph (53.9% of error cases),
- (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%).

Notably, 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.

4.4.3 Further Insights

For the following experiments, we used a slightly improved version of the EXPLAIGNN code (the version that is publicly available), which led to small improvements of results. For example, the P@1 metric on the CONVMIX test set is improved from 0.406 (Table 4.1) to 0.417 (Table 4.8).

Varying the evidence graph size within EXPLAIGNN. For our main experiments, we trained all GNNs from graphs with up to 500 evidence nodes. Our GNNs are designed to be agnostic to the input size (at least theoretically; in practice the size is still bounded by computational cost). This means that the same GNNs can be successfully applied on graphs of *smaller* size than seen during training. In this experiment, we test the effect of running the GNNs on graphs that are *larger* than seen during training.

The results for this analysis are shown in Table 4.8. Notably, our GNNs show promising generalizability capabilities when applied to larger graphs: even when trained only on graphs with 100 evidence nodes, the GNNs can be successfully applied on graphs with 1,000 evidence nodes during inference. Importantly, the graph size has direct efficiency implications: the larger the graphs, the higher the computational costs. For example, training the GNN on the CONVMIX train set for 5 epochs with 100 evidence nodes took 20 minutes (alternating encodings for entities) / 70 minutes (cross-encodings for entities). With 500 evidence nodes, the training time is increased to 90 minutes (alternating encodings) / 210 minutes (cross-encodings). Overall, GNNs trained on graphs with only 100 evidence nodes match the performance of GNNs trained on graphs with 500 evidence nodes: this opens up an interesting avenue to cut training costs.

Table 4.9: Effect of varying the k parameter of CLOCQ during training/inference.

Method ↓	P@1	MRR	Hit@5
EXPLAIGNN	0.417	0.489	0.582
EXPLAIGNN (CLOCQ: $k=5$ for train, $k=5$ for test)	0.427	0.498	0.592
EXPLAIGNN (CLOCQ: $k=5$ for train, $k=10$ for test)	0.426	0.498	0.593

Table 4.10: Effect of varying the maximum number of positive evidence pieces (as identified by distant supervision) during training. The results are reported on the CONVMIX dev set.

Method ↓	P@1	MRR	Hit@5	Run-time (HA)
EXPLAIGNN (all pos. evidence nodes)	0.439	0.508	0.594	176 ms
EXPLAIGNN (max. pos. evidence nodes: 20)	0.444	0.505	0.587	183 ms
EXPLAIGNN (max. pos. evidence nodes: 10)	0.447	0.509	0.587	170 ms
EXPLAIGNN (max. pos. evidence nodes: 8)	0.434	0.501	0.586	172 ms
EXPLAIGNN (max. pos. evidence nodes: 5)	0.425	0.493	0.583	170 ms

Increasing the number of CLOCQ-disambiguations. During evidence retrieval, we utilize CLOCQ [31] to disambiguate entities in the SR. Evidence is then retrieved for this set of entities. However, entity disambiguation is prone to errors. As a consequence, CLOCQ provides top- k disambiguations for each mention, to enhance recall of correct disambiguations. Under the default setting (used in our main experiments), CLOCQ sets this parameter k dynamically to a value between 1 and 5, for each mention individually, based on its uncertainty in disambiguating the specific mention. This mechanism aims to balance recall and precision. In this experiment, we test the effect of setting a fixed k of 5 or 10, which increases the number of disambiguated entities, and thus also the number of evidence snippets that are retrieved.

Table 4.9 shows the results. Interestingly, EXPLAIGNN can further improve its answering performance with a fixed k in CLOCQ of 5. Increasing the CLOCQ- k to 10 does not yield further gains, but does also not lead to a performance decrease. Notably, EXPLAIGNN successfully leverages the increased answer presence in the graphs, without being distracted by the incorrectly linked entities that are inevitable with 5 or 10 disambiguations per mention.

Varying the upper bound for positive evidence snippets. During our distant supervision mechanism for identifying relevance labels of evidence nodes, training instances with more than 10 evidence nodes connected to the gold answer are excluded. This aims to prune out instances with spurious paths [100] from the training data. During inference, all instances are retained. Table 4.10 shows the effect of different choices for this upper bound, including variants that do not prune out such training instances. Models are trained and run on graphs with 100 evidence nodes (for efficiency). As expected, the answering performance drops when utilizing all training data, including such spurious cases, with a notable drop for the P@1 metric (0.439 versus 0.447). Interestingly, excessive pruning also leads to a substantial performance loss, as informative instances are removed from the training data (P@1: 0.425 with 5 maximum positive evidence nodes versus 0.447 with our setting). Note that information redundancy is likely to appear in our setup with heterogeneous sources, which leads to more potential answering paths.

Choice of encoder LM. We also analyze the choice of the ELM used for encoding the graph nodes. Table 4.11 shows the result. Performance improves when using ROBERTA instead of DISTILROBERTA, but at the cost of run-time efficiency (1,125 ms versus 669 ms) due to the increased

Table 4.11: Comparison of EXPLAIGNN with different encoder LMs on the test set, all trained on graphs with 100 evidence nodes (inference on 500). Our proposed method uses DISTILROBERTA.

Method ↓	P@1	MRR	Hit@5	Run-time (HA)
EXPLAIGNN (DISTILROBERTA ²)	0.417	0.488	0.579	669 ms
EXPLAIGNN (ROBERTA ³)	0.427	0.496	0.580	1,125 ms
EXPLAIGNN (DISTILBERT ⁴)	0.410	0.476	0.560	827 ms
EXPLAIGNN (BERT ⁵)	0.408	0.476	0.559	1,404 ms

Table 4.12: 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	

number of layers (12 versus 6) and parameters (82M versus 125M). Further, DISTILROBERTA and ROBERTA generally perform slightly better than DISTILBERT and BERT.

4.5 User Study on Explainability

We evaluate the explainability of our pipeline via a dedicated user study on Amazon Mechanical Turk (AMT). The typical use case scrutinized is that *a user has an information need, obtains the answer predicted 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 easily, the explanations serve their purpose.

User study design. We propose the following 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 explanatory evidence. Examples of the input presented to the user are shown in Figure 4.13 and Figure 4.14 (we used five explanatory evidence snippets). We randomly sample 1,200 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.

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)certainly. 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.

4 Explainable Question Answering

Table 4.13: Representative examples from our user study, illustrating typical outputs of EXPLAINNN. In both cases, EXPLAINNN obtained the *correct* answer, and the users were *certain* about their decision.

<p>Question 1</p> <p>Who is the creator of Game of thrones?</p> <p>Answer 1</p> <p>D. B. Weiss</p> <p>Question 2</p> <p>Who is the actor behind Jon Snow?</p> <p>Answer 2</p> <p>Kit Harington</p>	<p>Question 1</p> <p>What country was the band Tears for Fears from?</p> <p>Answer 1</p> <p>England</p> <p>Question 2</p> <p>What year was the band formed?</p> <p>Answer 2</p> <p>1981</p>
<p>Current question</p> <p>Who is the actor behind Tormund Giantsbane?</p> <p>Provided answer</p> <p>Kristofer Hivju</p> <p>System interpretation</p> <p>Context entity: <i>Game of thrones</i> Current entity: <i>Tormund Giantsbane</i> Relation: <i>Who is the actor behind</i> Expected answer type: <i>human</i></p> <p>Supporting evidences</p> <ol style="list-style-type: none"> 1: Game of Thrones, cast member, Kristofer Hivju, character role, Tormund Giantsbane. 2: Tormund Giantsbane, present in work, Game of Thrones, performer, Kristofer Hivju. 3: Game of Thrones, award received, Primetime Emmy Award for Outstanding Supporting Actor in a Drama Series, point in time, 2011, winner, Peter Dinklage. 4: Satellite Award for Best Supporting Actor - Series, Miniseries or Television Film, winner, Peter Dinklage, point in time, 2011, for work, Game of Thrones. 5: Peter Dinklage, notable work, Game of Thrones. 	<p>Current question</p> <p>What's the name of their first album?</p> <p>Provided answer</p> <p>The Hurting</p> <p>System interpretation</p> <p>Current entity: <i>Tears for Fears</i> Relation: <i>Whats the name of their first album</i> Expected answer type: <i>album</i></p> <p>Supporting evidences</p> <ol style="list-style-type: none"> 1: Shout: The Very Best of Tears for Fears, The album contains the greatest hits of the band from their first album, The Hurting, to the much later Elemental. 2: Tears for Fears, To commemorate the 30th anniversary of the band's debut album The Hurting, Universal Music reissued it in October 2013 in two deluxe editions. 3: The Hurting, performer, Tears for Fears. 4: Pale Shelter, performer, Tears for Fears, has quality, re-recording, publication date, 1 March 1983, published in, The Hurting. 5: Tears for Fears, Towards the end of 1983, the band released a new, slightly more experimental single, 'The Way You Are', intended as a stopgap while they worked on their second album.
<p>Answer correctness (Correctness of the provided answer) ✓</p> <p>Predicted correctness (User assessment of the answer correctness) ✓</p> <p>User correctness (Correctness of the user assessment) ✓</p> <p>User certainty (User certainty about her assessment) ✓</p>	<p>Answer correctness (Correctness of the provided answer) ✓</p> <p>Predicted correctness (User assessment of the answer correctness) ✓</p> <p>User correctness (Correctness of the user assessment) ✓</p> <p>User certainty (User certainty about her assessment) ✓</p>

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

EXPLAINNN provides explainable answers. After pruning out the irrelevant assessments, in which users already knew the answer, or found the question (or conversation) confusing, 771 of the 1,200 observations remained. Findings are presented as a confusion matrix in Table 4.12 (values computed after removing confounding assessments). We found that the users can accurately decide the correctness of the system answer (76.1%), and are certain about their assessment most of the time (79.8%). This proves that explanations by EXPLAINNN 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: $P(\text{User correct} \mid \text{User certain}) = 0.792$.

Table 4.14: Representative examples from our user study, illustrating cases in which EXPLAIGNN obtained an *incorrect* answer. In the first case, the user could *certainly* tell that the answer is incorrect from the provided evidence. In the second case, the provided evidence is not sufficient for deciding the correctness of the answer (user was *uncertain*).

<p>Question 1</p> <p>Answer 1</p> <p>United States</p> <p>Question 2</p> <p>Answer 2</p> <p>8</p> <p>Question 3</p> <p>Answer 3</p> <p>22</p>	<p>Question 1</p> <p>Answer 1</p> <p>Rivaldo</p>		
<p>Current question</p> <p>Who played role of Christy Plunkett?</p> <p>Provided answer</p> <p>Allison Janney</p> <p>System interpretation</p> <p>Current entity: Mom Relation: Who played role of Christy Plunkett Expected answer type: human</p> <p>Supporting evidences</p> <ol style="list-style-type: none"> Mom (TV series), Allison Janney as Bonnie Plunkett: Christy's self-centered mother, a joyful if cynical recovering addict. Mom (TV series), Actor is Allison Janney, Character is Bonnie Plunkett, Seasons is 8. Mom (TV series), Mom has been met with widespread critical acclaim, with praise for its writing and performances, especially by Allison Janney and Anna Faris. Allison Janney, nominated for, Primetime Emmy Award for Outstanding Supporting Actress in a Comedy Series, for work, Mom, point in time, 2014. Allison Janney, award received, Primetime Emmy Award for Outstanding Supporting Actress in a Comedy Series, for work, Mom, statement is subject of, 67th Primetime Emmy Awards, point in time, 2015. 	<p>Current question</p> <p>Which football player was awarded FIFA world player of the year in 1999?</p> <p>Provided answer</p> <p>Santa Cruz Futebol Clube</p> <p>System interpretation</p> <p>Current entity: Rivaldo Relation: club he played for in the same year Expected answer type: association football club</p> <p>Supporting evidences</p> <ol style="list-style-type: none"> Rivaldo, Rivaldo started his career in 1991 with Brazilian club Santa Cruz, going on to have spells at Mogi Mirim, a loan spell at Corinthians, and Palmeiras. Rivaldo, Club is Santa Cruz, Season is 1991, Division is Serie B, Apps is 18, Goals is 8. Rivaldo, He went on to play for Santa Cruz in 1991. Rivaldo, Rivaldo signed a three-year deal with the Italian Serie A club Milan in 2002. Rivaldo, member of sports team, Brazil national under-20 football team, number of matches played/races/starts, +13, number of points/goals/set scored, +2, end time, 1993, start time, 1991. 		
<p>Answer correctness (Correctness of the provided answer)</p> <p>Predicted correctness (User assessment of the answer correctness)</p> <p>User correctness (Correctness of the user assessment)</p> <p>User certainty (User certainty about her assessment)</p>	<p>X</p> <p>X</p> <p>✓</p> <p>✓</p>	<p>Answer correctness (Correctness of the provided answer)</p> <p>Predicted correctness (User assessment of the answer correctness)</p> <p>User correctness (Correctness of the user assessment)</p> <p>User certainty (User certainty about her assessment)</p>	<p>X</p> <p>X</p> <p>✓</p> <p>X</p>

Anecdotal examples. Figure 4.13 and Figure 4.14 show example outputs of EXPLAIGNN, and the corresponding results from the user study. For the examples in Figure 4.13 the EXPLAIGNN answers were *correct*. Based on the explanation (system interpretation and supporting evidence pieces) the users could *certainly* tell the correctness of the answer. For the examples in Figure 4.14, the answers provided by EXPLAIGNN were *incorrect*. In the first case, the user was able to identify this incorrectness *with certainty* from the explanation (the first and second supporting evidence pieces mention that Allison Janney played Bonnie Plunkett, and not Christy Plunkett). The user could then try to reformulate [100] the question to obtain the correct answer. In the second case, the provided information was not sufficient, since there is no information on the club Rivaldo played for in 1999. So the user was *uncertain* about the answer correctness. Note that even in this scenario the end user would understand that the provided answer needs to be

verified. This is in contrast to incorrect answers generated by GLMs, which often look perfect on surface, giving the user a false sense of correctness.

The guidelines, code, and the results for the user study are publicly available⁶.

4.6 Demonstration

We provide a demo for EXPLAIGNN at <https://explaignn.mpi-inf.mpg.de>. Our demo provides two different configurations: (i) “*more efficient*” which constructs graphs from 100 evidence snippets, prunes to 5 evidence nodes and then predicts the answer, and (ii) “*better performance*” which starts with 500 evidence nodes, prunes to 100, then 5 and then predicts the answer. Note that the demo is run on a CPU only, without access to any GPU, leading to increased run-times. Evidence is retrieved on-the-fly from Wikidata via the CLOCQ-API⁷, or from Wikipedia. We retain an entity-level evidence cache for efficiency. Figure 4.4 shows a screenshot of the demo interface, and the explanations provided by EXPLAIGNN for its answers. Our demo shows the system interpretation (derived from the SR), and supporting evidence, consisting of a ranked list of 5 evidence snippets. Within the evidence, entities are linked to Wikidata, and each evidence snippet links to its original source.

4.7 Related Work

Conversational question answering. There has been extensive research on ConvQA [169, 175] in recent years, which can largely be divided into methods using a KB [107, 112, 137], methods using a text corpus [161, 162, 163], and methods integrating tables [83, 144]. Refer to Section 3.7 for more details on these source-specific approaches.

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 [33, 41]. The method proposed in [41] 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 (Chapter 3), evidence obtained from heterogeneous sources is concatenated and fed into an FID model to generate the answer. Both methods heavily rely on Seq2seq LMs that require the input to be plain token sequences. Further, the generated outputs are not explainable, and may even be hallucinated.

Explainable QA. Existing work is mostly on single-turn methods operating over a single source, with template [1, 188] and graph-based derivation sequences [93, 132] as mechanisms for ensuring explainability. Works on Text-QA provide end users with actual passages and reasoning paths used for answering [151, 240]. Post-hoc explainability for QA over KBs and text is investigated in [201]. 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 [249], 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.

⁶https://qa.mpi-inf.mpg.de/explaignn/explaignn_user_study.zip

⁷<https://clocq.mpi-inf.mpg.de>

Demo

More efficient Better performance

Question

Who wrote the book *Angels and Demons*?

Provided answer

Dan Brown

System interpretation

Current entity: *Angels and Demons*
Relation: *Who wrote the book*
Expected answer type: *human*

Supporting evidences

- #1: **Angels & Demons**, Author, **Dan Brown**. [Infobox]
- #2: **Angels & Demons**, author, **Dan Brown**. [KB]
- #3: **Angels & Demons** (film), *Angels Demons*, Based on, *Angels Demons* by **Dan Brown**. [Infobox]
- #4: **Angels & Demons**, after a work by, **Dan Brown**. [KB]
- #5: **Dan Brown**, notable work, **Angels & Demons**. [KB]

Question

the main character in his books?

Provided answer

Robert Langdon

System interpretation

Context entity: *Angels and Demons*
Current entity: *Dan Brown*
Relation: *the main character in his books*
Expected answer type: *fictional human*

Supporting evidences

- #1: **Dan Brown**, In 2021, Dan Brown's *The Lost Symbol* was adapted into a television series repositioned as an origin story for Brown's **Robert Langdon** character with Ashley Zukerman playing Langdon. [Text]
- #2: **Dan Brown**, Brown also claims to have based the character of **Robert Langdon** on Campbell. [Text]
- #3: **Dan Brown**, In a 2006 interview, Brown stated that he had ideas for about 12 future books featuring **Robert Langdon**. [Text]
- #4: **Dan Brown**, **Robert Langdon** is named after John Langdon, the artist who created the ambigrams used for the *Angels & Demons* CD and novel. [Text]
- #5: **Robert Langdon**, creator, **Dan Brown**. [KB]

Please enter your question here

Answer

Clear Last

Clear All

Figure 4.4: Screenshot of the EXPLAINNN demo.

4.8 Conclusion

There are three main takeaways for a broader audience from this work. First, at a time when generative language models like ChatGPT 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, effectiveness, 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.

5

Temporal Question Answering

Another challenge for QA, apart from incomplete conversational questions, is posed by questions with explicit or implicit *temporal constraints*, such as “...in 2019” or “...before COVID”. Similarly to the case in ConvQA, these questions require in-depth understanding of the user’s information need, for modeling the time condition and providing answers that satisfy this user constraint *faithfully*. We propose a new methodology to this end, named FAITH, which includes a dedicated temporal question understanding stage to represent such time conditions. These conditions are applied in a rule-based manner, on evidence that is retrieved from heterogeneous sources. The answering stage of EXPLAIGNN is then used for deriving the answer. For experiments, we create the TIQ benchmark with 10,000 questions that involve implicit time constraints.

All code and data are available at <https://faith.mpi-inf.mpg.de>.

Contents

5.1	Introduction	62
5.2	Concepts and Notation	64
5.3	The FAITH Methodology	64
5.3.1	Question Understanding (QU)	64
5.3.2	Evidence Retrieval and Scoring (ERS)	68
5.3.3	Heterogeneous Answering (HA)	69
5.4	The TIQ Benchmark	69
5.4.1	Existing Benchmarks and their Limitations	69
5.4.2	Construction of the TIQ Benchmark	70
5.4.3	Characteristics of the TIQ Benchmark	72
5.5	Experimental Setup	73
5.5.1	Benchmarks	73
5.5.2	Metrics	73
5.5.3	Baselines	73
5.5.4	Configurations	74
5.6	Experimental Results and Insights	75
5.6.1	Key Findings	75
5.6.2	Faithfulness Evaluation	76
5.6.3	Analysis	77
5.7	Related Work	81
5.8	Conclusion	82

5.1 Introduction

Motivation. Question answering (QA) comprises a spectrum of settings for satisfying users’ information needs, ideally giving crisp, entity-level answers to natural-language utterances [175]. Temporal QA specifically focuses on questions with temporal conditions (e.g., [93, 138, 182]), making up a substantial portion of user needs [254], but poses challenges that are not properly met by universal QA systems. Consider the following example:

q^1 : *Record company of Queen in 1975?*

The band **Queen** had different record companies over the years, so it is decisive for the QA system to consider the *explicit temporal constraint* (“in 1975”) when answering the question. Other questions with explicit time are look-ups of dates, such as:

q^2 : *When was Bohemian Rhapsody recorded?*

Another – underexplored and most challenging – situation is when questions express *implicit temporal constraints*. These can involve the need to compare different time points or intervals, even when the user input does not explicitly state it. Examples are:

q^3 : *Queen’s record company when recording Bohemian Rhapsody?*
 q^4 : *Queen’s lead singer after Freddie Mercury?*

For such questions, users expect the QA system to actually understand and apply the temporal constraint for identifying the correct answer. For q^4 , the system has to find out when Mercury died or left the band, in order to compute the correct answer that Brian May (the band’s guitarist) took over as lead singer.

Limitations of the State of the Art. The research literature on temporal QA is substantial, including [25, 29, 45, 90, 93, 94, 138, 182, 229]. Most methods address all kinds of temporal questions, but are typically less geared for implicit questions. Some methods operate over curated KBs (e.g., [45, 90, 93]), while others are designed for processing text corpora such as news collections or Wikipedia full-text (e.g., [25, 150]).

We observe three major issues in prior works:

- (i) Many methods use “soft-matching” techniques, based on latent embeddings or language models. This may lead to invalid answers, where the non-temporal part of a question is matched, but the temporal constraint is violated. For example, a question about “*Queen’s record company in 1990?*” may erroneously return **EMI** instead of the correct value **Parlophone**, because **EMI** is more prominent and was Queen’s company on most albums. Even when the output is correct, this could be by the prominence of the answer alone. For example, “*Who was Queen’s lead singer in 1975?*” could return the most popular entity **Freddie Mercury** without checking the time. When we vary the question into “... *in 2000?*”, many systems would still yield **Freddie Mercury**, although he had passed away by then. This indicates that the system has incomplete inference. We call this phenomenon *unfaithful QA*. Since the answer derivation of neural methods is often not explainable, end users have no chance to understand *how* the answer was actually obtained, and whether their temporal constraints were met.
- (ii) A weak spot of temporal QA systems is the handling of *implicit intents*. These are infrequent in established benchmarks. Some methods [45, 90, 146] aim to transform the implicit conditions into explicit temporal constraints, based on classifying phrases such as “*during*” or “*before*”. However, they heavily rely on handcrafted rules which are rather limited in scope and cannot robustly handle unforeseen utterances.

-
- (iii) Prior methods run on a *single information source*: either a KB or a text corpus. As discussed before, this limits QA coverage: KBs are incomplete and lack refined detail about events, whereas text collections are harder to extract answers from and often fail on complex questions. Existing methods for QA over heterogeneous sources [34, 154], including also web tables, do not support temporal conditions.

Approach. To overcome these limitations, we propose FAITH (Faithful Temporal Question Answering over Heterogeneous Sources), a temporal QA system that operates over *heterogeneous* sources, seamlessly combining a KB, a text corpus and web tables. FAITH follows the architecture of CONVINSE (Chapter 3) and EXPLAIGNN (Chapter 4), and consists of three main stages:

- (i) *question understanding* (QU) for representing the question intent into a structured representation (SR), with specific consideration of the temporal aspects;
- (ii) *evidence retrieval and scoring* (ERS) for identifying relevant pieces of evidence from KB, text and tables, with time-aware filtering to match the temporal conditions;
- (iii) *heterogeneous answering* (HA) to compute entity-level answers and supporting evidence for explanation from these information snippets.

Figure 5.1 provides an overview of the FAITH pipeline. A key novelty in the question understanding is that implicit constraints are resolved into explicit temporal values by generating intermediate questions and recursively calling FAITH itself. For example, the implicit condition “*when recording Bohemian Rhapsody*” in q^3 is transformed into “*when Queen recorded Bohemian Rhapsody?*”, and the recursive invocation of FAITH returns the explicit condition **August 1975 - September 1975**. This derived explicit condition is then used in a similar vein as the explicit condition **1975** in q^1 , making it easier to answer the information need. Note that this is not just question rewriting, but is driven by the full-fledged QA system itself over the full suite of heterogeneous sources.

A second key novelty is that, in contrast to most prior works including generative LMs, FAITH provides *traceable provenance* for the answer derivation. Providing intermediate results, such as the explicit temporal conditions and the SR identified in the question understanding stage, and the explanatory evidence for answers, make FAITH a truly faithful temporal QA system.

Existing benchmarks for temporal QA focus on a single information source at hand (either a KB or a text corpus), and include only few questions with implicit constraints (so the weak performance on these hardly affects the overall benchmark results). Further, none of the existing benchmarks target heterogeneous sources for answering, but are closely tied to the information source used for creating the dataset. Therefore, we devise a new method for automatically creating temporal questions involving *implicit constraints*. Our method allows for systematic control of different aspects, including the relative importance of different source types (text, infoboxes, KB), coverage of topical domains (sports, politics, etc.), fractions of prominent versus long-tail entities, question complexity, and more. This way, we construct a new dataset named TIQ with 10,000 questions and ground-truth answers accompanied by supporting evidence.

Contributions. The salient contributions of this work are:

- Proposing the first temporal QA system that taps into heterogeneous information sources, and provides faithful answers with explanatory evidence.
- Devising a mechanism that transforms implicit temporal constraints into explicit conditions, by recursively calling the QA system itself.
- Developing a principled method for automatic construction of diverse and difficult temporal questions with implicit constraints from heterogeneous sources, and releasing the TIQ benchmark with 10K such implicit questions.

5.2 Concepts and Notation

This section introduces relevant concepts and notation that are specific to the FAITH project.

Temporal value. A *temporal value* indicates a point in time or time interval. It can be a specific *date* (e.g., 24 November 1991), a *year* (e.g., 1975), or a *time period* (e.g., August 1975 - September 1975).

Temporal constraint. A *temporal constraint* specifies a condition about a time point or interval that has to be satisfied by the answer and its evidence. Temporal constraints consist of a temporal value, and a *temporal signal* (like *before*, *after*, *overlap*). An example of a (verbalized) temporal constraint is “*in 1970*”.

Explicit question. An *explicit question* mentions a specific temporal constraint explicitly, as in “*Record company of Queen in 1975?*”.

Implicit question. An *implicit question* also specifies a temporal constraint, but keeps this constraint *implicit* without mentioning the actual temporal value: “*Queen’s record company when recording Bohemian Rhapsody?*”.

Faithfulness. A system answer is *faithful* if its evidence (that is grounded to existing information sources), provided with the answer, matches the semantics of the question and contains:

- (i) the answer,
- (ii) all entities that appear in the question (with any surface name),
- (iii) all predicates that appear in the question (at least in paraphrased or implicit form),
- (iv) a temporal expression that satisfies the temporal constraint of the question.

5.3 The FAITH Methodology

Figure 5.1 provides an overview of the system architecture, illustrated with the processing of the running example q^3 . The following subsections present the three main components (understanding, retrieval, and answering).

5.3.1 Question Understanding (QU)

The goal of this first stage is to capture the temporal information need in a frame-like structure. Notably, this stage identifies and categorizes temporal constraints in the user input, which is later used for pruning temporally-inconsistent answer candidates.

Structured representation. We propose a new variant of our structured representation, geared to capture temporal questions. This variant includes both general-QA-relevant slots (similar to CONVINSE and EXPLAIGNN):

- *question entity*,
- *question predicate*,
- *expected answer type*,

and temporal-QA-relevant slots:

- *temporal signal*, indicating the kind of temporal relation),
- *temporal category*, indicating the type of temporal constraint,
- *temporal value*, the time point or interval of interest (if present).

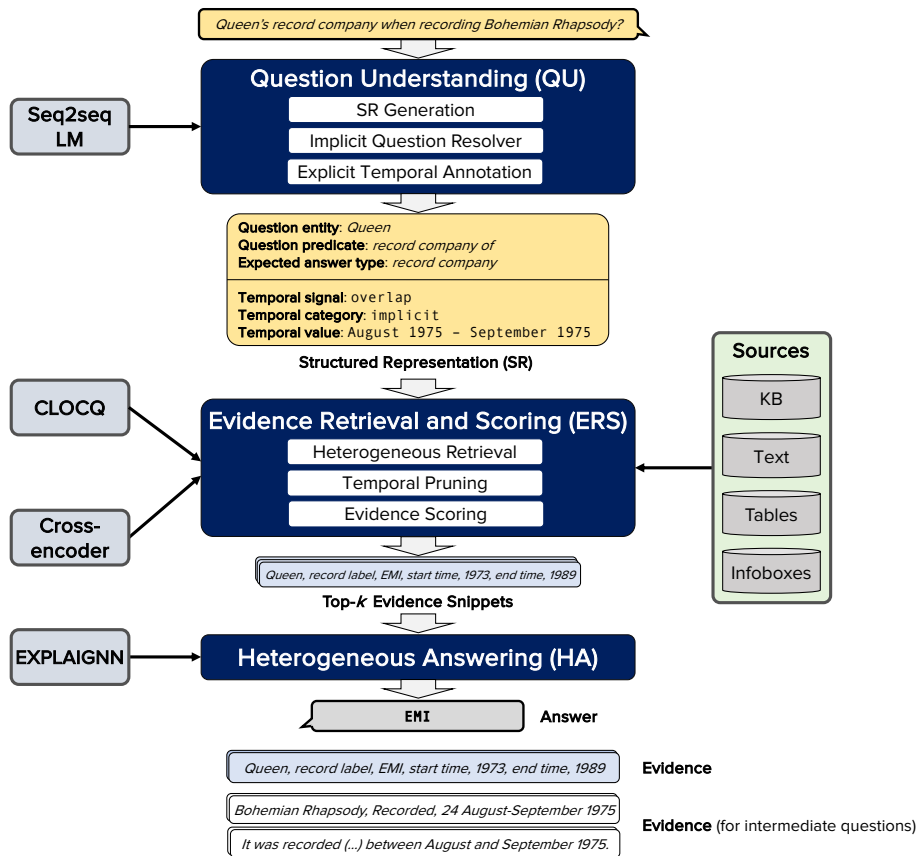


Figure 5.1: An overview of the FAITH methodology.

The *question entity* and *predicate* are taken from the surface form of the question (i.e., *not* linked to the KB) to allow for uniform treatment of heterogeneous sources. The *expected answer type* is learned from the training data, in which the KB-type of the gold answer is used. The *temporal signal* can be *overlap* (e.g., from cues like “in”, “during”), *before* (e.g., from cues like “before”, “prior to”), or *after* (e.g., from cues like “after”, “follows”). We categorize the constraint into *implicit* (e.g., q^3 and q^4) and *non-implicit* (e.g., q^1 and q^2). The *temporal value* can be a *year*, *month*, *date* or *time period*. Both the temporal signal and value are derived by identifying and normalizing key phrases in the input question.

For example, the SR for q^1 (“Record company of Queen in 1975?”) is:

Question entity: Queen Question predicate: record company of Expected answer type: record company
Temporal signal: overlap Temporal category: non-implicit Temporal value: 1975

Note that in case the question does not specify temporal constraints (e.g., q^2), the respective fields are simply kept empty (indicated by an underscore “_” as before).

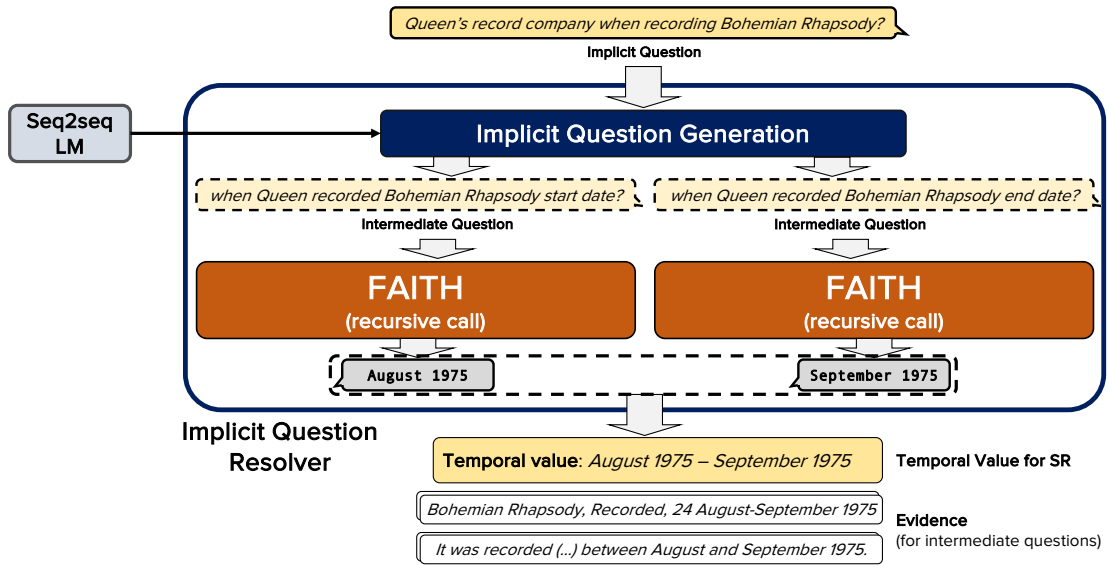


Figure 5.2: The figure illustrates the process for resolving the implicit constraint in q^3 (“Queen’s record company when recording Bohemian Rhapsody?”) within the QU stage, by generation of intermediate questions and recursive calls to the entire FAITH pipeline.

Resolving implicit questions. For the challenging case of implicit questions, such as q^3 or q^4 , the temporal value cannot be extracted from the question directly. To resolve this problem, we devise a novel mechanism, the *implicit question resolver*, based on recursively invoking the temporal QA system itself. To this end, the implicit temporal constraint in the question is identified and transformed into an *intermediate question*. For instance, the intermediate question for q^4 (“Queen’s lead singer after Freddie Mercury?”) would be “when Freddie Mercury lead singer of Queen?”. For q^3 , the temporal value should be a time interval (August 1975 – September 1975). Thus, two intermediate questions are required:

- (i) q^{31} : “When Queen recorded Bohemian Rhapsody start date?”
- (ii) q^{32} : “When Queen recorded Bohemian Rhapsody end date?”

The intermediate questions are fed into FAITH as a recursive call, to obtain the explicit temporal value for filling the SR of the original question. Although these formulations are ungrammatical, the QA system can process them properly, being robust to such inputs.

The SR for q^3 thus becomes:

Question entity: *Queen*
Question predicate: *recorded company*
Expected answer type: *record company*

Temporal signal: *overlap*
Temporal category: *implicit*
Temporal value: *August 1975 – September 1975*

Note the similarity to the SR of the explicit temporal question q^1 . Figure 5.2 illustrates this process of resolving implicit constraints for q^3 .

5.3.1.1 Generating Intermediate Questions

The intermediate questions are generated by a fine-tuned Seq2seq LM, specifically BART [115]. A major obstacle, though, is that no prior dataset has suitable annotations, and collecting such data at scale is prohibitive. Therefore, we generated training data using INSTRUCTGPT [155], leveraging its *in-context learning* (ICL) [14] capabilities. We randomly select 8 implicit questions from our train set and label them manually. For each question, we give the intermediate question and the expected answer type as output.

The expected answer type of an intermediate question can be `date` or `time interval`. When the expected answer type is a time interval (e.g., for q^3), two intermediate questions are created, appending “*start date*” and “*end date*” to the generated intermediate question, respectively (see q^{31} and q^{32} as example).

We use this technique to annotate all implicit questions in the train and dev sets, obtaining training data for fine-tuning the BART model. Note that INSTRUCTGPT is used only for the generation of training data. It is not used by FAITH at run-time to avoid its (computational, monetary, and environmental) cost and dependency on black-box models.

Training. To annotate intermediate questions (and its expected answer type) for implicit questions, we leverage ICL: we select and label 8 questions from the train set, and give these pairs as context to INSTRUCTGPT. This way we annotate the remaining questions in the train and dev sets of TIQ/TIMEQUESTIONS resulting in 5,875/847 instances in the train set and 1,949/287 instances in the dev set. On this data we fine-tune the BART model to be independent of INSTRUCTGPT at inference time. The exact prompts and ICL examples used are shown in Table 5.12.

Inference. For generating intermediate questions at run-time, we provide the implicit question as input to the trained BART model. The output is the intermediate question that describes the implicit constraint, and the expected answer type for this question, separated by two pipes.

Example output for “*Queen’s record company when recording Bohemian Rhapsody?*”:

```
“when Queen recording Bohemian Rhapsody || time interval”
```

5.3.1.2 Construction of the SR

We also use a fine-tuned Seq2seq LM, again BART, for generating the values for the question entity, question predicate, expected answer type, temporal signal, and temporal category slots of the SR representation. The training data for this SR construction model is obtained via:

- (i) distant supervision for the question entity and question predicate (cf. Section 3.2.5),
- (ii) KB-type look-ups for the expected answer type, and
- (iii) annotations in the benchmark for the temporal signal and temporal category.

The temporal values are obtained via the recursive mechanism discussed above for implicit questions, and via SUTIME [21] and regular expression matching for explicit questions. Phrases like “*today*” or “*current*” are considered as well and properly normalized. We use the creation time of the question [19], as provided in the benchmarks, as reference time.

The SR generated in the QU stage is used for representing the entire temporal information need in the subsequent retrieval and answering stages, including its key temporal characteristics.

Training. To obtain training data for the SRs, we follow a similar distant supervision approach as in Section 3.2.5 for obtaining the target *question entity* and *question predicate*. We run the heterogeneous retriever (see Section 5.3.2) on the full input question, which identifies entity men-

tions in the input, disambiguates these to KB-entities, and then retrieves information snippets for the KB-entities from heterogeneous sources. If the retrieved information snippets for a KB-entity contain the annotated gold answer, we annotate the corresponding entity mention as a relevant question entity. The remaining parts of the question are annotated as the question predicate mention(s). The *expected answer type* is the most frequent (proxy for most prominent) KB-type of the gold answer. The *temporal signal* and the *temporal category* are looked up from the annotations in the benchmarks. These individual parts of the SR are then combined and separated by pipes (“|”), to obtain the target generations that are used for training the SR construction model. Details on the training configuration are provided in Section 5.5.

Inference. The input to the BART model for SR construction is the question, as given by the user. The output is the concatenation of the individual slots, separated by two pipes (“||”): “{entities}||{predicate}||{expected answer type}||{temporal signal}||{temporal categorization}”. This output is then parsed to identify the different slots.

Example output for q^1 (“Record company of Queen in 1975?”):

```
“Queen || Record company of in 1975 || record company || overlap || non-implicit”
```

5.3.2 Evidence Retrieval and Scoring (ERS)

In this stage, we first retrieve evidence from heterogeneous sources, and then prune out information inconsistent with the temporal constraint, as expressed by the SR. We perform three steps: (i) heterogeneous retrieval, (ii) temporal pruning, and (iii) evidence scoring.

Heterogeneous retrieval. This step largely follows our retrieval in CONVINSE and EXPLAINNN (in Section 3.2.2), and makes use of entity linking. Entity mentions in the input are identified and linked via CLOCQ [31]. The input here is the concatenation of the *question entity*, the *question predicate*, and the *expected answer type* of the SR. For the resulting linked entities, we retrieve the Wikipedia pages for extracting text, tables, and infoboxes. Further, KB-facts with the linked entities are obtained from Wikidata.

All retrieved pieces of evidence are *verbalized* into textual sentences, for uniform treatment. The KB-facts are verbalized by concatenating their individual parts; the text evidence is split into sentences; table rows are transformed by concatenating the individual ⟨column headers, cell value⟩ pairs; infoboxes are handled by linearizing all attribute-value pairs.

Temporal pruning. Explicit temporal expressions in the retrieved pieces of evidence are identified and normalized similarly as in the understanding stage. Evidence that does not match the temporal criteria is pruned out. We address two kinds of *temporal pruning* situations:

- (i) the question aims for a temporal value as answer and does not have any temporal constraints, neither implicit nor explicit (e.g., “When ...?”);
- (ii) the question has a temporal constraint which needs to be matched by the evidence.

In the first case, all pieces of evidence that do not contain any temporal values, and are thus unable to provide the answer, are dropped. In the second case, we remove evidence pieces that do not match the temporal constraint, to ensure that answers are faithful to the temporal intent of the question. This is performed by applying deterministic rules.

Evidence scoring. As the set of candidate information snippets after temporal pruning can still be large, we use a re-ranker [152] to identify the top- k most relevant evidence pieces, based on cross-encodings obtained via DISTILROBERTA¹.

¹<https://huggingface.co/distilbert/distilroberta-base>

Table 5.1: Comparison of benchmarks for temporal QA.

Benchmark	No. of implicit questions	Knowledge source		
		KB	Text	Infobox
TIME-SENSITIVE QA [25]	–	✓	✓	✗
STREAMINGQA [123]	–	✗	✓	✗
TEMPQUESTIONS [89]	209	✓	✗	✗
TIMEQUESTIONS [93]	1,476	✓	✗	✗
TEMPQA-WD [146]	154	✓	✗	✗
TEMPTABQA [70]	7,242	✗	✗	✓
CRONQUESTIONS [182]	91,165 (5 KB-relations)	✓	✗	✗
TEMPREASON [205]	21,877 (10 KB-relations)	✓	✗	✗
Tiq (ours)	10,000	✓	✓	✓

The training data are the (question, information snippet) pairs, annotated with either a positive label (in case the snippet contains a gold answer) or a negative label (otherwise). We randomly sample (at most) one positive (question, information snippet) pair from each knowledge source and 15 negative pairs, for each question. These negative pairs are drawn from the result of the heterogeneous retrieval and temporal pruning, and thus constitute challenging negatives for boosting the ranking capabilities of the model. We use the concatenation of the question entity, the question predicate, and the expected answer type, as present in the SR, for representing the question in this step.

The output of this retrieval stage is a set of evidence pieces that faithfully reflects the time constraints of the temporal information need. The final answer and its explanation are computed from this pool of evidence.

5.3.3 Heterogeneous Answering (HA)

Since this part is not the main focus of this work, we make direct use of the state-of-the-art answering stage of EXPLAIGNN (Section 4.2.3), that is based on iterative GNNs, and computes a subset of supporting evidence for the predicted answer. Thus, we ensure that the answer can be traced back through the entire system including the answering stage, for end user explainability. The input query to the GNNs is the concatenation of the question entity, question predicate, and expected answer type.

5.4 The TIQ Benchmark

5.4.1 Existing Benchmarks and their Limitations

Most existing benchmarks for temporal QA, like TEMPQUESTIONS [89], TIMEQUESTIONS [93] or TEMPQA-WD [147], have only few implicit questions (209, 1,476, and 154, respectively), falling short of evaluating one of the key challenges in temporal QA. CRONQUESTIONS [182] and TEMPREASON [205] have a larger portion of implicit questions, but all questions are derived from a very small set of KB-relations (5 and 10, respectively) and handcrafted rules, thus lacking diversity of formulations and intents. *Syntactic diversity* poses a major challenge for all QA tasks [175]. Further, questions in these benchmarks are always answerable using a single information source (either KB or text corpus), which is not a realistic setting.

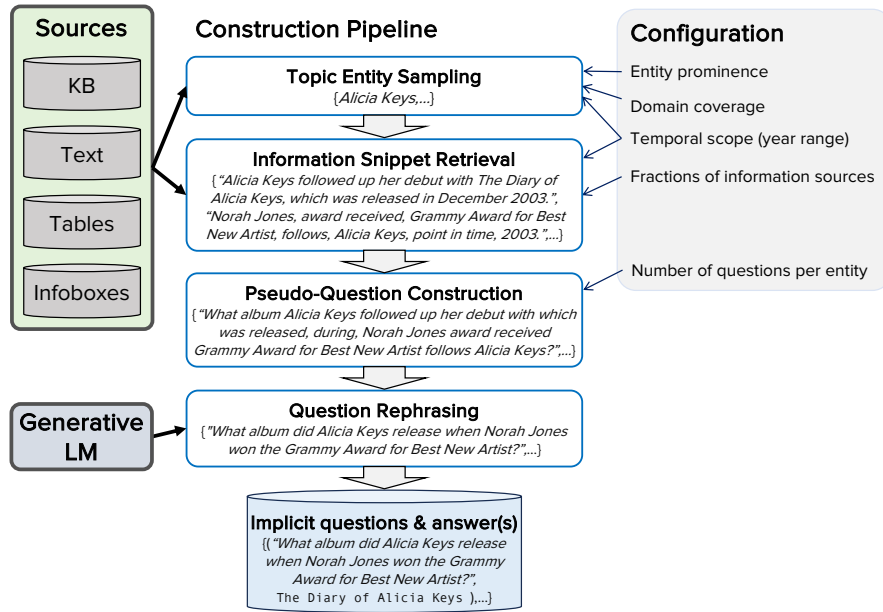


Figure 5.3: The steps to create implicit questions with our proposed methodology, highlighting the key configurable parts.

Consequently, we construct a new benchmark, named TIQ (Temporal Implicit Question), with a primary focus on challenging and diverse implicit questions. The obvious idea of using crowdsourcing is expensive and error-prone. Also, crowdworkers increasingly use GLMs as a shortcut [218]. Thus, we pursue an automated process instead. To ensure that questions are not specific to a single input source, our process considers multiple sources: Wikipedia text and infoboxes, and the Wikidata KB. Table 5.1 compares characteristics of benchmarks, vis-a-vis our new TIQ dataset.

5.4.2 Construction of the TIQ Benchmark

The following outlines our automatic construction pipeline for deriving challenging implicit questions, based on evidence from heterogeneous sources, at scale.

Overview. An implicit question has two parts: the *main question* that specifies the information need disregarding time (e.g., “Queen’s lead singer” for q^4), and the *implicit part* that provides the temporal constraint (e.g., “after Freddie Mercury” for q^4). The key idea is to build each of the two parts from independent pieces of evidence, denoted as *information snippets*. The two snippets can come from different sources, but need to be thematically related.

The construction process has the following steps:

- (i) sample a set of topic entities to start with;
- (ii) retrieve temporal information snippets for each such topic entity from Wikipedia text, Wikipedia infoboxes, and Wikidata;
- (iii) concatenate information snippets using a suitable temporal signal and construct an interrogative sentence, a *pseudo-question*;
- (iv) rephrase the pseudo-question into a natural question using a generative model.

An overview of this process is provided in Figure 5.3, including an example case of constructing an implicit question. Naturally, implicit constraints are based on global events (e.g., the COVID pandemic), or major events for a specific entity (e.g., receiving a prestigious award).

Sampling topic entities. To obtain such significant events, we start with Gregorian calendar year pages in Wikipedia (e.g., <https://en.wikipedia.org/wiki/2023>) that list notable events. From the pages for the years 1801-2025, we collect information snippets about such significant events. The entities in these snippets constitute the set of topic entities (“href” anchors are used for entity linking [55]). In our example in Figure 5.3 this set includes `Alicia Keys`.

Retrieving the grounding information snippets. We collect snippets about notable events in these year pages, and augment them with salient information about the topic entity from:

- (i) the first five sentences (\simeq first passage) of the entity’s Wikipedia page,
- (ii) the respective Wikipedia infobox, and
- (iii) the entity’s Wikidata facts.

As candidates for the main question part, we consider all information snippets that are retrieved for a topic entity from Wikipedia text, infoboxes and Wikidata, irrespective of their salience. To avoid questions that are trivially answerable without considering the time condition, multiple candidate snippets are retrieved for the main question, with different temporal scopes (e.g., a band’s singers from different epochs). This is implemented by measuring semantic similarity among candidates using a `SENTENCETRANSFORMER`² [170].

Creating a pseudo-question. Among the retrieved snippets for an entity, we identify pairs of candidate snippets that can be connected by a temporal conjunction/preposition (“*during*”, “*after*” and “*before*”). The temporal scopes have to be consistent with the temporal conjunction.

A valid pair for the conjunction “*during*” would be:

Main question part: “*Alicia Keys followed up her debut with `The Diary of Alicia Keys`, which was released in December 2003.*” [TEXT]
Implicit part: “*Norah Jones, award received, Grammy Award for Best New Artist, follows, Alicia Keys, point in time, 2003.*” [KB]

A *pseudo-question* is created by concatenating the main part with the conjunction and the implicit part. The answer is an entity (not the topic entity) from the main part, `The Diary of Alicia Keys` in this case. The answer is substituted by the prefix “*what*” followed by the most frequent KB-type of the answer, which would be `album` for this example.

The pseudo-question for the example is:

“*What album Alicia Keys followed up her debut with which was released, during, Norah Jones award received Grammy Award for Best New Artist follows Alicia Keys?*”

which is an ungrammatical and unnatural formulation.

Rephrasing to a natural question. Therefore, in the last step, we rephrase the pseudo-question to a natural formulation. We use `INSTRUCTGPT` with 8 ICL examples (pseudo-questions and their natural rephrasings), to generate the final question. The prompt is provided in Table 5.11. The demonstrations are selected from the set of pseudo-questions obtained from the previous step and annotated manually, involving rewording and restructuring the input.

²<https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2>

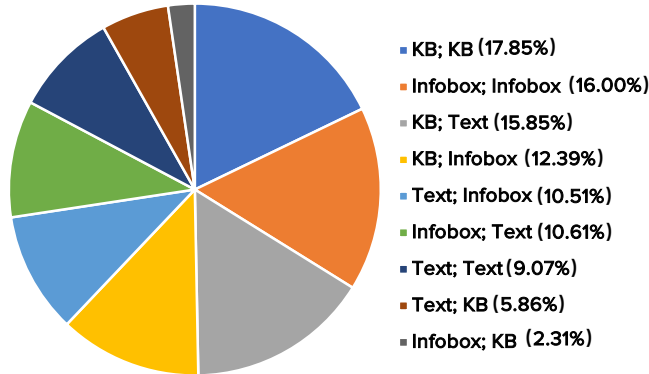


Figure 5.4: Distribution of questions over input source combinations (source for main question part; source for implicit part) in our TIQ dataset.

Table 5.2: Basic statistics for our TIQ benchmark.

Sources	Wikipedia text, infoboxes, and Wikidata
Questions	10,000 (train: 6,000, dev: 2,000, test: 2,000)
Avg. question length	17.96 words
Avg. no. of question entities	2.45
Unique topic entities covered	10,000
Long-tail topic entities covered	2,542 (with < 20 KB-facts)
Prominent topic entities covered	2,613 (with > 500 KB-facts)

The pseudo-question of the example is rephrased into the following implicit question:

“What album did Alicia Keys release when Norah Jones won the Grammy Award for Best New Artist?”

5.4.3 Characteristics of the TIQ Benchmark

Initialization. For creating the TIQ benchmark we started with the years 1801-2025 and obtained an initial set of 229,318 entities. From this set, we uniformly sampled 10,000 topic entities based on their frequency, to capture a similar amount of long-tail and more prominent entities (see Table 5.2 for details). These fractions can be configured as required. Since some entity types were over-represented in the calendar year pages (e.g., politicians or countries), we also ensured that individual entity types are not taking up more than 10% of the topic entities. In general, the topic entity set allows to control the domain coverage within the generated implicit questions, by choosing entities of the desired types. Note that for some entities it is not possible to obtain valid pairs of information snippets for grounding the question parts. In this case, we simply re-sample another topic entity.

We did not specifically configure the proportions to which the individual information sources are used within the questions, since we observed a naturally diverse distribution. Figure 5.4 shows the distribution among source combinations for initiating the main and implicit parts. The questions are finally split into train (6,000), dev (2,000), and test sets (2,000). Table 5.2 shows the basic statistics, and Table 5.3 shows representative questions of the TIQ benchmark.

Distribution of topic entities. In TIQ, each question is derived from a unique topic entity. There are 2,542 long-tail entities and 2,613 prominent entities in the benchmark (Table 5.2).

The topic entities are quite diverse, with a total of 1,100 different KB-types, covering top-level domains like `human`, `city`, `sports`, `country`, `business`, `university`, `politics`, `film/TV series`, `music`, `organization/company`, or `museum`.

Granularity of temporal values. We analyzed the granularity of temporal expressions in the information snippets. The number of years is 12,094, there are 538 months, and 7,994 dates. Months are naturally a bit rarer than years and dates. For the main question part, the numbers of years, months and dates are 6,670, 204 and 3,798, respectively. For the implicit part, the numbers of years, months and dates are 5,424, 334 and 4,196, respectively.

Correctness. We randomly sampled 100 questions from the benchmark and manually inspected to check if they preserve the intent of the original pseudo-questions. The portion of the questions preserving the original meaning is 82%. The reasons for failures include four aspects:

- (i) wrong predicate in main question part (8%),
- (ii) error with constraint (9%),
- (iii) incorrect answer type (6%), and
- (iv) wrong temporal conjunction (2%).

The number of error cases adds up to 25% (>18%) because few questions fail in multiple aspects. For example, for two questions, INSTRUCTGPT switched their main question and implicit parts resulting in errors in both parts of the rephrased questions.

Meta-data. TIQ provides implicit questions and gold answers, as strings as well as canonicalized to Wikipedia and Wikidata. Through our automated question creation process we also obtain a range of useful meta-data for the questions: TIQ includes the information snippets grounding the question, the sources these were obtained from, the explicit temporal value expressed by the implicit constraint, the topic entity, the question entities detected in the snippets, and the temporal signal.

The TIQ dataset is available at <https://faith.mpi-inf.mpg.de>.

5.5 Experimental Setup

5.5.1 Benchmarks

We conduct experiments on our new TIQ benchmark and on TIMEQUESTIONS [93], which has been actively used in recent work on temporal QA, and contains 16,181 temporal questions collected from existing benchmarks. Questions can have implicit or explicit temporal constraints, ask for temporal values, or specify ordinals (e.g., “*what was the first album by Queen?*”). For ordinal questions in TIMEQUESTIONS, we apply the same method as outlined in Section 5.3, but without the temporal pruning.

5.5.2 Metrics

We use the standard QA metrics *precision at 1* (**P@1**), *mean reciprocal rank* (**MRR**), and *hit at 5* (**Hit@5**) [175]. All metrics are averaged over all questions.

5.5.3 Baselines

We compare FAITH against a suite of baselines, covering a diverse range of competitors:

5 Temporal Question Answering

Table 5.3: Representative questions from the TIQ benchmark, including the information snippets used for initiating the main question part and the implicit part.

Topic entity	Clarence Andrew Cannon
Question	What was Clarence Andrew Cannon’s occupation before becoming a lawyer?
Answer	teacher
Evidence - Main	“Clarence Andrew Cannon, occupation, teacher, start time, 1904, end time, 1908” [KB]
Evidence - Implicit	“Clarence Cannon, He earned an LL.B. and joined the bar in 1908.” [TEXT]
Topic entity	Robert Bosch GmbH
Question	Who was the chief executive officer at Robert Bosch GmbH before revenue reached €78.74 billion?
Answer	Volkmar Denner
Evidence - Main	“Robert Bosch GmbH, chief executive officer, Volkmar Denner, start time, 2012, end time, 2021” [KB]
Evidence - Implicit	“Robert Bosch GmbH, Revenue, €78.74 billion (2021)” [INFOBOX]
Topic entity	Carlos Alberto Torres
Question	Which national football team did Carlos Alberto Torres manage before joining Flamengo?
Answer	Oman national football team
Evidence - Main	“Carlos Alberto Torres, Managerial career, 2000–2001, Oman” [INFOBOX]
Evidence - Implicit	“Carlos Alberto Torres, Managerial career, 2001–2002, Flamengo” [INFOBOX]
Topic entity	Alan Page
Question	What hall of fame did Alan Page become a member of while serving as Associate Justice of the Minnesota Supreme Court?
Answer	College Football Hall of Fame
Evidence - Main	“Alan Page, In 1993, he was inducted into the College Football Hall of Fame.” [TEXT]
Evidence - Implicit	“Alan Page, Associate Justice of the Minnesota Supreme Court, In office January 4, 1993 – August 31, 2015” [INFOBOX]

- **Generative LMs.** We compare with **INSTRUCTGPT** [155] (“text-davinci-003”) and **GPT4** [153] (“gpt-4”). We tried different prompts, and found the following to perform best:

Please answer the following question by providing the crisp answer entity, date, year, or number.

For computing P@1, we check whether the generated answer string matches with the label or any alias of the gold answer. If this is the case, P@1 is 1, else 0. Other (ranking) metrics are not applicable for GLMs.

- **Temporal QA methods.** We also compare with state-of-the-art methods for temporal QA: **TEMPOQR** (TempoQR-Hard) [138], **CRONKGQA** [182], and **EXAQT** [93].

Finally, we show results for a variant of our approach, which does *not prune out* evidence that is temporally-inconsistent with the time constraint (i.e., drops the temporal pruning component). We term this variant **Un-FAITH**.

5.5.4 Configurations

Wikidata [220] is used as the KB for FAITH and all baselines. We use Wikipedia text, tables and infoboxes as additional information sources for methods operating over heterogeneous sources. For BART we use the base model³. When fine-tuning BART (for the SR generation and generation of intermediate questions), we use AdamW as optimizer with a learning rate of 5×10^{-5} , batch size of 10, weight decay of 0.01, 5 epochs and 500 warm-up steps. In the implicit question resolver, we choose the candidate at rank 1 as the answer for intermediate questions.

In our ERS stage (Section 5.3.2), we score all information snippets (known to be temporally faithful) with our re-ranker, and consider the top-100 in our answering stage. During retrieval, we set the k parameter of CLOCQ to 5. For fine-tuning the re-ranker on this data, we use AdamW

³<https://huggingface.co/facebook/bart-base>

Table 5.4: Main results comparing the performance of FAITH against baselines on the *test* sets of TIQ and TIMEQUESTIONS. **Bold** indicates the best performance per column, and the second best performance is underlined.

Benchmark → Method ↓	TIQ			TIMEQUESTIONS		
	P@1	MRR	Hit@5	P@1	MRR	Hit@5
INSTRUCTGPT [155]	0.237	n/a	n/a	0.224	n/a	n/a
GPT4 [153]	0.236	n/a	n/a	0.306	n/a	n/a
UNIQRN [160]	0.236	0.255	0.277	0.331	0.409	0.538
UNI-K-QA [154]	0.425	0.480	0.540	0.424	0.453	0.486
EXPLAIGNN [34]	0.446	0.584	<u>0.765</u>	0.525	0.587	<u>0.673</u>
TEMPOQR [138]	0.011	0.018	0.022	0.438	0.465	0.488
CRONKQQA [182]	0.006	0.011	0.014	0.395	0.423	0.450
EXAQT [93]	0.232	0.378	0.587	<u>0.565</u>	<u>0.599</u>	0.664
FAITH (Proposed)	0.491	0.603	0.752	0.535	0.582	0.635
Un-FAITH	<u>0.459</u>	0.604	0.799	0.571	0.640	0.724

as optimizer with a learning rate of 2×10^{-5} , batch size of 16, weight decay of 0.01, 4 epochs, and a warm-up ratio of 0.1. For EXPLAIGNN, we retain the original settings and parameters for optimization, and employ one pruning iteration (from 100 to 20 evidence nodes) and then predict the answer (from a graph with 20 evidence nodes).

We follow an epoch-wise evaluation strategy for each module and baseline, and take the version with the best performance on the respective dev set. All training processes and experiments are run on a single GPU (NVIDIA Quadro RTX 8000, 48 GB GDDR6).

Training FAITH on TIQ. FAITH requires train questions asking for temporal values to answer intermediate questions. Such questions exist in TIMEQUESTIONS, but our new TIQ benchmark only has implicit questions (by design). We thus generate intermediate questions on the train and dev sets, using the implicit questions as input (similar to the case in Section 5.3.1). For these intermediate questions, the gold answer is the temporal value of the implicit part as annotated in the TIQ benchmark, resulting in (question, temporal value) pairs. If the answer type of an intermediate question is a time interval, we create two questions asking for “*start date*” and “*end date*” respectively, as in Section 5.3.1. We obtain 7,723 such pairs from the TIQ train set and 2,542 questions from the dev set.

5.6 Experimental Results and Insights

5.6.1 Key Findings

The end-to-end answering performance of FAITH and baselines on the TIMEQUESTIONS and TIQ benchmarks are shown in Table 5.4.

FAITH outperforms baselines on TIQ. The main insight from Table 5.4 is that FAITH surpasses all baselines on the TIQ dataset for P@1, which is the most relevant metric, demonstrating the benefits of our proposed method for answering implicit temporal questions. Temporal QA methods operating over KBs lack the required coverage on the TIQ dataset, and perform worse than general-purpose QA methods operating over heterogeneous sources. Our general-purpose QA system EXPLAIGNN comes close to the performance of FAITH, and even slightly improves

Table 5.5: Comparing the faithfulness of FAITH and Un-FAITH for correct answers, and how often temporal constraints are violated or ignored.

Benchmark →	TIQ		TIMEQUESTIONS	
Method ↓	Faithful	Temporally Unfaithful	Faithful	Temporally Unfaithful
FAITH	0.95	0.00	0.94	0.01
Un-FAITH	0.90	0.08	0.87	0.13

on the Hit@5 metric. Note, however, that EXPLAINNN and all other baselines do not verify that temporal constraints are met during answering. Thus, the most prominent among answer candidates may simply be picked up, even if no temporal information is provided or matched. Such possibly “accidental” and *unfaithful* answers are, by design, not considered by FAITH.

Trade-off between faithfulness and answering performance. Results for Un-FAITH illustrate the effect of this phenomenon on our approach: especially the MRR and Hit@5 results are substantially improved. Consequently, Un-FAITH outperforms all competitors on TIMEQUESTIONS. However, its answers are not always faithfully grounded in evidence sources. These results emphasize the trade-off between faithfulness and answering performance.

FAITH shows robust performance on TIMEQUESTIONS. FAITH also shows strong performance on the TIMEQUESTIONS benchmark, on which it outperforms all baselines on P@1, except for EXAQT. This indicates the robustness of FAITH across different datasets. Existing methods for temporal QA show major performance gaps between the two benchmarks: the P@1 of the strongest method on TIMEQUESTIONS, EXAQT, substantially drops from 0.565 at P@1 to 0.232 on the TIQ benchmark. Note that all methods are trained on the specific benchmark, if applicable.

GLMs fall short on temporal questions. Another key insight from Table 5.4 is that current GLMs are clearly not capable of answering temporal questions. INSTRUCTGPT and GPT4 can merely answer $\simeq 23\text{-}30\%$ of the questions correctly, and are constantly underperforming FAITH and baselines operating over heterogeneous sources. One explanation is that reasoning with continuous variables, such as time, is a well-known weakness of GLMs [43].

5.6.2 Faithfulness Evaluation

Our main results in Table 5.4 indicate that ignoring the temporal condition in the question can yield improvements on automatic metrics (compare P@1 performance of FAITH versus Un-FAITH on TIMEQUESTIONS). However, we observe that this can lead to critical failure cases of QA systems and sometimes boils down to lucky guesses of the answer based on priors (e.g., prominence of an answer candidate).

FAITH refrains to answer in absence of consistent evidence. If there is no temporal information associated with the evidence of candidate answers, or the temporal information does not satisfy the temporal constraint, FAITH will refuse answering the question. For example, for the question “*Who did Lady Jane Grey marry on the 25th of May 1533?*”, there is no answer satisfying the temporal constraint because *Lady Jane Grey* did not marry anyone *on the 25th of May 1533*, since she was only born four years later in 1537. As a result, after processing the question in the understanding stage and identifying that there is no temporally consistent

answer, FAITH refrains from answering the question. However, all of the baselines provide an answer to the question, without indicating that the temporal constraint is violated.

Since such questions without a temporally-consistent answer are not available at large scale, we randomly sample 500 explicit questions from TIMEQUESTIONS, and replace the temporal value with a random date (e.g., “12 October 6267”), such that none of the resulting questions has a temporally-consistent answer. As expected, the competitors still provide answers⁴. In contrast, FAITH successfully refrained from answering for 467 of the 500 questions (93.4%). Upon investigating the failure cases, we noticed that the date recognition identifies four-digit numbers as years matching with the constraint (e.g., “6267” was detected as a year in the infobox entry “Veysonnaz, SFOS number, 6267”).

Fallback to Un-FAITH. As discussed before, FAITH refrains from answering in absence of answer candidates matching the temporal criteria. Completely refraining from answering could also be sub-optimal: the user might have made a typo (e.g., “May 1533” instead of “May 1553”). We investigate a variant that falls back to Un-FAITH in such scenarios, which could be indicated to end users with an appropriate warning. Performance on both datasets was slightly improved: the P@1 metric increased from 0.491 to 0.492 on TIQ and from 0.535 to 0.539 on TIMEQUESTIONS. We further investigate to fall back to Un-FAITH in case FAITH answered incorrectly. The P@1 metric was improved substantially on both datasets: from 0.491 to 0.622 on TIQ and from 0.535 to 0.653 on TIMEQUESTIONS.

Manual analysis. Finally, we investigate the faithfulness of *correct* answers provided by FAITH and Un-FAITH, to understand how often the question is answered correctly even though the evidence is not faithful to the question. To analyze this qualitatively, we randomly selected 100 questions (from each benchmark) for which both FAITH and Un-FAITH answered correctly, and then manually verified the faithfulness of the answer, based on the definition in Section 5.2. Results are shown in Table 5.5. FAITH provides faithful answers and evidence in 95% (TIQ) / 94% (TIMEQUESTIONS) of the time. By design, answers are faithful to the temporal constraints in the question (except for one question which specifies two different temporal constraints). In comparison, Un-FAITH violates or ignores the temporal condition in 8% (TIQ) / 13% (TIMEQUESTIONS) of the cases.

For example, to answer the question “What movies starring Taylor Lautner in 2011?” (answer: `Abduction`), the evidence for FAITH is “Taylor Lautner, Year is 2011, Title is Abduction, Role is Nathan Harper” (from a Wikipedia table), while the evidence for Un-FAITH is “Abduction, cast member, Taylor Lautner” (from the KB). Even though both pieces of evidence mention the correct answer `Abduction`, Un-FAITH fails to satisfy the temporal constraint (“in 2011”) with its evidence, potentially guided solely by the prominence of the answer candidate.

5.6.3 Analysis

Integrating heterogeneous sources is decisive. We further investigated the effect of integrating heterogeneous sources into FAITH, and tested giving each individual information source independently, and their pairwise combinations as input, in comparison to the default setting with “All sources”. Results are depicted in Table 5.6. Overall, each information source contributes to the performance of FAITH, and integrating more information sources consistently enhances all metrics. This observation matches with what we observed for CONVINSE (in Section 3.5.1) and EXPLAINNN (in Section 4.4.1), and demonstrates that our key motivation in this dissertation, integrating heterogeneous sources into QA systems, is indeed reasonable.

⁴Except for the GLMs for which we are not able to investigate the behavior at scale, since they would often generate longer texts.

Table 5.6: Ablation study using different source combinations as input for FAITH on *dev* sets. Note that FAITH is trained using *all sources* as input for all cases.

Benchmark → Method ↓	TIQ			TIMEQUESTIONS		
	P@1	MRR	Hit@5	P@1	MRR	Hit@5
KB	0.293	0.368	0.468	0.425	0.464	0.513
Text	0.194	0.262	0.351	0.224	0.269	0.320
Infoboxes	0.169	0.223	0.296	0.093	0.117	0.149
Tables	0.032	0.057	0.083	0.078	0.094	0.114
KB + Text	0.429	0.527	0.649	0.520	0.567	0.626
KB + Tables	0.299	0.379	0.480	0.435	0.479	0.536
KB + Infoboxes	0.384	0.488	0.634	0.443	0.487	0.543
Text + Tables	0.196	0.267	0.362	0.252	0.298	0.350
Text + Infoboxes	0.283	0.372	0.490	0.251	0.299	0.355
Tables + Infoboxes	0.179	0.244	0.331	0.143	0.174	0.208
All sources	0.497	0.610	0.756	0.538	0.583	0.639

Table 5.7: Ablation studies of FAITH on dev sets.

Benchmark → Method ↓	TIQ	TIMEQUESTIONS
	P@1	P@1
FAITH	0.497	0.538
w/o temporal pruning	0.443	0.573
w/o implicit question resolver	0.467	0.559
w/o GNN-based answering	0.316	0.399

Ablation studies. We tested variations of our pipeline on the dev sets. Table 5.7 shows results for Un-FAITH (w/o temporal pruning), results without the implicit time resolver, and results with a Seq2seq LM for answering (we used BART) instead of the EXPLAINNN-based approach. Using our GNN-based answering plays a crucial role, and enhances not only answering performance, but also explainability. The implicit question resolver is decisive on TIQ, but slightly decreases performance on TIMEQUESTIONS. Un-FAITH also shows strong performance on the dev sets. However, all modules contribute to the explainability and faithfulness of our approach.

Anecdotal examples. Table 5.8 and Table 5.9 show sample cases for which FAITH provided the correct answer from the TIQ and TIMEQUESTIONS benchmarks, and illustrate the answer derivation process providing traceable evidence for end users.

Error analysis. To better understand failure cases, we conducted an error analysis measuring the *answer presence* throughout the pipeline. Answer presence is measured as the fraction of questions for which the gold answer is present in the candidate set of information snippets [31]. We identified the following error cases and list their percentage among all failure cases (*failure cases* for TIQ and TIMEQUESTIONS, respectively):

- (i) the answer was not found in the initial retrieval stage (3.14% / 29.89%),
- (ii) the answer is lost during temporal pruning (22.00% / 25.81%),
- (iii) the answer is lost during scoring/graph shrinking (8.45% / 10.33%),
- (iv) the answer is not considered among top-5 answers (15.13% / 12.47%),
- (v) the answer is among top candidates but not at rank 1 (51.28% / 21.51%).

Table 5.8: Anecdotal example that FAITH answered correctly on the TIQ benchmark. “Evidence” shows the supporting information snippets along with their source provided in brackets. The part mentioning the predicted answer is in **bold**, and the detected temporal values are underlined. We also show the answering process of the intermediate question, which can be used by end users to verify the entire answer derivation of the system.

Question	<i>After managing FC Nantes, which football club did Antoine Raab take on next?</i>	
Answer	Stade Lavallois	
SR	<div style="border: 1px solid black; padding: 5px; background-color: #fff9c4;"> <p>Question entity: <i>Antoine Raab, FC Nantes football</i> Question predicate: <i>After managing which club did take on next</i> Expected answer type: <i>association football club</i></p> <hr/> <p>Temporal signal: <i>after</i> Temporal category: <i>implicit</i> Temporal value: <u>[1946, 1949]</u></p> </div>	
Evidence	“ <i>Antoine Raab, Managerial career, 1949–1950, Stade Lavallois.</i> ” [INFOBOX]	
Intermediate questions	(i) <i>When Antoine Raab managed FC Nantes start date?</i> (ii) <i>When Antoine Raab managed FC Nantes end date?</i>	
Answers (to int. questions)	(i) 1946 (ii) 1949	
SRs (for int. questions)	<div style="border: 1px solid black; padding: 5px; background-color: #fff9c4;"> <p>Question entity: <i>FC Nantes, Antoine Raab</i> Question predicate: <i>When managed start date</i> Expected answer type: <i>year</i></p> <hr/> <p>Temporal signal: <i>_</i> Temporal category: <i>non-implicit</i> Temporal value: <i>_</i></p> </div>	<div style="border: 1px solid black; padding: 5px; background-color: #fff9c4;"> <p>Question entity: <i>FC Nantes, Antoine Raab</i> Question predicate: <i>When managed end date</i> Expected answer type: <i>year</i></p> <hr/> <p>Temporal signal: <i>_</i> Temporal category: <i>non-implicit</i> Temporal value: <i>_</i></p> </div>
Evidence (for int. questions)	(i, ii) “ <i>Antoine Raab, Managerial career, 1946–1949, FC Nantes.</i> ” [INFOBOX] (ii) “ <i>Antoine Raab, After the liberation of Nantes in 1944 Raab joined FC Nantes and played for the club until 1949.</i> ” [TEXT]	

Especially the fine-grained answer ranking (case v) leaves scope for improvements, and accounts for more than half of our error cases on the TIQ benchmark.

Relaxed temporal pruning. We also compute the effect of the number of answer candidates in the implicit question resolver on the end-to-end performance (our default setting considers only the top-1 answer). Figure 5.5 shows the results, varying the number of candidates k from 1 to 20. On TIMEQUESTIONS, we observe that P@1 improves gradually, as k increases, until the set of candidate snippets converges, resulting in a stable P@1. On TIQ, we only observe an improvement of the P@1 metric when increasing k to 2. As k increases further, more noisy candidate snippets are considered, resulting in a lower performance.

Answer presence analysis. We measure the answer presence after the initial heterogeneous retrieval, and the effect of the subsequent pruning and scoring steps.

We conduct the analysis on the test sets of TIQ/TIMEQUESTIONS. The answer presence after the heterogeneous retrieval is 0.984/0.861. After temporal pruning the answer presence drops to 0.872/0.741. Note that in this step, temporally inconsistent evidence is pruned out, enhancing the faithfulness of the approach. In the evidence scoring stage (based on a cross-encoder), the answer presence is mostly retained (0.867/0.726). Inside the EXPLAINNN pipeline, the answer presence after evidence pruning is 0.829/0.693. In general, as discussed in the error analysis, the key source of error is the fine-grained answer ranking step.

5 Temporal Question Answering

Table 5.9: Anecdotal example that FAITH answered correctly on the TIMEQUESTIONS benchmark. “Evidence” shows the supporting information snippets along with their source provided in brackets. The part mentioning the predicted answer is in **bold**, and the detected temporal values are underlined>.

Question	What award did Thomas Keneally receive in the year 1982?
Answer	Booker Prize
SR	<div style="border: 1px solid black; padding: 5px; background-color: #fff9c4;"> <p>Question entity: <i>Thomas Keneally</i> Question predicate: <i>What award did receive in the year 1982</i> Expected answer type: <i>science award</i></p> <hr/> <p>Temporal signal: <i>overlap</i> Temporal category: <i>non-implicit</i> Temporal value: <i>1982</i></p> </div>
Evidence	<p>“Man Booker Prize, winner, <i>Thomas Keneally</i>, point in time, <u>1982</u>, for work, <i>Schindler’s Ark</i>.” [KB] “<i>Thomas Keneally</i>, Awards is Booker Prize, is <i>Schindler’s Ark</i>, winner <u>1982</u>.” [TABLE] “<i>Thomas Keneally</i>, He is best known for his non-fiction novel <i>Schindler’s Ark</i>, the story of Oskar Schindler’s rescue of Jews during the Holocaust, which won the Booker Prize in <u>1982</u>.” [TEXT]</p>

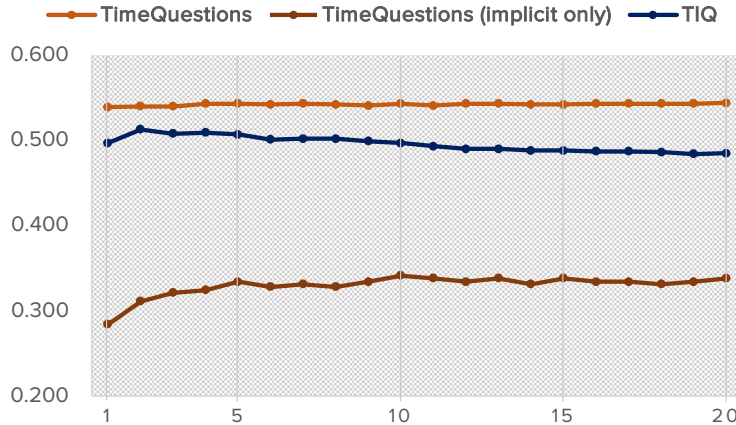


Figure 5.5: P@1 of FAITH when considering top- k answers for the generated intermediate question(s) of implicit questions.

FAITH performance on questions from mixed sources. TIQ has questions originating from different source combinations (see Figure 5.4). Table 5.10 shows how this affects the performance of FAITH (all sources are used as input for answering).

Results demonstrate that FAITH can deal with all of these questions, and there is no combination for which FAITH completely fails, indicating that FAITH successfully incorporates all of the heterogeneous sources during answering. FAITH shows the best performance on questions from text (P@1 of 0.598), KB (P@1 of 0.582), and text and infoboxes (P@1 of 0.573), and the worst performance on questions from text and KB (P@1 of 0.359).

Temporal signal accuracy. We measure the accuracy of the generated temporal signals (*before*, *after* or *overlap*) in our SR construction (ground-truth annotations are available in both benchmarks). On TIMEQUESTIONS the accuracy is 93.0%, and on TIQ it is 97.8%. The high accuracy scores indicate that our approach of generating the temporal signal is feasible.

Temporal category accuracy. We also measure the accuracy in predicting the temporal

Table 5.10: Performance of FAITH (with all sources) on questions from different source combinations (source for main question part; source for implicit part) in the TIQtest set .

Question-source combination	P@1	MRR	Hit@5
[Text; Infobox] (157 questions)	0.573	0.644	0.752
[KB; Text] (378 questions)	0.519	0.634	0.770
[Infobox; KB] (43 questions)	0.395	0.534	0.721
[Infobox; Text] (225 questions)	0.476	0.595	0.760
[KB; Infobox] (251 questions)	0.478	0.598	0.753
[Text; KB] (142 questions)	0.359	0.509	0.725
[KB; KB] (127 questions)	0.582	0.676	0.807
[Text; Text] (306 questions)	0.598	0.679	0.787
[Infobox; Infobox] (376 questions)	0.407	0.529	0.689
All test questions (2,000 questions)	0.497	0.610	0.756

category, differentiating between implicit and non-implicit questions. The accuracy is 98.9% on the TIMEQUESTIONS benchmark, and 100% on TIQ which has only implicit questions.

Performance of implicit question resolver. The performance of the implicit question resolver is crucial for correctly answering implicit questions, as the resulting temporal values are directly used for pruning out evidence in the remainder of the pipeline. As there might be several temporal values per question (due to multiple intermediate questions), we measure macro-averaged precision, recall, and F1-score. We conduct experiments on the test set of TIQ, which has the ground-truth temporal values.

When using the top-1 answer candidate per intermediate question (default setting), precision is 0.537, recall is 0.562 and F1-score is 0.525. When increasing the number of candidates to 3, precision is 0.294, recall is 0.714 and F1-score is 0.401. With the top-5 candidates, precision is 0.196, recall is 0.773 and F1-score is 0.304.

Recall improves as we consider more answers, since the resulting explicit temporal constraint is relaxed. Hence, the evidence retained is noisier and may not satisfy the user-intended temporal constraints any more. This can negatively affect the system’s faithfulness, and even the performance (as shown in Figure 5.5).

5.7 Related Work

Temporal QA. Prior work that specifically targets temporal QA [25, 29, 45, 90, 93, 94, 129, 138, 146, 182, 186, 187, 227, 229, 241], can be divided into work using a KB (e.g., [93, 138, 146]), and work using text (e.g., [25, 150]). *Methods operating over KBs*, include template-based [90, 45, 146], KB-embedding-based [182, 229, 29, 138], and graph-based approaches [93, 187, 241]. *Methods using textual inputs* typically involve an extractive or generative reader [25, 150].

EXAQT [93] proposed a method based on GNNs that operates on a KB subgraph which is enhanced with temporal facts. CRONKGQA [182] approaches the problem by obtaining embeddings of answer candidates and temporal values individually. These embeddings are then combined and used for scoring answer candidates directly. TEMPOQR [138] takes a similar approach and augments a question embedding with answer candidate, context, and temporal encodings. Again, this latent encoding is used for predicting an answer score for the embedded answer candidate. These three methods [93, 138, 182] represent the state-of-the-art on temporal QA. However, temporal constraints are handled solely in the latent space, without explicitly (or

faithfully) pruning out temporally inconsistent answer candidates. Other approaches are based on handcrafted rules, and hence bound to fail for unseen question patterns (e.g., TEQUILA [90]).

Further, different from general-purpose QA, there has not yet been work on temporal QA that combines heterogeneous information sources for improving the answer coverage.

Temporal KBs. There is substantial work on temporal KBs [16, 63, 113, 140, 156, 225, 230], to assign temporal scopes to KB facts. Advances on the KB itself benefits QA, including FAITH, but is an orthogonal direction.

Explainable QA. More traditional QA methods would obtain the answer to a question using logical forms over a KB [2, 188]. Such methods are inherently interpretable, since the logical form provides insights in the system understanding. However, such methods cannot be adapted to a setting in which heterogeneous sources are used for improving the answer coverage.

5.8 Conclusion

The general objective of the FAITH project was to provide *faithful answers* that align with the time constraints specified by the user. Our approach demonstrates strong performance on temporal questions, including challenging cases with implicit temporal constraints. Notably, we identified a trade-off between faithfulness and answering performance: an *unfaithful* variant of our approach improves over FAITH on non-implicit temporal questions, which often include explicit temporal constraints. One explanation is that Un-FAITH often derives an answer based on evidence with inconsistent temporal information, or even without any temporal information, relying on the prominence of answers – this strategy enables improvements of the answering performance in some cases.

This trade-off brings up the fundamental question whether to prioritize pure performance, which may be achieved without satisfying the specified user intent, or rather opt for slight performance deficits, with *faithful* answers and convincing traces of the answer derivation. Similar questions can be raised for GLMs that are commonly used without any external knowledge – it is totally unclear to what extent user constraints are matched within the GLM’s inference, even if the generated output says so. In the context of temporal QA, we found that recent GLMs are falling far behind the state-of-the-art approaches, and ignorance of temporal constraints could be one of the reasons (this is hard/not possible to verify in experiments).

Table 5.11: Instruction and ICL examples for rephrasing pseudo-questions into natural questions, leveraging in-context learning.

<p>Please rephrase the following input question into a more natural question.</p> <p><i>Input:</i> What album Sting (musician) was released, during, Sting award received German Radio Award?</p> <p><i>Question:</i> which album was released by Sting when he won the German Radio Award?</p> <p><i>Input:</i> What human President of Bolivia was the second and most recent female president, after, president of Bolivia officeholder Evo Morales?</p> <p><i>Question:</i> Which female president succeeded Evo Morales in Bolivia?</p> <p><i>Input:</i> What lake David Bowie He moved to Switzerland purchasing a chalet in the hills to the north of , during, David Bowie spouse Angela Bowie?</p> <p><i>Question:</i> Close to which lake did David Bowie buy a chalet while he was married to Angela Bowie?</p> <p><i>Input:</i> What human Robert Motherwell spouse, during, Robert Motherwell He also edited Paalen's collected essays Form and Sense as the first issue of Problems of Contemporary Art?</p> <p><i>Question:</i> Who was Robert Motherwell's wife when he edited Paalen's collected essays Form and Sense?</p> <p><i>Input:</i> What historical country Independent State of Croatia the NDH government signed an agreement with which demarcated their borders, during, Independent State of Croatia?</p> <p><i>Question:</i> At the time of the Independent State of Croatia, which country signed an agreement with the NDH government to demarcate their borders?</p> <p><i>Input:</i> What U-boat flotilla German submarine U-559 part of, before, German submarine U-559 She moved to the 29th U-boat Flotilla?</p> <p><i>Question:</i> Which U-boat flotilla did the German submarine U-559 belong to before being transferred to the 29th U-boat Flotilla?</p> <p><i>Input:</i> What human UEFA chairperson, during, UEFA chairperson Sandor Barcs?</p> <p><i>Question:</i> Who was the UEFA chairperson after Sandor Barcs?</p> <p><i>Input:</i> What human Netherlands head of government, during, Netherlands head of state Juliana of the Netherlands?</p> <p><i>Question:</i> During Juliana of the Netherlands' time as queen, who was the prime minister in the Netherlands?</p>

5 Temporal Question Answering

Table 5.12: Instruction and ICL examples used to obtain the training data for generating intermediate questions, leveraging in-context learning.

TIMEQUESTIONS
<p>Generate an explicit question and answer type for the implicit part of the temporal input question.</p> <p><i>Input:</i> what position did djuanda kartawidjaja take after he was replaced by sukarano <i>Output:</i> when djuanda kartawidjaja replaced by sukarano date</p> <p><i>Input:</i> american naval leader during the world war 2 <i>Output:</i> when world war 2 time interval</p> <p><i>Input:</i> who became president after harding died <i>Output:</i> when harding died date</p> <p><i>Input:</i> who did luis suarez play for before liverpool <i>Output:</i> when luis suarez play for liverpool time interval</p> <p><i>Input:</i> which countries were located within the soviet union prior to its dissolution <i>Output:</i> when soviet union dissolution date</p> <p><i>Input:</i> who started the presidency earliest and served as president during wwii in the US <i>Output:</i> when wwii time interval</p> <p><i>Input:</i> who replaced aldo moreo as the minister of foreign affairs <i>Output:</i> when aldo moreo replaced as minister of foreign affairs date</p> <p><i>Input:</i> what did harry s truman work before he was president <i>Output:</i> when harry s truman president time interval</p>
TIQ
<p>Generate an explicit question and answer type for the implicit part of the temporal input question.</p> <p><i>Input:</i> Who was the second director of the Isabella Stewart Gardner Museum when it was built <i>Output:</i> When Isabella Stewart Gardner Museum was built time interval</p> <p><i>Input:</i> When Wendy Doniger was president of the Association for Asian Studies, what publishing house was she based in New York <i>Output:</i> When Wendy Doniger was president of the Association for Asian Studies time interval</p> <p><i>Input:</i> What administrative entity was Ezhou in before Huangzhou District became part of it <i>Output:</i> When Huangzhou District became part of Ezhou date</p> <p><i>Input:</i> After Bud Yorkin became the producer of NBC's The Tony Martin Show, who was his spouse? <i>Output:</i> When Bud Yorkin became the producer of NBC's The Tony Martin Show date</p> <p><i>Input:</i> What book did Ira Levin write that was adapted into a film during the same time he wrote the play Deathtrap <i>Output:</i> When Ira Levin wrote the play Deathtrap date</p> <p><i>Input:</i> What basketball team was Nathaniel Clifton playing for when his career history with the Rens began <i>Output:</i> When Nathaniel Clifton's career history with the Rens began time interval</p> <p><i>Input:</i> What team did Stevica Ristić play for before joining Shonan Bellmare? <i>Output:</i> When Stevica Ristić joining Shonan Bellmare time interval</p> <p><i>Input:</i> Which album was released by the Smashing Pumpkins after Mike Byrne joined the band <i>Output:</i> When Mike Byrne joined Smashing Pumpkins time interval</p>

6

Complex Question Answering

We found that iterative GNNs can substantially improve answering performance and enhance explainability in both EXPLAIGNN (Chapter 4) and FAITH (Chapter 5), compared to Seq2seq LMs. However, we also noticed that more than one third of errors stem from the fine-grained answer ranking. The goal in this chapter is to investigate whether large-scale GLMs, integrated into our pipeline (established in the previous chapters), can improve the answering effectiveness. Note that this might come at the cost of reduced *explainability*. We devise a unified pipeline, that integrates QU and ERS stages similar to previous chapters. Notably, the evidence scoring makes use of the iterative GNNs proposed in EXPLAIGNN. The final answering step utilizes a moderate-size GLM that generates the answer from a small set of evidence pieces, and enables answering more complex questions with comparisons or joins. For experiments, we release the COMPMIX benchmark, with 9K complex questions for heterogeneous QA.

All code and data are available at <https://qa.mpi-inf.mpg.de/quasar>.

Contents

6.1	Introduction	86
6.2	The QUASAR Methodology	87
6.2.1	Question Understanding (QU)	87
6.2.2	Evidence Retrieval and Scoring (ERS)	89
6.2.3	Heterogeneous Answering (HA)	90
6.3	The COMPMIX Benchmark	90
6.3.1	Existing Benchmarks and their Limitations	90
6.3.2	Construction of the COMPMIX Benchmark	91
6.3.3	Characteristics of the COMPMIX Benchmark	93
6.4	Experimental Setup	94
6.4.1	Benchmarks	94
6.4.2	Baselines	95
6.4.3	Metrics	96
6.4.4	Configurations	96
6.5	Experimental Results and Insights	96
6.5.1	Key Findings	96
6.5.2	Analysis	97
6.6	Related Work	100
6.7	Discussion	101

6.1 Introduction

Motivation. Recent GLMs [143, 257] alone are capable of accurately answering many *factoid* questions, simply from their pre-trained parametric memory that latently encodes huge text corpora and other online contents. However, this critically depends on the frequency of evidence in the underlying contents and the complexity of the information need. For example, asking for the “*MVP of the 2024 NBA season*” would likely return the correct answer *Nikola Jokic*, but questions asking for the “*highest-scoring German NBA player*” or the “*MVP of the 2024 German basketball league*” pose a big challenge. The reason is that GLMs alone do not easily recall information about not-so-popular or long-tail entities [103, 104, 199].

Retrieval-augmented generation (RAG) [62, 72, 116] addresses these bottlenecks. In addition to cleverly crafted prompts and few-shot examples, the GLM is provided with the results of an explicit retrieval step, like web search or knowledge base look-ups. The former is often necessary for freshness of answers, and the latter may help with long-tail entities and also mitigates the notorious risk of hallucinations [139]. Still, current RAG architectures are limited in how broad and how deep they tap into external sources. Popular AI assistants like Gemini or ChatGPT seem to primarily retrieve from the text of web pages (including Wikipedia articles), and academic research has additionally pursued knowledge augmentation by enhancing prompts with facts from large KBs (e.g., Wikidata).

An additional content modality that is still underexplored are *web tables*: a wide range of tabular data including HTML tables in web pages, spreadsheets and statistics, all the way to CSV and JSON datasets that are abundant on the Internet.

Examples. All three heterogeneous sources are crucial not only for answering different questions from different kinds of evidence, but also for combining multiple pieces of evidence from different source types to infer correct and complete answers. To illustrate the need for tapping into all sources, consider the following questions:

- q^1 : Which Chinese basketballers have played in the NBA?
 q^2 : Who was the first Chinese NBA player?
 q^3 : Which Chinese NBA player has the most matches for Miami Heat?

In the previous chapters of this dissertation, we established a principled pipeline for integrating heterogeneous sources into the answering process. However, questions q^1 and q^3 are still out-of-scope: q^1 requires creating a set of answers, while our approaches would either return a single answer (CONVINSE) or a ranked list of answers without cut-off (EXPLAIGNN, FAITH). For q^3 , the QA system needs to reason over a handful of evidence pieces for deriving the answer. Such operations are out-of-scope for our Seq2seq LMs and GNNs.

Approach. To this end, we aim to combine our established pipeline with a GLM for answer generation, in a RAG-based manner, and derive a unified approach for QA over heterogeneous sources. We call our method QUASAR (QUestion Answering over Heterogeneous Sources with Augmented Retrieval). The key idea is to leverage an EXPLAIGNN-like approach until the final answer prediction step (after the pruning iterations of the EXPLAIGNN GNNs). To derive the answer from the small set of evidence pieces, we utilize a GLM, which can combine information from the evidence pieces when generating the answer. This helps with answering questions like q^1 and q^3 . For q^1 , the GLM can generate an answer list based on the relevant information in the input¹. Question q^3 requires comparison of numerical information from only a few evidence

¹While Seq2seq LMs are, in principle, also capable of generating such answer lists, this would require large-scale training data; GLMs, on the other hand, obtain such capabilities from their pre-training.

pieces for deriving the answer. While current GLMs generally struggle with numerical data, such small-scale inputs are already in scope (there are only two relevant players for q^3).

Based on our pipeline in chapters 3, 4 and 5, QUASAR has three stages:

- (i) automatically casting user questions into a structured representation of the information need, which is then used to guide
- (ii) retrieval from heterogeneous sources, and judicious ranking of search results, with multiple rounds of re-ranking and pruning empowered by our EXPLAIGNN GNNs, followed by
- (iii) answer generation by a GLM with answers grounded in traceable evidence.

We consider re-ranking based on cross-encoders as an alternative in the second step.

To test our architecture, we make the case for a benchmark that *inherently requires* the usage of a mixture of information sources, as a natural testbed for evaluating QA systems operating over heterogeneous sources. To this end, we release COMPMIX (Complete Questions over a Mixture of Sources), a *crowdsourced* QA benchmark with questions that require heterogeneous sources for answering. The dataset has 9,410 questions created by *humans* from *five different domains*: books, movies, music, TV series and soccer.

Finally, this chapter also presents experiments with existing benchmarks for temporal QA and complex QA, covering various flavors of questions. We focus on one-shot questions; conversational QA is out of scope here, but QUASAR itself is also applicable to this case. Our experiments demonstrate that our approach is competitive, on par with large-scale GLMs and often better, while being several orders of magnitude lower in computational and energy cost. The experimental findings also highlight that question understanding, with structured representation of user intents, and iterative re-ranking of evidence, are crucial for good performance. Finally, integrating heterogeneous sources consistently boosts effectiveness. These findings are in-line with our observations in previous chapters, and demonstrate the feasibility of our ideas, and their applicability to various QA scenarios.

Contributions. The salient contributions of this work are:

- Proposing a unified system architecture for RAG-based question answering over heterogeneous sources, with strong points regarding both effectiveness (i.e., answer quality) and efficiency (i.e., computational cost).
- Releasing COMPMIX, a new benchmark for QA over heterogeneous sources, with 9K questions that exhibit complex phenomena, like temporal conditions, multiple entities and relations, aggregations and comparisons.

6.2 The QUASAR Methodology

An overview of the QUASAR system is illustrated in Figure 6.1. It generally follows the same three stages introduced in CONVINSE (Chapter 3), and utilized in EXPLAIGNN (Chapter 4) and FAITH (Chapter 5). However, the evidence scoring and pruning is performed by iterative GNNs, that were introduced in Section 4.2, or alternatively by cross-encoders.

6.2.1 Question Understanding (QU)

To prepare the retrieval from different kinds of sources, including a KB, ad-hoc tables and text documents, it is useful to analyze and decompose the user question. Thus, we follow the same procedure as in Section 4.2.1 for deriving a *structured representation*.

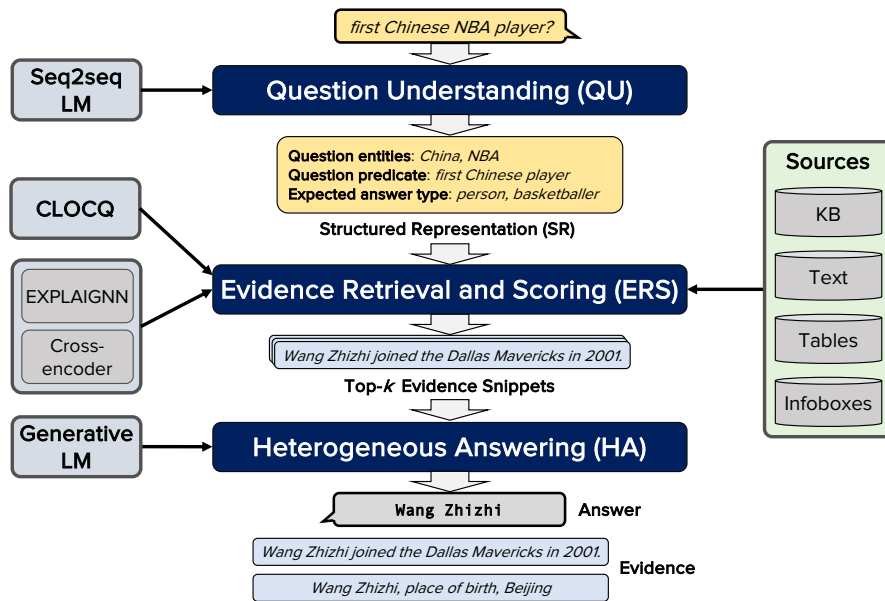


Figure 6.1: Overview of the QUASAR system.

The SR consists of the following slots:

- *Entities*: the salient entities in the question,
- *Predicate*: phrases that indicate which relation the user is interested in, and
- *Expected answer type*: the expected answer type (or types when considering different levels of semantic refinement).

The ideal SR for example question q^2 would look like:

Question entities: *China, NBA*
Question predicate: *first Chinese player*
Expected answer type: *person, basketballer*

In general the SR is a frame with faceted cues as slots, or equivalently, a concise set of key-value pairs. So the SR could easily be extended with additional keys, depending on the specific question at hand. An example is our specialized SR for temporal questions presented in Section 5.3.1, which has dedicated slots for temporal information needs.

The SR is generated via a fine-tuned BART [115] model², that is trained using distant supervision (see Section 3.2.1). We utilize hallucination prevention as proposed in Section 4.2.1.

Alternatively, one could generate (question, SR)-pairs via ICL, similar as for intermediate questions in the FAITH project (Section 5.3.1). This would be the obvious approach if one were to extend the SR with additional slots (e.g., for geo-spatial constraints). A combination of training data for different question types (and thus different sets of SR-slots) could then be used to train a model for automatically choosing the appropriate slots to fill. Further, although this chapter focuses on single-shot questions, the same architecture is also geared for conversational QA. Again, this would require appropriate training data, derived either via distant supervision (Section 3.2.1), or via ICL.

²<https://huggingface.co/facebook/bart-base>

6.2.2 Evidence Retrieval and Scoring (ERS)

This stage consists of two steps: (i) the *evidence retrieval*, which retrieves and verbalizes evidence pieces from heterogeneous sources, and (ii) the *evidence scoring*, that iteratively re-ranks the verbalized evidence pieces for deriving the most relevant information snippets.

6.2.2.1 Evidence Retrieval (ER)

The evidence retrieval follows the same steps as in EXPLAIGNN (Section 4.2.2), including the restriction of disambiguations to mentions in the entity slot. CLOCQ links entities in the question entities' slot, and retrieves KB-facts as well as Wikipedia text, infoboxes and tables for these entities. Text is split into sentences, and other evidence is verbalized for unified treatment. We can think of this entire pool of evidence as an on-the-fly corpus of potentially relevant sentences.

6.2.2.2 Evidence Scoring (ES)

BM25 scoring. Overall, the ER step compiles a substantially large set of evidence, typically several thousands of evidence pieces. Therefore, we first restrict this pool to a subset of high-scoring pieces, like the top-1000. For this initial pruning, simple BM25 scoring is applied, with the SR as query and the evidence pieces as documents. By default, we treat all evidence pieces uniformly with global scoring, independent of their original source.

For reducing the noise, costs of computation and energy consumption of the GLM invocation, this pool of top-1000 evidence pieces is further reduced. To this end, we leverage lightweight techniques for iteratively reducing the top-1000 pieces to a small subset, say top-30 or top-10, that can be fed into an GLM at much lower cost, and allow for investigation of all explicit information by end users. The difficulty is, of course, to do this without losing good evidence and reducing answer presence [31]. For scoring evidence snippets, we either use the pruning GNNs from our EXPLAIGNN work (Section 4.2), or cross-encoders [122].

EXPLAIGNN-based ES. Given a large pool of evidence pieces from all sources, a bipartite graph is constructed:

- *nodes* being evidence pieces or entities that occur in these pieces, and
- *edges* connecting an evidence piece and an entity if the entity occurs in the evidence.

The task for the GNN is to jointly score the evidence and the entity nodes in a multi-task learning setup. The latter are the *answer candidates*, and the evidence should give a *faithful explanation* for the answer. We build on our earlier work on explainable QA in Chapter 4.

The node encodings are initialized with cross-encodings for node contents and the SR of the question. The inference iteratively adjusts the encodings based on message passing from neighboring nodes. The EXPLAIGNN-GNN is trained via weak supervision from question-answer pairs: evidence nodes are labeled as relevant if they are connected to a gold answer. Refer to Section 4.2 for further technical details.

QUASAR invokes the GNN in multiple rounds, iteratively reducing top- k to top- k^* nodes with $k^* \ll k$. In practice, we consider two rounds: re-ranking top-1000 and pruning to top-100, and then reducing to top-30 or top-10, which are passed to the answer generation stage. Note that the GNNs in EXPLAIGNN are of small-scale, dominated by the size of the encoder LM, so that its computational costs at inference-time are much smaller than those of a GLM.

Cross-encoder-based ES. An alternative to our GNN inference is to employ a cross-encoder for scoring and re-ranking the evidence pieces. In our case, the cross-encoder is fed with the question SR and an evidence piece, and a feed-forward layer is used to derive a scalar. This process is

repeated for all evidence pieces, to derive one score for each of them. In our experiments, we make use of two different cross-encoders, both trained on the MS MARCO benchmark for passage retrieval [6], and fine-tuned on the respective benchmark (leveraging the same weak supervision data as for the GNNs), the difference being in model size. We use the smaller model³ to reduce top-1000 to top-100, and the larger model⁴ to further prune from top-100 to top-30.

6.2.3 Heterogeneous Answering (HA)

The answering stage invokes a GLM in a retrieval-augmented manner. We employ a GLM of moderate size, specifically a fine-tuned LLAMA model with 8B parameters⁵. The GLM is then provided with the following information:

- the SR to represent the information need, and
- the top-30 (or other top- k^* with small k^*) evidence pieces.

The specific prompt used is:

```
SR: <SR> Evidence: <evidence pieces>
```

We considered adding an instruction to the prompt, such as “*answer this question solely based on the provided evidence snippets*”, for instructing the GLM to rely on the provided evidence and not its own parametric knowledge. However, this turned out to be ineffective.

By the previous down-filtering of the original pool of relevant evidence pieces, this last step has affordable cost in terms of computation time and energy consumption.

Training. We fine-tune the HA stage with the top-30 evidence pieces as input. Training instances, for which the correct answer is not mentioned in the retrieved evidence, are dropped in this step. Thus, through the fine-tuning process, the model learns the important role of grounding evidence for RAG-based QA.

Explanations. The top-30 evidence pieces can be used to provide users with explanations of answers. Optionally, these could be reduced further for comprehensibility. Since we can infer which evidence pieces in the input mention the annotated ground-truth answers, our method could provide such *answering evidence* as well (cf. [61]). However, unlike the answer derivation within EXPLAIGNN, these explanations are not causal, and the model could as well have used solely its parametric knowledge, ignoring the retrieval results.

6.3 The COMPMIX Benchmark

6.3.1 Existing Benchmarks and their Limitations

There are many datasets for KB-QA (like WEBQUESTIONS [10] or SIMPLEQUESTIONS [13]), Text-QA (like SQUAD [166] or NATURALQUESTIONS [111]), and Table-QA (like WIKITABLEQUESTIONS [157] or NQ-TABLES [76]). However, these benchmarks were created with the intention of having a specific source for answering, which contains almost all answers to the questions. This restricts their utility as a testbed for heterogeneous QA. Thus, existing work on heterogeneous QA, being forced to rely on these benchmarks, would often remove significant chunks of

³<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-4-v2>

⁴<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2>

⁵<https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>

Table 6.1: Comparison of benchmarks for QA over heterogeneous sources.

Dataset	KB	Text	Table	Info	OR	HQ	OD
HYBRIDQA [26]	✗	✓	✓	✗	✗	✓	✓
MULTIMODALQA [204]	✗	✓	✓	✗	✓	✗	✓
OTT-QA [24]	✗	✓	✓	✗	✓	✓	✓
MANYMODALQA [74]	✗	✓	✓	✗	✗	✓	✓
WIKIMOVIES [141]	✓	✓	✗	✗	✓	✗	✗
TAT-QA [259]	✗	✓	✓	✗	✗	✓	✗
FINQA [28]	✗	✓	✓	✗	✗	✓	✗
HETPQA [191]	✗	✓	✓	✓	✗	✓	✗
COMP MIX (ours)	✓	✓	✓	✓	✓	✓	✓

OR: Open Retrieval; **HQ:** Human Questions; **OD:** Open Domain.

Table 6.2: Basic statistics for the COMP MIX benchmark.

Domains	Books, Movies, Music, TV series, Soccer
Questions	9,410 (train: 4,966, dev: 1,680, test: 2,764)
Avg. question length	9.19 words (min=2, median=9, max=28)
Avg. no. of question entities	1.11 (min=1, median=1, max=4)
Avg. answer length (text)	2.17 words (min=1, median=2, max=21)
Avg. no. of answers	1.02 (min=1, median=1, max=6)
Entities covered	5,413 (long-tail: 2,511, with <50 KB-facts)

information from this “main” information source ($\simeq 50\%$ of Freebase removed for evaluating on WebQuestions in [196]), to show that the usage of heterogeneous sources is helpful.

All existing benchmarks suffer from one or more of the following issues:

- (i) their questions are not fully human-generated, and hence lack diversity in formulations by real users [141, 204, 255];
- (ii) they span only two sources, like tables and text [24, 28, 259], or text and KBs [141, 255];
- (iii) they are restricted to small or artificial KBs, orders of magnitude smaller than large curated KBs like Wikidata [141, 255], not reflecting real-world knowledge;
- (iv) they explore only a specific domain like finance [28, 259], geography [120], or e-commerce [191];
- (v) their questions are only in conversational form with implicit intent, unsuitable for evaluating stand-alone QA methods [33, 41, 145] (including our CONVMIX benchmark from Section 3.3).

6.3.2 Construction of the COMP MIX Benchmark

Design principles. To overcome these limitations, we devise a new benchmark for heterogeneous QA, with the following desiderata and resulting design principles:

- (i) questions should be created by humans;
- (ii) questions should naturally require heterogeneous sources;
- (iii) sources should not be artificially constructed or restricted.
- (iv) the benchmark should not be domain-specific, but cover a wide spectrum of topics; and
- (v) questions should be self-contained and complete.

Table 6.3: Answer coverage on the COMPMIX benchmark across information sources.

Source(s)↓	Answer presence (human annotation)	Answer presence (automated measurement)
KB	0.308	0.807
Text	0.280	0.690
Tables	0.112	0.272
Infoboxes	0.299	0.545
KB + Text	0.588	0.853
KB + Tables	0.420	0.821
KB + Infoboxes	0.607	0.831
Text + Tables	0.393	0.702
Text + Infoboxes	0.580	0.734
Tables + Infoboxes	0.412	0.610
KB + Text + Tables	0.701	0.857
KB + Text + Infoboxes	0.888	0.861
KB + Tables + Infoboxes	0.720	0.841
Text + Tables + Infoboxes	0.692	0.743
All sources	1.000	0.865

Metadata. COMPMIX includes questions, their domains, and their corresponding answers. Answers are Wikidata identifiers (text labels are also provided), plain text strings, or normalized dates. This enables consistent evaluation across extractive and generative answering models. Entity markup in questions is provided by crowdworkers. Answer sources used by the annotators are also available: “KB”, “text”, “table”, or “infobox”.

6.3.3 Characteristics of the COMPMIX Benchmark

Answer coverage. One key desideratum of the benchmark is that heterogeneous sources are actually required for answering the questions inside. We thus analyzed the answer coverage of each information source, which is the fraction of questions that a source contains the answer for. Ideally, each source should have an answer coverage far less than 100%.

At the time of benchmark creation, Turkers provided the source they consulted for locating their answer. For computing coverage, we first consider these *source annotations by the crowdworkers*. However, this measurement only captures whether a specific information source has the desired information, without any implications concerning the other sources.

Therefore, we also conducted an automatic analysis of the answer coverage using a recall-oriented retriever that, given a question, tries to obtain as many relevant pieces of evidence as possible from all our sources. This retriever is implemented exactly as the retriever used for QUASAR (presented in Section 4.2.2), and uses CLOCQ with $k=10$ for enhanced answer recall. We measure this *automated answer coverage* as the fraction of questions for which the gold answer is present in the pool of retrieved evidence. As with any large-scale automated analysis, this statistic is a noisy proxy, because the mere presence of an answer does not necessarily mean that the surrounding evidence is question-relevant.

The results of both analyses are in Table 6.3. First, we see that the AMT annotators used the KB, text and infoboxes almost equally often to answer their questions, and tables are also consulted about 10% of the times. This proves that COMPMIX is *not biased* towards any specific underlying source. Second, from an automated measurement, we learn that adding an infor-

Table 6.4: Answer redundancy on the COMPMIX benchmark across our information sources.

Answer redundancy	Fraction of questions
Answer found in 1 source	0.157
Answer found in 2 sources	0.168
Answer found in 3 sources	0.341
Answer found in all sources	0.199

mation source always improves the answer coverage. Note that this is a natural expansion, as opposed to augmentation by some sources after artificial suppression of large parts of other sources, which is a typical approach in existing work on heterogeneous QA [154, 180, 196, 197]. Also, our recall-oriented retriever only provides a loose upper bound ($\simeq 87\%$): the performance of an actual retriever that balances recall and precision would currently reach a lower number. Thus, our benchmark leaves substantial room for the development of smart *heterogeneous retrievers*. Overall, these measurements suggest that all four sources are naturally required for answering the questions in COMPMIX, and different sources complement each other nicely.

Answer redundancy. Answer redundancy creates scope to test a heterogeneous system’s ability to boost confidence in its prediction when multiple matches happen across sources. For each question, we thus measured the number of sources touched by the retrieved pieces of evidence that actually contain the gold answer. Results are provided in Table 6.4. What we can see from here is that for a substantial proportion of questions, the answer is located in two ($\simeq 17\%$) or three ($\simeq 34\%$), out of four, sources. A reasonable chunk even has redundancy across all sources ($\simeq 20\%$). This shows that COMPMIX has ample answer redundancy to be exploited.

Anecdotal examples. Table 6.5 shows representative examples from the COMPMIX benchmark. These examples illustrate that our dataset has a wide range of questions in terms of both *syntactic structure* – from well-formulated fluent questions (1, 4, 5, 9) to ad-hoc telegraphic queries (6, 7), as well as *semantic complexity* – from simple intents (6, 8) to more complex ones requiring conjunction (2), temporal understanding (3, 5), or aggregation (9).

6.4 Experimental Setup

6.4.1 Benchmarks

We run experiments on three benchmarks with different characteristics.

- **COMPMIX.** Our COMPMIX benchmark (Section 6.3) which is specifically designed for evaluating QA systems tapping into heterogeneous sources. The dataset has 9,410 questions, out of which 2,764 are used for testing. Answers are crisp entity names, dates, or other literals.
- **CRAG.** We further evaluate on a subset of complex questions in the CRAG [239] dataset, which was recently released as a testbed for RAG-based QA systems. We utilize the same pipeline and sources as outlined in Section 6.2, without using the web snippets or APIs provided with CRAG. This way we focus on entity-centric questions that do not require access to live web data (e.g., news feeds), and disregard cases where the results would be up-to-date quantities. This restricts the test data to 436 entity-centric questions.
- **TIMEQUESTIONS.** To showcase the generalizability of our pipeline, we conduct experiments on TIMEQUESTIONS [93], a benchmark for temporal QA. The dataset requires temporal understanding and reasoning, which are well-known limitations of GLMs [43]. TIMEQUESTIONS has 16,181 questions (3,237 for testing).

Table 6.5: Representative questions from the COMPMIX benchmark. Sources that can be used for answering these questions are in brackets.

Books	Movies	Music	TV series	Soccer
1. <i>What did Rayford Steele from Left Behind do as a job?</i> Pilot [KB, Text]	2. <i>Which lead actress appeared in both Terms of Endearment and The Evening Star?</i> Shirley MacLaine [KB]	3. <i>Who replaced Ozzy Osbourne in Black Sabbath the first time?</i> Ronnie James Dio [Text, Info]	4. <i>What TV show featured the character called Carrie Mathison?</i> Homelande [KB, Text, Info]	5. <i>Where did the Uruguay national football team play their first recorded match?</i> Paso del Molino [Text]
6. <i>Author of the book To Kill a Mockingbird?</i> Harper Lee [KB, Text, Table, Info]	7. <i>Film in which Wallace Reid played the role of Walter Jarvis?</i> The Ghost Breaker [KB, Text]	8. <i>What is the singer Lemmy's birth name?</i> Ian Fraser Kilmister [KB, Text, Info]	9. <i>How many episodes of The 100 did Jason Rothenberg write?</i> 16 [KB, Text, Table]	10. <i>Who was runner up in the 1998 World Cup?</i> Brazil football team [KB, Text, Info]
11. <i>Name the fifth book in Malory Towers series.</i> In the Fifth at Malory Towers [KB, Table]	12. <i>Which movie is longer, Hamlet or Gone with the Wind?</i> Hamlet [KB, Info]	13. <i>What year was Inna's Hot album released in the US?</i> 2009 [Text]	14. <i>Which season of Teen Wolf did Tyler Posey become a co-producer?</i> 5 [Text, Info]	15. <i>Which soccer player scored the most number of goals in the UEFA Euro 2004 tournament?</i> Milan Baroš [KB, Text, Info, Table]
16. <i>What years were the two volumes of Little Women published?</i> 1868, 1869 [KB, Info]	17. <i>What is the run time of Titanic?</i> 195 minutes [KB, Infobox]	18. <i>What is the name of the second single in the album Arise?</i> Dead Embryonic Cells [Text, Table]	19. <i>What year was Matt Groening born?</i> 1954 [Text]	20. <i>Who was the kit manufacturer of Chelsea Football Club from 1981 to 1983?</i> Le Coq sportif [Text, Table]

Typical examples for the questions in these three benchmarks are:

COMPMIX: *Which player won the most number of MotM titles in the FIFA world cup of 2006?*

CRAG: *What was the worldwide box office sales for little hercules?*

TIMEQUESTIONS: *Which club did Cristiano Ronaldo play for before joining Real Madrid?*

6.4.2 Baselines

As reference points for QUASAR, we study the performance of the following methods:

- **Generative LMs.** We compare QUASAR against out-of-the-box GLMs: **INSTRUCTGPT** (“text-davinci-003”), **GPT4** (“gpt-4”) and **LLAMA**⁶. The same prompt is used for all GLMs, consistent with previous work [35, 252]:

Please answer the following question by providing the crisp answer entity, date, year, or numeric number. Q: <question>

- **Heterogeneous QA methods.** UNIK-QA [154], CONVINSE (Chapter 3), and EXPLAINNN (Chapter 4) are QA methods designed to integrate heterogeneous sources: text, tables and KB. All of these integrate the exact same sources as QUASAR.
- **STATE-OF-THE-ART.** For COMPMIX and TIMEQUESTIONS, we also compare against state-of-the-art methods from the literature: SPAGHETTI [252] and our unfaithful variant of FAITH (Chapter 5), UN-FAITH, which are among the best performing systems.

Results are taken from the literature whenever applicable. On CRAG, we use the models trained on COMPMIX for QUASAR and heterogeneous QA baselines.

⁶<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

6.4.3 Metrics

We measure *precision at 1* (**P@1**) as our main metric [175] on all benchmarks. On CRAG, we manually annotate answer correctness, as the ground-truth answer formats vary (e.g., entity name variants, lists, sentences). We also compute the number of neural parameters aggregated over all sub-modules (**#Parameters**). Parameter counts for GPT-models are taken from [143] (GPT4 might have less active parameters during inference). For further analysis we measure *answer presence at k* (**AP@k**), i.e. whether the answer is present in the top- k ranked evidence pieces, and *mean reciprocal rank* within the top- k evidence pieces (**MRR@k**).

6.4.4 Configurations

For the QU and ERS stages we adopt the code from our EXPLAINNN project. During evidence retrieval, we use CLOCQ, setting its specific parameters to $k=10$ and $p=1,000$. As default, we use the GNN technique for the evidence scoring step. For efficiency, we use lightweight models for initializing the GNN encoders – the same models used for the cross-encoder-based ES⁷. The GNNs are trained for 5 epochs with an epoch-wise evaluation strategy, i.e. we choose the model with the best performance on the respective dev set. We train the GNNs on graphs with a maximum of 100 evidence and 400 entity nodes (as scored by BM25). During inference, the first GNN is applied on graphs with 1,000 evidence and 4,000 entity nodes, shrinking the pool of evidence pieces to the top-100. The second GNN then runs on graphs with 100 evidence and 400 entity nodes. The factor of 4 entities per evidence node (on average) holds sufficient for the observed data, and enables batched inference. Default values are used for the other parameters.

Our implementation of the HA stage uses a LLAMA model with 8B parameters (the same as when used as a baseline)⁸, fine-tuned for 2 epochs with a warm-up ratio of 0.01 and a batch size of 8, again with an epoch-wise evaluation strategy. Other parameters are set to the default Hugging Face training parameters.

We used a single GPU (NVIDIA Tesla H100NVL) for model training and local inference.

6.5 Experimental Results and Insights

6.5.1 Key Findings

The main results of our experiments are shown in Table 6.6.

QUASAR is competitive on all benchmarks. First of all, we note that QUASAR achieves competitive performance across all three benchmarks.

On COMPMIX, baselines for heterogeneous QA and LLAMA perform similarly, whereas GPT-based GLMs can answer more than 50% of the questions correctly. QUASAR exhibits substantially higher performance, on par with the state-of-the-art method SPAGHETTI [252] (which is based on GPT4, and thus computationally much more expensive in terms of actual compute used).

On the CRAG dataset, P@1 drops for all methods except for GPT4. The benchmark includes questions that can be ambiguous/confusing (“*who was the director for the report?*”), on “exotic” entities with answers in social media (“*how many members does the teknoist have?*”), or require up-to-date information (“*when did chris brown release a song or album the last time?*”), and other cases that are challenging for all methods.

⁷<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-4-v2> and

<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2>

⁸<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

Table 6.6: End-to-end P@1 of QUASAR and baselines on three benchmarks. Results for INSTRUCTGPT and GPT4 are taken from the literature [35, 91]. INSTRUCTGPT is not accessible anymore, hence no results on CRAG.

Benchmark → Method ↓	COMP MIX	CRAG	TIMEQUESTIONS	#Parameters
INSTRUCTGPT	0.502	–	0.224	175,000 M
GPT4	0.528	0.633	0.306	1,760,000 M
LLAMA [209]	0.431	0.385	0.178	8,030 M
CONVINSE [33]	0.407	0.298	0.423	362 M
UNI-K-QA [154]	0.440	0.280	0.424	223 M
EXPLAIGNN [34]	0.442	0.303	0.525	328 M
STATE-OF-THE-ART	0.565 (SPAGHETTI [252])	–	0.571 (UN-FAITH [91])	–
QUASAR (ours)	0.564	0.362	0.754	8,218 M

Table 6.7: Answer presence and answering precision of QUASAR with different combinations of input sources (on the respective test sets).

Benchmark →	COMP MIX			TIMEQUESTIONS		
Input sources ↓ / Metric →	P@1	AP@100	AP@30	P@1	AP@100	AP@30
Text	0.455	0.563	0.531	0.539	0.515	0.487
KB	0.481	0.677	0.637	0.724	0.701	0.674
Tables	0.432	0.501	0.482	0.536	0.347	0.328
Text + KB	0.537	0.749	0.706	0.745	0.776	0.748
Text + Tables	0.503	0.632	0.594	0.567	0.578	0.549
KB + Tables	0.524	0.728	0.692	0.743	0.731	0.703
Text + KB + Tables	0.564	0.759	0.724	0.754	0.776	0.749

Finally, QUASAR establishes new state-of-the-art performance on the TIMEQUESTIONS benchmark. Consistent with our findings in Chapter 5, all of the tested GLMs show greatly reduced performance on such temporal questions, which inherently require temporal understanding and reasoning – a known weakness of stand-alone GLMs.

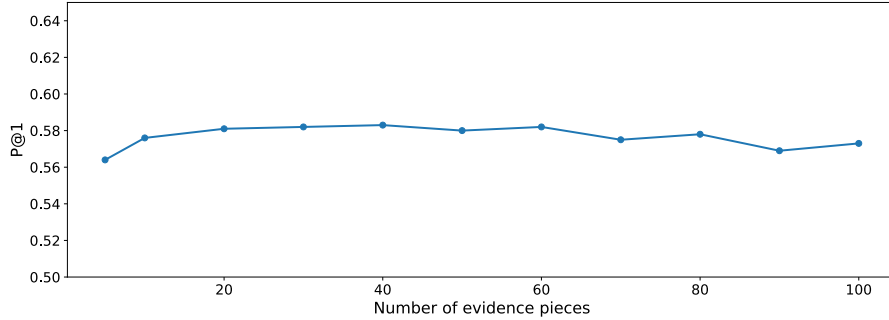
Integration of heterogeneous sources is vital. QUASAR integrates evidence from text, KB and tables into a unified framework. We aim to better understand how this affects the answering performance of the method. Table 6.7 shows the end-to-end performance of QUASAR with different combinations of the input sources. The results clearly indicate that all types of sources contribute, with the option “Text + KB + Tables” performing best, with a large margin over tapping only into single source types. Note that “Tables” also includes infoboxes, which we group here for simplicity (conceptually, infoboxes are a simple form of a table).

6.5.2 Analysis

Unified retrieval enhances performance. In the ES step, we re-rank and filter evidence from different source types, and feed the unified top- k^* into the HA stage. We conduct a comparison in which we consider the top-10 evidence pieces from each source type individually. This gives equal influence to KB, text and tables, whereas our default is based on global ranking. Table 6.8

Table 6.8: Answer presence and precision of QUASAR with different evidence scoring choices for deriving the top-30 evidence pieces (on COMPMIX dev set).

Input evidence pieces ↓ / Metric →	P@1	AP@30
Top-30 Text + KB + Tables (ours)	0.574	0.710
Top-10 Text + Top-10 KB + Top-10 Tables	0.560	0.709

**Figure 6.3:** Answering performance of QUASAR (measured by P@1) on the COMPMIX dev set with different numbers of evidence pieces.

shows the results for this analysis, showing our default choice performs better. The reason is that different questions require different amounts of evidence from each of the source types.

QUASAR works well with small amounts of evidence. We investigate the influence of the number of evidence pieces fed into the HA stage, varying it from 5 to 100. Results are shown in Figure 6.3. As the curve shows, there is a sharp increase in precision as we add evidence up to 30 or 40 pieces, which is around our default of top-30. This indicates that a certain amount of evidence is needed, to overcome the inherent noise and arrive at sufficient answer presence. As we increase the amount of evidence further, we observe a saturation effect, and eventually a degradation of performance. Too much evidence not only has diminishing returns, but can actually be confusing for the HA stage. This reconfirms our heuristic choice of top-30: enough for good answering while keeping computational costs reasonably low.

Ablation study on re-ranking. For more insight on the possible configurations of the ES stage, we conducted an ablation study with different options, including relying solely on the initial BM25 scoring without explicit re-ranking. The results are shown in Table 6.9. We observe that the iterative reduction in two steps is slightly better than the single-step variants (going down from top-1000 to top-30 in one ES step). Between the two options of using a GNN or a CE, in this case, the differences are relatively small, except for the MRR@100 metric. One reason could be the pre-trained CE used as encoder (we used a general-purpose ELM in Chapter 4).

A notable effect is that our ES techniques retain the answer presence at a very high level, only a bit lower than for the initial top-1000. The last two rows of Table 6.9 demonstrate that our ES is crucial: without explicit re-ranking, the technique of just picking smaller top- k from the original BM25 model leads to substantial degradation in both answer presence and precision.

Quality of the SR. To assess the quality and robustness of the structured representation, we inspected a sample of questions and their SRs. Table 6.10 gives three anecdotal examples. We show SRs generated by QUASAR, which makes use of the distant supervision approach presented in Section 3.2.5. As the training data is obtained via different heuristics, this can be a limiting

Table 6.9: Ablation study for different ES strategies of QUASAR on the COMPMIX dev set. The answer presence in the ES input with top-1000 evidence pieces is 0.760.

ES Method ↓ / Metric →	COMPMIX (dev set)			
	P@1	AP@100	AP@30	MRR@100
GNN: 1000 → 100 → 30	0.574	0.738	0.710	0.572
CE: 1000 → 100 → 30	0.573	0.740	0.721	0.553
GNN: 1000 → 30	0.567	<i>n/a</i>	0.710	0.567
CE: 1000 → 30	0.570	<i>n/a</i>	0.715	0.558
BM25: 100 (w/o GNN or CE)	0.490	0.652	<i>n/a</i>	0.259
BM25: 30 (w/o GNN or CE)	0.468	<i>n/a</i>	0.534	0.259

Table 6.10: Anecdotal examples for pairs of questions and generated SRs.

Question	Current SR by QUASAR	SR via ICL
<i>what was disneys first color movie?</i>	<p>Question entities: <i>disneys</i></p> <p>Question predicate: <i>was first color movie</i></p> <p>Expected answer type: <i>animated feature film</i></p>	<p>Question entities: <i>Disney</i></p> <p>Question predicate: <i>first color movie</i></p> <p>Expected answer type: <i>film, animated film</i></p> <p>Time: <i>first</i></p>
<i>at the oscars, who won best actor in 2018?</i>	<p>Question entities: <i>at the oscars</i></p> <p>Question predicate: <i>who won best actor in 2018</i></p> <p>Expected answer type: <i>human</i></p>	<p>Question entities: <i>Oscars, 2018</i></p> <p>Question predicate: <i>won best actor</i></p> <p>Expected answer type: <i>person, actor</i></p> <p>Time: <i>2018</i></p>
<i>which German state had the most Corona-related death cases in the first year after the outbreak?</i>	<p>Question entities: <i>Germany, Corona</i></p> <p>Question predicate: <i>which state had the most related death cases in the first year after the outbreak</i></p> <p>Expected answer type: <i>state</i></p>	<p>Question entities: <i>Germany, Corona-related deaths</i></p> <p>Question predicate: <i>highest count of death cases</i></p> <p>Expected answer type: <i>location, state</i></p> <p>Location: <i>Germany</i></p> <p>Time: <i>first year after Corona outbreak</i></p>

factor when user intents become more complex.

Therefore, we also looked at SRs derived via in-context learning using GPT4 with five hand-crafted examples. As shown in our earlier work on temporal QA [91], such data can be used for training smaller models (e.g., BART), which can greatly boost the completeness and overall quality of the generated SRs.

From the sampled set, we observed that the ICL-based SRs are more diverse with additional question-specific slots, whereas the BART-based SRs, by design, include only the general slots for entities, predicates and answer types. However, both approaches achieve very high quality in filling the slots, capturing the user’s information need very well.

Interestingly, when questions get complicated, with nested phrases, the ICL-based variant succeeds in decomposing the questions, learning from only five ICL examples. For example, for the question “*which German state had the most Corona-related death cases in the first year after the outbreak?*” the “Time” slot becomes “*first year after Corona outbreak*”, which can be resolved to identify the temporal scope (as in Chapter 5). In general, we believe that such question decomposition, beyond simple temporal constraints, is an interesting theme for complex QA.

Table 6.11: Performance on the COMPMIX dataset, with the option to refrain from answering.

	COMPMIX			
Metric → Method ↓	P@1	P@1 (answered)	Refrain rate	Refrain accuracy
LLAMA	0.431	0.471	0.089	<i>n/a</i>
QUASAR (selective)	0.497	0.713	0.303	0.838

Table 6.12: Performance on TIMEQUESTIONS, with the option to refrain from answering.

	TIMEQUESTIONS			
Metric → Method ↓	P@1	P@1 (answered)	Refrain rate	Refrain accuracy
LLAMA	0.177	0.276	0.392	<i>n/a</i>
QUASAR (selective)	0.597	0.804	0.257	0.864

Refraining from answering. We can train our model to refrain from answering in scenarios where the provided evidence does not contain an answer to the question. Specifically, during training, when the answer is not present in the evidence, we change the target answer to *unknown* (instead of removing the training instance). This variant is referred to as QUASAR (*selective*).

We measure the ratio of questions for which *unknown* is provided as answer, and the P@1 restricted to questions that are answered. The accuracy of refraining from answering is measured as well, based on whether the answer is present in the evidence or not. We conduct this experiment on COMPMIX and TIMEQUESTIONS, for which we can compute answer presence exactly. We also compute results for LLAMA, which is already instructed with the option to answer “*don’t know*”. Table 6.11 and Table 6.12 show the results. For COMPMIX, we observe that QUASAR has high accuracy on refraining when appropriate, whereas LLAMA tends to be overconfident with a very small rate of *unknowns*, leading to incorrect answers. Among the questions that QUASAR answered (without *unknown*), we observe a substantial improvement in the P@1 metric, answering 71% (COMPMIX) and 80% (TIMEQUESTIONS) of questions correctly.

6.6 Related Work

The RAG paradigm came up as a principled way of enhancing GLM factuality including provenance and mitigating the risk of hallucination [72, 116]. It is highly related to the earlier retriever-reader architectures for QA [23, 105], especially when the reader uses the fusion-in-decoder method [85, 154]. Since its invention, there has been substantial work on improving RAG methods, introducing a wide suite of extensions, such as batched inputs, interleaving retrieval and generation steps, and more (see the recent surveys [62, 257]).

Most relevant for this work is the research on jointly leveraging all different sources: text, KBs, and tables (including CSV and JSON files). This includes the UNIK-QA system [154], the SPAGHETTI [252] and SUQL [125] projects, the MATTER method [114], and our own prior work [33, 34]. Out of these, we include UNIK-QA, SPAGHETTI and our own systems CONVINSE and EXPLAINNN as baselines in the experimental evaluation. Their architectures are similar to ours, but UNIK-QA and SPAGHETTI do not have our distinctive elements of question understanding and iterative re-ranking, originally introduced in CONVINSE (Chapter 3) and EXPLAINNN (Chapter 4).

6.7 Discussion

Benchmark performance. Our method, QUASAR with an 8B LLAMA model, outperforms much larger GLMs like GPT4 on two of the three benchmarks, with a substantial margin for temporal questions. Obviously, pre-trained GLMs have only limited sense of properly positioning “remembered” facts on the timeline even with training data that exceeds ours by several orders of magnitude. This confirms our intuition that GLMs alone are not good at “recalling” higher-arity relations that require combining distant pieces of evidence. This is a sweet spot for RAG. Only for the CRAG benchmark, QUASAR is substantially inferior to a full-blown GLM. This is likely due to the nature of the questions: not necessarily the complexity of the information needs, but the need for more web sources (beyond what our experiments tap into).

Cost/Performance ratio. The most important takeaway from our experiments is that QUASAR achieves its competitive performance at a much lower cost than the full GLMs. Assuming that the consumed GFlops (=giga floating point operations per second) are proportional to the number of model parameters, QUASAR achieves a cost reduction by a factor of 200x compared to INSTRUCTGPT and 2000x compared to GPT4. This does not only mean less computation, but also a massively lower electricity bill and climate impact.

Role of re-ranking. As our ablation studies show, merely using top- k evidence from an initial BM25-style ranking does not provide good performance. Also, there seems to be a sweet spot in the choice of k : we need enough evidence for connecting the dots if the question requires multiple pieces of information, or for corroborating candidates if the question finds many useful but noisy pieces. In the experiments, $k=30$ turns out to be good choice; much lower k results in insufficient evidence, and much larger k leads to saturation and ultimately degrading performance. Our argument for iteratively shrinking the candidate set in multiple rounds of re-ranking is substantiated in our experiments. More research is needed to better understand the role of evidence ranking for RAG.

Limitations of evidence retrieval. Our evidence retrieval generally yields high answer presence, in the order of 75% in the top-100 or even top-30. An important case where our ER is still insufficient are questions that require aggregating information over a large number of evidence pieces. An example is asking for the life-time total of 3-point scores of the basketball player Dirk Nowitzki. This requires collecting a set of per-season tables with NBA player statistics, but also other web sources with numbers for his career before he joined the NBA (including his youth teams). Importantly, such questions do not allow for general top- k cut-offs, but require incorporating *all* relevant evidence, for deriving the correct answer. Of course, there are sometimes shortcuts like a Wikipedia article or biography mentioning the total number explicitly, but this cannot be universally assumed. The bottom line is that evidence retrieval should be reconsidered as well, striving to improve the recall dimension.

Limitations of answer generation. For the answer generation, we utilize a GLM for enhanced performance, using it as an extractor (“reader”) from the given evidence. Despite the wide belief that GLMs can perform deep reasoning over many pieces of evidence, our observation is that the extraction works only well – robustly and faithfully – for relatively simple questions with joins or simple aggregations over a few snippets. However, complicated questions such as asking for the top-100 NBA players with the largest number of life-time 3-point scores (again including their pre-NBA careers) are currently out of scope, even when assuming perfect evidence retrieval. This offers many opportunities for pushing the envelope further.

7

Personal Question Answering

The previous chapters of this dissertation target general world knowledge covered in publicly available sources. In this chapter, we aim to answer questions over non-public data, exemplified by personal information. User devices log vast amounts of data every day, such as calendar entries, workout statistics, shopping records, streaming history, and more. Such data naturally appears in heterogeneous form – streaming data or workouts are captured in structured form, while mails, social media posts or notes consist of unstructured text. Information needs range from simple look-ups to queries of analytical nature, involving more complex operations than targeted in previous chapters. We present REQAP, a novel method that creates an executable operator tree for a given question, via recursive decomposition of the question complexity. Operators are designed to enable seamless integration of structured and unstructured sources. The execution of the operator tree yields an answer that can be traced back to its sources. For training and experiments, we construct the PERQA dataset, with synthetic but realistic user data and questions.

All code, data and a demo are available at <https://reqap.mpi-inf.mpg.de>.

Contents

7.1	Introduction	105
7.2	Data Model	107
7.3	The REQAP Methodology	107
7.3.1	Question Understanding and Decomposition (QUD)	108
7.3.2	Operator Tree Execution (OTX)	109
7.3.3	Cache for Sub-Question Results	113
7.4	The PERQA Benchmark	113
7.4.1	Existing Benchmarks and their Limitations	113
7.4.2	Construction of the PERQA Benchmark	113
7.5	Experimental Setup	116
7.5.1	Configurations	116
7.5.2	Baselines	117
7.5.3	Metrics	118
7.6	Experimental Results and Insights	119
7.6.1	Key Findings	119
7.6.2	Analysis	119
7.6.3	Experiment with TIMELINEQA	123

7 Personal Question Answering

7.7	User Study for Evaluation	123
7.7.1	Setup	124
7.7.2	Results	124
7.8	Demonstration	125
7.8.1	Personal User Data	125
7.8.2	REQAP Inference	126
7.8.3	Implementation	128
7.8.4	Usability	128
7.9	Related Work	128
7.10	Conclusion	129

7.1 Introduction

Motivation. As discussed and demonstrated in previous chapters of this dissertation, QA over combinations of structured and textual data has been substantially advanced by verbalizing all contents and encoding it with a language model [33, 154, 207, 245].

A prominent case of such heterogeneous content, of growing importance, is *personal information* [73]. User devices produce and collect vast amounts of personal data on a daily basis. Each interaction with digital services, such as Amazon Marketplace, Netflix or Spotify, is stored for the specific user, and services running on smartphones or smart watches track data without explicit user actions, such as travel routes or workouts. Users want convenient tools to query their data, as easy as with a GLM-powered AI co-pilot. At the same time, *data protection* is a crucial requirement, as full control over sensitive content is a top priority for users [102, 136]. Ideally, all data and its processing resides locally on the owner’s devices (mobile, tablet or PC), tightly limiting the available computational resources. Reconciling these goals is a challenge [73].

Users’ needs range from look-ups of events to queries of analytical nature [206]:

q^1 : Which books did I buy for my friends last Christmas?

q^2 : When did I last have pizza with both my parents?

q^3 : How often did I eat Italian food after playing football?

q^4 : Daily kilometers I ran during business trips?

Such questions require fine-grained understanding of the user intent, and entail complex operations like *grouping*, *aggregations*, *joins* or matching *temporal conditions*, often with hundreds of relevant information pieces from different sources.

The goal of this work is to devise a QA system for personal data that

- (i) supports such complex information needs;
- (ii) is user-friendly and intuitive for lay people; and
- (iii) has low footprint regarding computation, memory and energy, suitable for running directly on user devices.

Limitations of the State of the Art. State-of-the-art methods for QA over heterogeneous data would mostly follow the *verbalization* theme. Relevant data is retrieved, linearized into textual form and cast into token sequences [33, 154]. These are fed into a GLM [143] for generating answers, in the spirit of *retrieval-augmented generation* (RAG) [4, 124, 207]. The combination of verbalization and RAG has been successfully applied for heterogeneous QA when few evidence pieces are sufficient for answering (cf. Chapter 6). However, this paradigm falls short as relevant user data often exceeds the context window of GLMs, going well beyond 100K tokens (e.g., for q^3 and q^4). Even if their context size was unlimited, GLMs struggle with more complex operations over hundreds of events, such as aggregation, grouping or temporal reasoning. This limitation is amplified for numerical data, which is encoded as simple tokens, hindering precise aggregation.

The predominant theme for such complex questions, with large-scale aggregation and joins, is *translation*. A GLM is provided with the natural-language question and the data schema (i.e., headers of tables), in order to generate code in the form of a logical query, typically for languages like SQL or SPARQL (e.g., [53, 118, 127]), which is then directly executed. This *code generation* (CODEGEN) is often referred to as NL2SQL or Text2SQL [118, 127].

Translation works well when all data appears in structured form, with a well-designed data schema, such that a single query can access all relevant information jointly. In our setting, this is not the case, as exemplified in Figure 7.1. Relevant user data naturally appears in text, which is not easily accessible by queries. Processing such unstructured text is a key requirement in the context of QA over personal data (e.g., for q^2 , q^3 , and q^4).



Figure 7.1: Excerpt of relevant user data for the question q^3 , stored in *structured* (workouts), *semi-structured* (calendar) and *unstructured* (social media, mails) sources.

Consider question q^3 (“How often did I eat Italian food after playing football?”) as our running example, for which Figure 7.1 shows a small excerpt of relevant user data. Relevant information, such as playing football, could be stored in different form, including *structured sources* like workout tables, *semi-structured sources* such as calendar entries, or *unstructured sources* such as social media posts. Moreover, the very same event (e.g., eating Pizza) is often captured by multiple sources (e.g., calendar entry on 11 October and social media post on 12 October), which could lead to double-counting. Finally, cues are often subtle (e.g., the football icon in the second social media post), which means that simple matching of surface text is not sufficient.

Approach. To overcome these limitations, the key idea of this work is to bridge the gap between verbalization and translation, and combine the strengths of both paradigms. Our approach creates an *operator tree*, similar to a query in the translation paradigm, for which we propose judiciously designed operators that enable structured access to unstructured text, based on ideas of the verbalization paradigm.

Specifically, we present a novel methodology for the task of QA over personal data, named REQAP (Recursive Question Understanding for Complex Question Answering over Personal Data). All personal data sources are treated as lists of events. Each event is described by a set of key-value pairs (e.g., capturing a single mail or workout). For a given user question, REQAP derives a database-style operator tree. Notably, in order to generate these operator trees, we propose a new mechanism for recursively *decomposing* the complexity of questions, via in-context learning with GLMs to enable fine-tuning of small-scale LMs. This enables deployment of the entire QA system on end user devices (e.g., notebooks or mobile phones). Apart from standard SQL-like operators, we introduce two novel operators, RETRIEVE and EXTRACT, that enable efficient retrieval from both structured and unstructured data sources, and enhancing unstructured text with question-related key-value pairs, via on-the-fly information extraction. The generated operator trees are executed for deriving the traceable answer.

For evaluation, we construct a new benchmark, PERQA, which synthesizes realistic user data and questions based on personas, and has 3,500 complex questions and more than 40,000 events per persona. Experiments on PERQA demonstrate substantial improvements of REQAP compared to baselines based on the verbalization or translation paradigms.

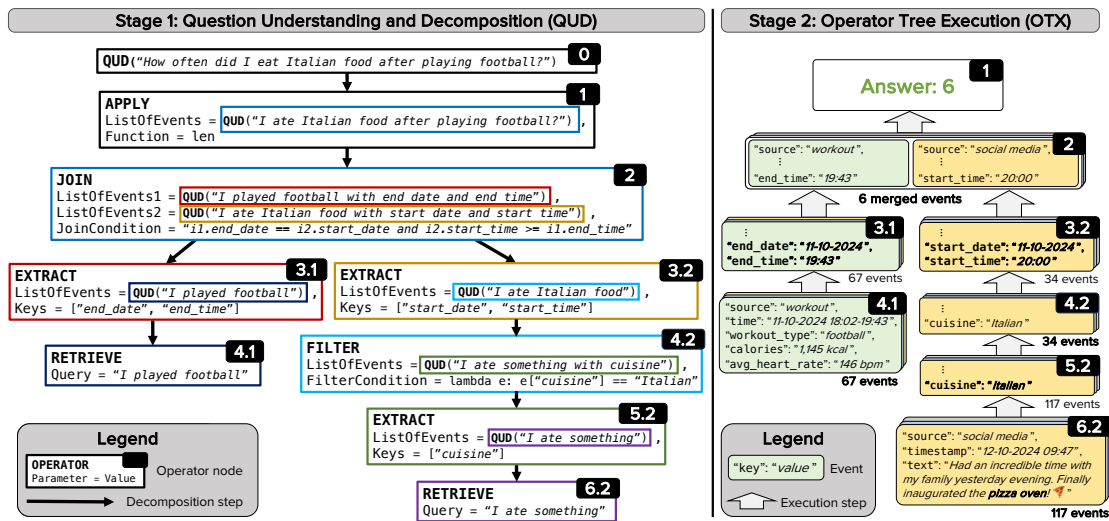


Figure 7.2: Example inference of the REQAP methodology. The left side illustrates the output of the QUD stage, an operator tree, for the example question q^3 . The right side shows the execution of the operator tree within the OTX stage to derive the answer, show-cased by relevant user events and their processing.

Contributions. The salient contributions of this work are the following:

- Proposing a novel approach, REQAP, for QA over heterogeneous personal data, based on a recursive question understanding mechanism.
- Developing a complete QA system that can run on end user devices, which is trained via GLMs and their in-context learning capabilities.
- Constructing a new large-scale benchmark, PERQA, with realistic user data and questions that can facilitate future work on QA over personal data.

7.2 Data Model

We treat all data sources as temporally ordered lists of *events*, where each event is a dictionary with a set of *key-value pairs*. For example, each mail is an event with keys *date*, *time*, *sender*, *recipients*, *subject* and *body*; for music streams, each event would have keys like *date*, *start_time*, *end_time*, *title*, *artists*. Example events are shown in the bottom right of Figure 7.2 (workout in light green and social media in yellow).

Our current implementation covers the following event types, accessed from data exports: *calendar entries*, *notes*, *mails* and *social media posts* with semi-structured and unstructured data, and *workouts*, streams (*movies*, *TV series*, *music*) and *online purchases* with structured data. Note that event types can be added easily with this agile data model, which treats all contents as key-value pairs.

7.3 The REQAP Methodology

REQAP processes questions in two stages:

- (i) the *question understanding and decomposition* (QUD) for constructing the operator tree, and
- (ii) the *operator tree execution* (OTX) for deriving the answer with the corresponding events.

Figure 7.2 shows a characteristic example that illustrates the two stages of REQAP, and Table 7.1 lists the operators that are currently implemented.

7.3.1 Question Understanding and Decomposition (QUD)

Functionality. The task of QUD is to map a natural-language question into an operator tree, that yields the correct answer upon execution.

Challenges. The QUD stage requires fine-grained understanding of the users’ information needs. These questions are often complex, involving multiple operators for grouping, joining, or aggregation, even for a single question. Most critically, there is no training data available, and human annotations are non-trivial to create for crowdworkers.

Inference via ICL. Due to the absence of appropriate training data, we leverage *in-context learning* [14] to generate our operator trees. We handcraft operator trees for a diverse set of 40 questions, and then *dynamically* retrieve 8 ICL examples at each step via BM25.

In principle, we could prompt the GLM to generate a complete operator tree in a single shot, following the code generation line of works [118]. However, this approach is often too complex, even for a GLM [51], with many output trees being incomplete or wrong. Instead, we devise a novel method for *recursive decomposition*, where we call the GLM repeatedly to generate *partial operator trees* that have the freedom to resolve sub-questions of the user’s input in *recursive* QUD calls. These sub-questions are then the input to subsequent GLM calls, in order to refine the partial operator tree. The recursion stops when all leaf nodes are properly mapped to operators without any open QUD calls.

Example: Consider the operator tree in Figure 7.2.

- (i) The first GLM call generates node 1, with an *aggregate function* (list length, i.e., counting) applied to the question results.
- (ii) This is refined into node 2, with the *join* over two *sub-questions*: “I played football” and “I ate Italian food”.
- (iii) Each of two sub-questions is translated into an **EXTRACT** operator (operators 3.1 and 3.2 in the figure), and the left branch invokes the leaf-node **RETRIEVE** operator 4.1.
- (iv) The right branch 3.2 is further refined, leading to a cascade of **FILTER** (4.2), **EXTRACT** (5.2) and **RETRIEVE** (6.2) operators.
- (v) This operator tree is then executed bottom-up, as visualized on the right side of Figure 7.2.

To facilitate this method, our ICL examples for the GLM follow the same approach, showing how a complex question can be split into an operator part and a simpler sub-question. The ICL samples are judiciously crafted to bring out key difficulties that the model has to deal with, while covering all OTX operators. Table 7.16 shows five such ICL examples. Each example consists of multiple user and assistant turns, enabling the decomposition of questions. Figure 7.3 shows the instruction used for this ICL-variant of the QUD component.

In general, during GLM-inference for a given sub-question, the previous input sub-questions and output operators are retained for context. Thus, each few-shot sample is a sequence of decomposition steps.

Training. Recall that we aim to deploy REQAP on the users’ personal devices directly, including mobiles. This would be impossible with the large models ($\geq 70\text{B}$ parameters) used above for ICL.

The solution is to harness the (question, operator tree) pairs obtained via ICL to *distill* [81] the GLM into a smaller model (e.g., LLAMA with 1B parameters). To ensure high-quality training

```

You are given a question and need to generate a plan to infer the answer.
This plan follows Python syntax. You can make use of the following functions:
'QUD (question: str)'
    Used to make a recursive call to the QUD module.
'RETRIEVE (query: str) -> QExecutionResult'
    Retrieves all events that match the query.
'EXTRACT(l: QExecutionResult, attr_names: List[str], attr_types: List) -> QExecutionResult'
    Extracts the values for the provided attributes from each event.
'GROUP_BY(l: QExecutionResult, attr_names: List[str]) -> QExecutionResult'
    Function to group the results by the provided attributes.
'JOIN(l1: QExecutionResult, l2: QExecutionResult, condition: str) -> QExecutionResult'
    Function to join two results based on the condition. Condition is a string, and makes use of l1 and l2
    to refer to items in the list.
'APPLY(l: QExecutionResult, fct: Callable) -> QExecutionResult'
    Apply a callable on the list of events in l.
'MAP(l: QExecutionResult, fct: Callable, res_name: Optional[str]="map_result") -> QExecutionResult'
    Map a callable to each event or group (GROUP_BY result) in l, and store each result in the key 'res_name'.
'FILTER(l: QExecutionResult, filter: Callable) -> QExecutionResult'
    Filter the events using the provided callable.
'UNNEST(l: QExecutionResult, nested_attr_name: str, unnested_attr_name: str) -> QExecutionResult'
    Derives a single event for each value in the list in key 'nested_attr_name', with the value being stored
    in 'unnested_attr_name'.
'ARGMIN / ARGMAX (l: QExecutionResult, arg_attr_name: str, val_attr_name: Optional[str]) -> QExecutionResult'
    Computes the argmin/argmax based on the provided attribute ('arg_attr_name'), and returns the
    corresponding value ('val_attr_name').
'SUM / AVG / MAX / MIN (l: QExecutionResult, attr_name: str) -> QExecutionResult'
    Computes the sum/avg/max/min based on the provided attribute ('attr_name').

'QExecutionResult' is a special class that is used to store the results of functions.

Whenever accessing an attribute in a function call, make sure that this attribute is derived in a subsequent
call via an appropriate call (e.g., EXTRACT or MAP). For example, when val_attr_name="workout_type" is used
in a ARGMAX call, there has to be an EXTRACT call with attr_names=["workout_type",...] in the following.

Please provide only the next step, including the recursive QUD call to take (if needed).
Learn from the examples below.

```

Figure 7.3: Our GLM instructions for generating operator trees. This is followed by 8 in-context learning examples, which are dynamically chosen per step. Sample ICL examples are shown in Table 7.16.

Table 7.1: The REQAP operators with their input types.

RETRIEVE (Query, Optional: ListOfEvents)
EXTRACT (ListOfEvents, Keys)
JOIN (ListOfEvents1, ListOfEvents2, JoinCondition)
GROUP_BY (ListOfEvents, ListOfKeys)
FILTER (ListOfEvents, FilterCondition)
MAP (ListOfEvents, Function)
APPLY (ListOfEvents, Function)
UNNEST (ListOfEvents, KeyToUnnest)
Other operators: ARGMIN, ARGMAX, MIN, MAX, SUM, AVG

instances, we sample multiple operator trees for each training question, evaluate them on the data, and keep only those that return correct results.

Inference via SFT model. These *supervised fine-tuned* (SFT) models are deployed for generating operator trees from user questions, and follow the same decomposition procedure as with ICL. In our experiments, we report on different model sizes for the QUD stage.

7.3.2 Operator Tree Execution (OTX)

The OTX stage executes the operator tree obtained in the QUD stage to derive the answer. Table 7.1 provides the full list of operators. The following starts with the two most important ones, RETRIEVE and EXTRACT.

7.3.2.1 RETRIEVE Operator

Functionality. The RETRIEVE operator takes as input a *query* generated by the QUD stage. The output consists of *all* events from the personal data sources that soft-match the query, via matching against all *verbalized* key-value pairs.

Challenges. Unlike standard retrieval settings (as, e.g., in Chapter 6), where retrieval of few relevant evidence pieces is typically sufficient, the RETRIEVE operator has to obtain hundreds of relevant events, as many queries require large-scale aggregation. Missing only a small fraction of these would already distort the final answer. Therefore, we need to ensure that this operator has *high recall*, ideally near 100%, without adding too many false positives. Moreover, the qualifying events identified by RETRIEVE may have redundancy, as the same event can be mentioned in calendar, workout, social media, email, etc. This makes the de-duplication of redundant entries a crucial requirement of the RETRIEVE operator.

Execution. The RETRIEVE operator conceptually scans *all* personal data sources (e.g., music streams or mails) and can find matches among all their key-value pairs. However, our method aims to reduce the computational costs by early pruning of entire sources that do not contribute. This goal is achieved by executing the operator in a pipeline with five steps:

- **Step 1:** We use the efficient *sparse neural retriever* SPLADE [57] to obtain an initial pool of candidate events, aiming at high recall (i.e., minimizing the risk of missing a good result). All events with a non-zero score >0.1 are retained.
- **Step 2:** The resulting pool of events is analyzed to identify *patterns*: key-value pairs that occur frequently (i.e., above a specified frequency threshold) in the candidate pool; for example, `subject="Team meeting"` in calendar or `type="movie"`. In addition, for each source with matching events, we add a specific pattern (e.g., `source="mail"`), to score the entire source at once. The goal is to identify *sets of events* with shared properties, and score this set of events at once (in the next step), for efficient processing of the large input.
- **Step 3:** Finally, the patterns, each representing a set of events, are classified into three categories: (a) relevant (all events are kept), or (b) completely irrelevant (all events are dropped), or (c) partially relevant (events further filtered by key-value pairs). This step enables pruning out or accepting entire sets of events at once. An example would be disregarding all workouts with `type="gym"` for queries on football, by classifying this pattern into (b). Similarly, for a question about music, the pattern `source="music streams"` would be accepted with class (a), without scoring each of the potentially thousands of events individually.
- **Step 4:** A second classifier runs on all remaining events that have been assigned class (c) in step 3. It labels events into two bins: (a) keep for downstream operators or (b) drop.
- **Step 5:** Finally, to tame the potential redundancy of qualifying events (e.g., a calendar entry and a mail on the same meeting), we perform de-duplication. Whenever two events have overlapping time intervals, we combine them into a single event with all key-value pairs and the union of the time spans.

The pruning of entire sources in step 3 (e.g., music streams for queries on food or friends) yields a massive speed-up, and early elimination of frequent key-value pairs also contributes to the efficiency of REQAP. At the same time, to ensure high recall, our pipeline is designed to avoid unduly dismissing relevant events.

Training. The two classifiers used in steps 3 and 4 are implemented as *cross-encoders* (CE) [40, 122] for high precision. They are more expensive than the initial sparse retriever, but benefit cost-

wise from the staged pruning of their inputs. Our experiments use a standard CE¹, fine-tuned with examples for pattern and event classification.

We derive training data from our dataset: for each question, we identify relevant events as the ones used by the ground-truth query (as present in the PERQA dataset). For each RETRIEVE call in the operator tree, we would then add these events as positive training instances, and uniformly sample hard negatives from events retrieved by SPLADE. Patterns are labeled based on all matching events, as either (a) fully relevant, (b) fully irrelevant, or (c) partially relevant.

7.3.2.2 EXTRACT Operator

Functionality. The EXTRACT operator takes as input a set of *keys*, like attributes/field names, and a list of *events*, typically returned by an upstream RETRIEVE operator (i.e., child node in the operator tree). The operator generates values for all combinations of events and keys in the input, and augments events with these new key-value pairs. This enables structured processing of events and their additional key-value pairs in downstream operators (ancestors in the tree).

Challenges. The EXTRACT operator faces some very difficult cases when the input key does not semantically match any of the data keys (beyond simple cases such as matching “*day*” to *date* in calendar or workout data). The challenges arise when the operator’s input key requires *extracting* information from a longer text field, such as the body of a mail or the content of a social media post. Example keys would be “*participants*”, “*trip destination*”, or “*cuisine*” – none of which equals any existing data key. Note that the number of input events can often be in the order of hundreds or thousands, necessitating an efficient implementation.

Execution. Hence, we utilize a small-scale *sequence-to-sequence model*, which is given (i) the input key, (ii) the *verbalized* key-value pairs, and (iii) notable user information (e.g., a list of friends) as input, and generates a short phrase as output. For the running example about Italian food after football (question q^3 in Figure 7.1), the input key is “*cuisine*” and the value “*Italian*” is generated based on the phrase “*...pizza oven...*” in the mail body. For efficiency, EXTRACT automatically creates a frozen mapping for simpler input keys (e.g., “*date*”→*start_date*), covering cases where at least 70% of the first 50 inputs lead to the very same event-key (*start_date*). For the remaining input events, this mapping is then applied directly, avoiding further generation.

Training. To enable this rich behavior, we fine-tune a small Seq2seq LM, like BART [115]. We derive high-quality training data from our dataset, which provides parallel data of *canonicalized* events and their verbalized counterparts. Figure 7.4 shows an example of a canonicalized entry representing a meeting, and a corresponding social media post that is generated via a GLM.

7.3.2.3 Other Operators

There is a series of more standard operators currently implemented within REQAP. The following provides details, covering their inputs and outputs.

JOIN. The JOIN operator has the following inputs and outputs:

- **Input:** two *lists of events*, computed as intermediate results for two sub-questions by the upstream operators (descendants in the tree), and a *join condition* specifying a comparison between key-value pairs of the two input lists.
- **Output:** a set of *combined events* (the two sets of key-value pairs are merged) containing all event pairs from the two input lists that satisfy the join condition.

¹<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2>

Unlike joins in mainstream multi-hop QA [240], where all joins are equi-joins (based on equality conditions), our setting requires support for non-equi joins (e.g., based on inequality conditions), most notably, for temporal comparisons with before/after and overlap of time intervals.

Our method for this general join processing is a sort-merge algorithm. The two sets of events to be joined are sorted by the keys that the join predicate refers to, and the join result is obtained by sequentially scanning and merging the two ordered streams.

GROUP_BY. The **GROUP_BY** operator has the following inputs and outputs:

- **Input:** a list of events and a list of keys in this data, by which the events are grouped.
- **Output:** a partitioning of the events, represented as a list of lists. Each partition (i.e., one of the inner lists) consists of all events that share the same values for the input keys.

For the implementation, we adopted a hash-based algorithm that maps events into a hash table by their values of the grouping keys. The simplest case is a single key for grouping, but grouping by multiple keys can often be required as well: an example is for questions like “*in which month did I run the most?*”, which logically require grouping by month and year.

FILTER. The **FILTER** operator has the following inputs and outputs:

- **Input:** a set of events and a filter condition which is implemented as a function that can refer to key-value pairs (e.g., `cuisine=“Italian”`).
- **Output:** all events that satisfy the filter condition.

The implementation supports arbitrary filter conditions expressed as Python code, including exact matching, numeric matching (e.g., inequalities), temporal comparisons, or sub-string matching. Within the **FILTER** operator, calls to the QUD module are allowed as well, enabling a flexible framework for various situations.

MAP. The **MAP** operator has the following inputs and outputs:

- **Input:** a list of events, a Python function, and a key.
- **Output:** the list of input events, each augmented by a new key-value pair consisting of the input key and the return value of the input function applied to the respective event.

This operator is key for plugging in arbitrary functions into the execution plan, similar to user-defined functions (UDFs) in SQL. This is frequently needed, for example, for calendar-centric calculations, such as deriving the weekday of events or the duration of the period between dates. By enabling such plug-in functions, the REQAP system is easily extensible.

APPLY. The **APPLY** operator has the following inputs and outputs:

- **Input:** a list of events and a Python function.
- **Output:** the result of the function applied over the entire list of events, which is often a single scalar (e.g., for the length function).

The most important role of **APPLY** is to compute aggregation functions, such as the length of a list (for counting), summing up numeric values, etc. **APPLY** provides a single return value, the result of the input function applied to the entire input list – this is the key difference to the **MAP** operator that applies its input function to each event individually.

UNNEST. The SQL-inspired **UNNEST** operator has the following inputs and outputs:

- **Input:** a list of events, and a key pointing to the nested lists.
- **Output:** an extended list of events, which holds an independent event for each item in the nested lists, flattening this key-value pair.

An example use-case of **UNNEST** is a user question like “*Artist I listened to the most?*”, that requires grouping by individual artist names, while the events themselves have a key `artists`

holding multiple names. UNNEST creates an independent event for each artist name in the list, enabling the correct answering of such questions.

The implementation of APPLY, MAP and UNNEST is straightforward. The remaining operators (ARGMIN, ARGMAX, MIN, MAX, SUM, and AVG) are self-explanatory.

7.3.3 Cache for Sub-Question Results

When REQAP is deployed, we populate a *cache* of results for sub-questions, such as “*playing football*” or “*movie night*”, storing their returned list of events. The cached result can be directly utilized, without further execution, when questions need to evaluate the very same sub-question (this could be extended to semantic matches). Our implementation utilizes a *least recently used* (LRU) cache, and we set the maximum cache size to 100 by default. In our experiments, which include user data of multiple personas, we maintain one such cache per persona.

7.4 The PERQA Benchmark

7.4.1 Existing Benchmarks and their Limitations

There is a wealth of established benchmarks for all kinds of QA [173]. However, almost all of these are centered on objective contents about real-world facts, and thus unsuitable for our setting. QA over personal data needs to consider subjective events including personally sensitive details. Such data is subject to privacy laws, and this explains the lack of an appropriate benchmark.

To the best of our knowledge, the only existing data source that is suitable for our task is TIMELINEQA [206], which provides code for generating persona-based contents and questions. It allows generating personal data at large scale. However, the specified attributes for synthetic personas are basic, and user content is generated based on a limited number of simple templates. Further, complex questions are derived from solely 42 question templates (versus $\approx 2,000$ in PERQA). Thus, TIMELINEQA exhibits very limited diversity, not suitable to capture real-world challenges in personal QA.

7.4.2 Construction of the PERQA Benchmark

To bridge this gap in the existing literature, we devise a principled mechanism to create the PERQA dataset. The construction proceeds in three steps:

- (i) *persona collection*, in which we let humans (co-workers at our research group) craft realistic and detailed biographical information on *fictional persons*,
- (ii) *user data creation*, which makes use of the persona information (such as their music or travel preferences) and creates large amounts of realistic user data, and
- (iii) *question generation*, that generates complex questions and their ground-truth answers, for the personas and their user data.

Persona collection. The personas are created by having humans (from our research group) complete questionnaires with 30 fields about age, gender, family, education, work history, traveling, hobbies, fitness activities, interests in music and movies, food taste and other preferences. We collect 20 such human-crafted personas.

Persona details. For each fictional persona, the following information is collected: *name*, *gender*, *birth date/city*, *mother*, *father*, *siblings*, *kids* (including birth date), *pets* (including type, start/end date), *friends*, *education* (including city, start/end date), *career* (including company,

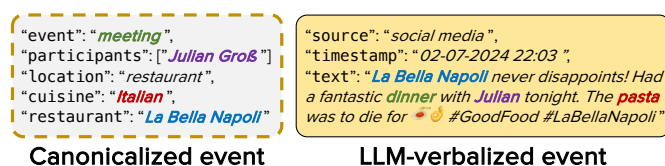


Figure 7.4: Example canonicalized event (*not* accessible at inference), with the corresponding verbalized version (accessible at inference).

city, start/end date), *residences* (including start/end date). Further, we collect user preferences: *favorite music genres*, *movie genres*, *TV series genres*, *shopping categories*, *travel regions*, *cuisines*, *hobbies* and *workouts*, as well as favorite instances (e.g., favorite songs) and the average frequency (e.g., average movies per week).

Canonicalized events. The next step is to create large-scale event repositories for each persona. In this stage, events are *canonicalized*: for each event type, events have a specific set of key-value pairs. This allows us to derive ground-truth answers later.

For streaming data (movies, TV series, music), we utilize Wikidata [220] to derive real-life entities matching the persona preferences (e.g., the favorite genres and songs). For example, for collecting candidate songs for a persona, we look up all songs from the user’s favorite music genres, as captured within Wikidata. Similarly, for online shopping, we utilize a public data dump² to obtain real-life products [148]. For realistic travel itineraries in-line with the persona information, we prompt GPT4O. For workouts, we utilize public data from Endomondo [149], which was a platform for tracking and sharing workouts. We align anonymized workout statistics of real users with the workout preferences specified in our human-crafted personas.

We further add special events, like personal milestones (marriage, birth of a child, new job, ...) based on the persona information. Finally, we include regular events like doctor appointments, meetings with friends, or New Year’s parties. This is done based on pre-defined probability and frequency values, inspired by the procedure in [206].

On average, for each persona, we create 40K such canonicalized events. The left side of Figure 7.4 provides an example canonicalized event.

Canonicalized event types. PERQA integrates the following canonicalized event types:

- *anniversaries* (birthday, child’s birthday, partner anniversary, valentines day, halloween, christmas, New Year’s Eve, pet anniversary);
- *doctor appointments* (dentist, gp, ophthalmologist, dermatologist, paediatrician, ...);
- *personal milestones* (new job/residence, start of new education, marriage, new child, ...);
- *traveling* (including 2-5 notable events);
- *meetings* (with friends and family, including restaurant visits etc.);
- *streaming* (music, TV series, movies);
- *online shopping*; and
- *workouts*.

Observable events. In reality, events such as restaurant visits, birthday parties or childbirth are usually not captured in a canonicalized form, but in unstructured text in calendar entries, mails and social media posts. We utilize LLAMA3.1³ to verbalize such events, and create realistic and diverse user data. Figure 7.4 illustrates the connection between canonicalized and verbalized

²<https://nijianmo.github.io/amazon>

³<https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct>

```

You are an assistant that generates questions, and the corresponding SQL query for deriving the answer, on
personal data.

You are provided with the SQL schema of the underlying DB, and a set of example questions and SQL queries.
Your task is to generate 25 questions and their SQL queries.

Make sure that the questions are natural, and the SQL query is compliant with DuckDB -- make use of the
provided SQL schema and examples. Please generate complex but realistic questions and SQL queries -- based
on multiple tables and columns, with joins, group by's, and variable declarations.

Also try to generate diverse questions: formulations can be ad-hoc or well-formed, and their sentence
structure could differ. Remember that questions should be natural: do not spell out the exact table names
or columns, but make it realistic.

Your output should be a plain JSON object, with no other text.

```

Figure 7.5: Our GLM instructions for generating complex questions. In addition, we randomly sample 20 questions (for each generation step) that are provided as ICL examples.

events. Streams, workouts and online purchases are kept in a canonicalized form with high probability, but are also verbalized with a relatively small probability.

Natural user behavior. A key criterion when creating user data is naturalness. Hence, we aim to derive sequences of events that are as natural as possible. Consider the music streaming behavior as an example: it is very unlikely that someone listens to a single song at 2AM and then to another song at 5AM. We thus base the sequences of events on real user behavior patterns. Such user behavior patterns are derived from streaming data of the authors and colleagues (only the patterns, no personally identifiable data is used). These patterns are then used to define the sequence of events for streaming data, workouts and online shopping.

Generation of complex questions. For generating questions, we handcraft a diverse set of 40 complex questions to start with – the same questions that are used for ICL in the QUD stage. We then iteratively sample up to 20 questions from this set, and prompt a GLM (with a temperature of 0.6) to generate 25 new questions. In addition to the questions, the ICL examples (and the model outputs) also contain SQL queries, run on a database with the canonicalized events, for deriving the ground-truth answers. Figure 7.5 shows the instruction used for creating these question-query pairs. We utilize GPT4O and LLAMA3.1⁴ to construct 1,000 questions each, for enhanced diversity. Generated question-query pairs are de-duplicated, and checked for correctness semi-automatically.

Cleaning of unnatural questions. The generation process may still yield some questions that are misinformed for specific personas. For example, there could be question templates asking for the birth date of the first-born child, but not all personas have children. Such *empty-answer* questions (no output, or a count of 0 for aggregations), are unnatural, so we removed them for the respective personas.

Deriving PERQA. We release PERQA with 20 personas, 40K events per persona, and 3,567 unique questions. This data is split along two dimensions, personas and question templates, to derive the train (12 personas, 1,200 templates), dev (2 personas, 170 templates), and test (6 personas, 600 templates) sets. Question templates are initiated based on the personas, to ensure natural and meaningful questions. The key statistics of PERQA are shown in Table 7.2. All data is in English. The personas collected are artificial, and do not represent real human beings.

Example questions. Table 7.3 shows representative example questions from our new PERQA dataset, demonstrating challenges such as aggregation, temporal conditions, numerical constraints, or the requirement of joining information from different sources.

⁴<https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct>

Table 7.2: Statistics for the PERQA dataset.

Category ↓ / Split →	Train	Dev	Test
Personas	12	2	6
Question templates	1,200	170	600
Unique questions	2,571	170	826
Movie streams	1,257	424	1,245
TV series streams	238	148	230
Music streams	33,754	31,616	38,257
Workouts	635	622	822
Online purchases	531	1,074	850
Calendar entries	1,552	1,367	2,106
Mails	506	490	636
Social media posts	670	579	819
Total events (per persona)	39,142	36,322	44,965

Example user data. Table 7.4 illustrates example events in PERQA. Such events can be of an unstructured, semi-structured or structured nature, and form the personal data in PERQA that systems have to process for answering the questions.

7.5 Experimental Setup

7.5.1 Configurations

We experiment with different options for the QUD stage: using **GPT4O** (“gpt-4o”) or **LLAMA3.3**⁵ with 8 ICL examples, or a 1B model⁶ fine-tuned as described in Section 7.3.1 (**SFT**).

We fine-tuned the QUD model for 4 epochs. The learning rate was tuned on a with-held dev set, and was eventually set to $5 * 10^{-6}$, with a linear scheduling. The warm-up ratio was set to 0.05, with a weight decay of 0.01, and a batch size of 4. Our train set for the QUD model has 17,933 instances, and the dev set 1,135. These are obtained by running the ICL version of REQAP (with GPT4O) on the train and dev sets, sampling $n=3$ generations per input with a temperature of 0.1. This results in multiple candidate operator trees per training instance – we retain only the ones leading to a correct answer, and a maximum of 3 per instance. In our experiments, we provide results for different QUD model sizes of the SFT version (default is 1B).

To initialize the **EXTRACT** operator, we use **BART**⁷, and the **RETRIEVE** operator uses **SPLADE**⁸ and a cross-encoder pre-trained on the MS MARCO dataset⁹ that is further fine-tuned to implement the two classifiers used in Section 7.3.2.1. As for the QUD model, we experiment with **RETRIEVE** and **EXTRACT** models of different sizes (XS, S, M and L) – an overview is provided in Table 7.8. Our default **RETRIEVE** models (size L) were fine-tuned for 10 epochs with a learning rate of $5 * 10^{-5}$, a batch size of 64, and a weight decay of 0.001. Our default **EXTRACT** model (size L) was fine-tuned for 3 epochs with a learning rate of $1 * 10^{-4}$, with a batch size of 16, and a weight decay of 0.01.

⁵<https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>

⁶<https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct>

⁷<https://huggingface.co/facebook/bart-base>

⁸<https://huggingface.co/naver/splade-cocondenser-ensembledistil>

⁹<https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2>

Table 7.3: Representative questions in the PERQA benchmark.

Questions – Workouts
<ul style="list-style-type: none">• “What is the total distance I ran in kilometers since I started working at Shopify?”• “My maximum heart rate during the most recent workout?”• “How many km did I run in the first week of 2019?”• “How many workouts did I do in 2018 with a maximum heart rate of more than 170?”
Questions – Music
<ul style="list-style-type: none">• “Which song did I listen to the most in the last month?”• “What is the name of the song I listened to the most in the summer of 2022?”• “Which song did I listen to most after a badminton workout?”• “which artist did I listen to most?”
Questions – Movies / TV series
<ul style="list-style-type: none">• “how many different movies did I watch?”• “the TV series I watched the most in summer 2022?”• “What’s the total number of hours spent on TV series?”• “Which day of the week do I stream TV series the most?”
Questions – Mail / Social media
<ul style="list-style-type: none">• “Who was the recipient of the first email I sent this year?”• “Who are my top 3 most contacted people via email?”• “How many emails did I send after 6 PM?”• “What was the last social media post I made?”
Questions – Others
<ul style="list-style-type: none">• “How many times did I meet with Carla Díaz at a restaurant?”• “What day of the week do I most frequently schedule doctor appointments?”• “What was the total amount spent on online purchases last year?”• “How many times did I meet at a park in 2022?”

Other QUD, RETRIEVE and EXTRACT models are trained with similar hyperparameters, slightly adapted depending on their size (all parameters are available in the configuration files of our public code¹⁰). All models were trained using an epoch-wise evaluation strategy, and with AdamW as optimizer. Our experiments are implemented in Python, and the REQAP operators also follow Python syntax and are executed as Python code.

Resources. We trained all models on a single GPU (NVIDIA Tesla H100NVL). Training of the QUD model took up to 2.5 hours (for the 3B LLAMA model), for the RETRIEVE models up to 1.5 hours (for the 33M parameter model), and also up to 1.5 hours for the EXTRACT model (139M version). For inference of the 70B LLAMA models, we used 4 GPUs (NVIDIA Tesla H100NVL).

7.5.2 Baselines

We compare REQAP against baselines for RAG and CODEGEN:

- **RAG:** This retrieval-augmented generation follows the *verbalization* paradigm. For retrieving relevant events, a two-stage technique is employed, with SPLADE retrieval followed by a fine-tuned cross-encoder (similar to Section 7.3.2.1).

¹⁰<https://reqap.mpi-inf.mpg.de>

Table 7.4: Representative user data for a fictional persona in PERQA.

Unstructured sources
<code>{"source": "mail", "subject": "Halloween was a blast!", "timestamp": "2016-11-01T10:44:21Z", "sender": "Isabella Ruiz", "recipient": "Lucia Hernández", "text": "Hey Lucia, hope you're doing well! I just wanted to share with you that I had a fantastic time celebrating Halloween with Carla last night. We stayed in and had a movie marathon, but it was still super fun. I loved the decorations and the costumes - Carla's was amazing! I'll have to show you the pictures soon. How about we plan something for this weekend? Maybe we can go for a hike or something? Let me know what you think. Take care, Isabella"}</code>
<code>{"source": "social_media", "text": "Pumped up after a grueling 117-minute weight training session. My heart rate reached a max of 182, but I managed to keep it steady throughout. Feeling strong and energized now! #weightlifting #fitnessmotivation"}</code>
Semi-structured sources
<code>{"source": "calendar", "start_date": "2024-08-19", "start_time": "12:00:00", "end_date": "2024-08-19", "end_time": "13:00:00", "summary": "Lunch with Mum and Dad", "location": "The Parthenon", "description": "Greek food"}</code>
Structured sources
<code>{"source": "online_purchase", ..., "order": "1 x Cosmic Funk", "product_quantity": "1", "product": "Cosmic Funk", "category": "CDs & Vinyl", "price": "5.99 EUR"}</code>
<code>{"source": "tvseries_stream", ..., "duration": "2460", "stream_end_time": "00:24:17", "stream_style": "tv_series", "tvseries_title": "Scrubs", "season_name": "Scrubs, season 1", "episode_name": "My Nickname", "episode_number": "10", "tvseries_season": "1", "duration_unit": "seconds"}</code>
<code>{"source": "movie_stream", ..., "duration": "6660", "stream_end_time": "01:10:41", "stream_style": "movie", "movie_title": "Beverly Hills Cop III", "stream_full_title": "Beverly Hills Cop III", "duration_unit": "seconds"}</code>
<code>{"source": "workout", ..., "workout_type": "soccer", "duration": "126", "duration_unit": "min", "minimum_heart_rate": "120", "maximum_heart_rate": "188", "average_heart_rate": "156.87"}</code>

- **CODEGEN:** We also compare against a code generation baseline following the *translation* paradigm. The input is the user question and the SQL schema of the personal data, and the output is a SQL query. For fine-tuning, we utilize SQL queries that lead to the correct answer (similar to Section 7.3.1).

For fair comparison, these baselines use the same base models as the QUD stage of REQAP: ICL with 8 examples using GPT4O or LLAMA3.3, or a fine-tuned decoder-only LM.

Baselines follow the same training scheme used for REQAP. For the RAG baseline, we tuned the number of events in the input on the dev set, and found the performance to be best with 100 events (with GPT4O and LLAMA3.3) and 10 events (with the SFT variant). Note that including >500 events exceeded the context window of GPT4O and LLAMA3.3.

7.5.3 Metrics

We measure *hit at 1* (**Hit@1**) as our main metric for personal QA, where users expect a single concise answer. As some questions aggregate over hundreds of events, we also measure *relaxed hit at 1* (**Rlx-Hit@1**), which allows a $\pm 10\%$ slack for numeric answers. Both metrics are micro-averaged over all questions in the test set. Statistical significance of improvements by REQAP compared to the best baseline (for each model) is measured via McNemar’s test with $p < 0.05$, and is indicated by an asterisk (*).

Table 7.5: Main results on PERQA *test* set. Metric is Hit@1, and Rlx-Hit@1 is in parentheses.

Model → Method ↓	GPT4O (≥100B)	LLAMA3.3 (70B)	SFT (1B)
RAG	0.149 (0.20)	0.123 (0.18)	0.029 (0.06)
CODEGEN	0.319 (0.44)	0.239 (0.33)	0.315 (0.47)
REQAP (ours)	0.386* (0.52)*	0.322* (0.46)*	0.380* (0.53)*

7.6 Experimental Results and Insights

7.6.1 Key Findings

REQAP wins over baselines. Table 7.5 shows the main results on PERQA. First of all, REQAP substantially outperforms baselines for each variant. The difference is most pronounced with LLAMA3.3, where REQAP improves Hit@1 by 8.3 percentage points compared to CODEGEN, the best baseline. The same observation holds for Rlx-Hit@1, with similar improvements over baselines.

RAG fails on PERQA. While the baselines based on CODEGEN remain competitive, RAG performs poorly. It cannot cope with questions that need to aggregate (sum, average, ordering, etc.) over tens or hundreds of events.

SFT is a viable solution for on-device inference. The variants based on GPT4O and LLAMA3.3 are useful for comparison purposes, but cannot be run directly on user devices. Remarkably, our SFT variant, which is suitable for such on-device inference, exhibits competitive performance even with a 100x smaller model size. Its Hit@1 of 0.380 is almost as good as the best observed score of 0.386 (by REQAP with GPT4O), and its relaxed Hit@1 of 0.53 is the best across the board.

Approximate answers are good. The questions in PERQA are difficult for all methods, with Hit@1 being below 40% throughout the field. Nevertheless, REQAP often yields decent order-of-magnitude answers. An indicator is the Rlx-Hit@1 metric, for which REQAP achieves 53% with the small SFT variant. Recall that this allows $\pm 10\%$ slack, which is on the conservative side. Even more relaxed approximations could still be tolerable by users.

7.6.2 Analysis

REQAP performs well across question types. We categorized questions in PERQA into different complexity types:

- Ordering (e.g., “Which song did I listen to the most during my last trip?”),
- Grouping (e.g., “which four restaurants do I go to the most?”),
- Temporal (e.g., “How many times did I visit the doctor in the last 3 years?”),
- Aggregation (e.g., “What was the total amount spent on online purchases last year?”),
- Join (e.g., “How many times did I watch a movie after a run?”), and
- Multi-source (e.g., “which of my trips was the shortest?”).

Table 7.6: Performance by complexity types of questions.

Complexity ↓	No. of questions	RAG (GPT4O)	CODEGEN (GPT4O)	REQAP (GPT4O)
Ordering	1,236	0.167	0.440	0.529
Grouping	931	0.172	0.444	0.537
Temporal	1,945	0.129	0.290	0.417
Aggregation	1,790	0.130	0.228	0.296
Join	369	0.073	0.176	0.236
Multi-source	1,710	0.196	0.237	0.365
All	3,044	0.149	0.319	0.386

Table 7.7: Ablation studies on the PERQA *dev* set.

Method ↓ / Metrics →	Hit@1	Rlx-Hit@1
REQAP (SFT)	0.396	0.54
w/o QUD – decomposition	0.356	0.50
w/o RETRIEVE – cross-encoders	0.269	0.36
w/o EXTRACT – LM generation	0.138	0.23

Note that each question can be assigned multiple such complexity types.

Results for REQAP and baselines on questions of different complexity types are shown in Table 7.6. REQAP consistently outperforms baselines, and performs especially well on questions that require ordering or grouping. Join questions are the most challenging for all methods.

RETRIEVE and EXTRACT operators are decisive. Table 7.7 shows the results for ablation studies of REQAP, investigating its performance when

- (i) the operator tree is generated in one shot, *without recursive decomposition*;
- (ii) all events retrieved by SPLADE are retained; *without cross-encoders*,
- (iii) values are extracted from events only by exact matches of keys, *without LM generation*.

Performance degrades dramatically for each of these variants. Most notably, the devised implementation of the RETRIEVE and EXTRACT operators substantially contributes to REQAP’s performance. Also, the recursive decomposition of the question intent strikes one as a key asset of REQAP as generating the operator trees in a single step is prone to errors.

REQAP adapts to low-resource computation. Our goal is a method that can run on user devices. However, there is high variance in the available compute-power and memory on user hardware, and QUD alone may use cloud services without violating data protection. Hence, we trained neural components of REQAP based on models of varying sizes: from 135M to 3B parameters for the QUD model, 4M to 33M for the cross-encoders of RETRIEVE, and 16M to 139M for the EXTRACT model. Table 7.8 lists the details of the models used. Note that even the larger ones are small compared to GLMs commonly used on servers. Table 7.9 shows the results.

While there is a substantial performance gap between the smallest variant (QUD: XS, Operators: XS) and the largest variant (L, L), we observe interesting trade-offs and compromises. When the memory footprint has to be low, but computational costs and run-time are secondary, the (XS, L) variant is a good choice, achieving an Rlx-Hit@1 of 51%. Recall that QUD runs on the user question only, *without* access to user data. When questions are not privacy-sensitive, a larger cloud model can be used for this stage.

Table 7.8: Base models used for the model-size analysis of REQAP in Table 7.9.

Module	Size	#Parameters	Base model
QUD	XS	135M	HuggingFaceTB/SmolLM2-135M-Instruct
	S	360M	HuggingFaceTB/SmolLM2-360M-Instruct
	M	1B	meta-llama/Llama-3.2-1B-Instruct
	L	3B	meta-llama/Llama-3.2-3B-Instruct
RETRIEVE	XS	4M	cross-encoder/ms-marco-TinyBERT-L-2-v2
	S	16M	cross-encoder/ms-marco-MiniLM-L-2-v2
	M	23M	cross-encoder/ms-marco-MiniLM-L-6-v2
	L	33M	cross-encoder/ms-marco-MiniLM-L-12-v2
EXTRACT	XS	16M	google/t5-efficient-tiny
	S	31M	google/t5-efficient-mini
	M	70M	lucadiliello/bart-small
	L	139M	facebook/bart-base

Table 7.9: Effect of model sizes of the individual modules on the end-to-end performance (Hit@1) of REQAP. The metric is Hit@1, with Rlx-Hit@1 being in parentheses.

	QUD → Operators ↓	XS (135M)	S (360M)	M (1B)	L (3B)
XS	RETRIEVE (4M)	0.167	0.185	0.193	0.193
	& EXTRACT (16M)	(0.28)	(0.28)	(0.31)	(0.30)
S	RETRIEVE (16M)	0.240	0.287	0.302	0.302
	& EXTRACT (31M)	(0.37)	(0.41)	(0.42)	(0.42)
M	RETRIEVE (23M)	0.331	0.353	0.378	0.389
	& EXTRACT (70M)	(0.48)	(0.51)	(0.50)	(0.52)
L	RETRIEVE (33M)	0.356	0.364	0.396	0.400
	& EXTRACT (139M)	(0.51)	(0.52)	(0.54)	(0.55)

Table 7.10: Effect of using only structured or unstructured sources on the performance of the SFT variant of REQAP (on the PERQA dev set).

Method ↓ / Metrics →	Hit@1	Rlx-Hit@1
REQAP (All sources)	0.396	0.54
w/ structured sources only	0.236	0.36
w/ unstructured sources only	0.189	0.26

Combining structured and unstructured sources leads to substantial improvements.

We conducted ablation studies to investigate the importance of integrating both structured and unstructured sources within REQAP. We either utilized *only structured sources* (i.e., music, movie and TV series streaming, online shopping, and workouts) or *only unstructured sources* (i.e., calendar entries, mails, and social media posts) for answering questions. The results are shown in Table 7.10. In both cases, the answering performance drops substantially, for the Hit@1 and the Rlx-Hit@1 metric. This demonstrates the importance of integrating heterogeneous sources for QA over personal data.

7 Personal Question Answering

Table 7.11: In-depth results for PERQA *test* personas 1-3. The metric is Hit@1, and Rlx-Hit@1 is in parentheses.

Method ↓	Persona 1 (489 questions)			Persona 2 (495 questions)			Persona 3 (530 questions)		
	GPT4o	LLAMA3.3	SFT	GPT4o	LLAMA3.3	SFT	GPT4o	LLAMA3.3	SFT
RAG	0.180 (0.26)	0.174 (0.25)	0.037 (0.06)	0.154 (0.21)	0.125 (0.19)	0.038 (0.06)	0.147 (0.21)	0.117 (0.17)	0.032 (0.06)
CODEGEN	0.344 (0.45)	0.254 (0.33)	0.358 (0.48)	0.349 (0.47)	0.269 (0.35)	0.337 (0.48)	0.326 (0.46)	0.240 (0.33)	0.319 (0.48)
REQAP (ours)	0.421 (0.54)	0.348 (0.46)	0.427 (0.56)	0.402 (0.54)	0.331 (0.47)	0.394 (0.53)	0.425 (0.56)	0.353 (0.49)	0.404 (0.55)

Table 7.12: In-depth results for PERQA *test* personas 4-6. The metric is Hit@1, and Rlx-Hit@1 is in parentheses.

Method ↓	Persona 4 (460 questions)			Persona 5 (524 questions)			Persona 6 (546 questions)		
	GPT4o	LLAMA3.3	SFT	GPT4o	LLAMA3.3	SFT	GPT4o	LLAMA3.3	SFT
RAG	0.133 (0.17)	0.089 (0.15)	0.015 (0.08)	0.166 (0.22)	0.130 (0.17)	0.029 (0.06)	0.119 (0.17)	0.103 (0.16)	0.024 (0.04)
CODEGEN	0.272 (0.41)	0.209 (0.31)	0.285 (0.46)	0.307 (0.44)	0.227 (0.32)	0.292 (0.45)	0.315 (0.45)	0.234 (0.33)	0.299 (0.46)
REQAP (ours)	0.341 (0.49)	0.300 (0.45)	0.357 (0.52)	0.385 (0.53)	0.313 (0.45)	0.368 (0.52)	0.342 (0.49)	0.286 (0.43)	0.335 (0.50)

Error analysis. We manually investigated 50 incorrect answers judged by the Rlx-Hit@1 metric. REQAP failed because of

- (i) an incorrect operator tree (50%),
- (ii) errors by the RETRIEVE operator (28%), or
- (iii) errors by the EXTRACT operator (22%).

Note that often “small” mistakes, such as few false negatives or false positives in the RETRIEVE operator can lead to an incorrect answer, underlining the difficulty of PERQA.

REQAP is robust across personas. Table 7.11 and Table 7.12 provide results on the test set of PERQA for each persona individually. All of the methods show variance across the different personas. Notably, REQAP performs better than baselines across all personas and models.

Run-time analysis. We conducted a run-time analysis for REQAP, utilizing the QUD model of size M and operators of size L. A single GPU (NVIDIA Quadro RTX 8000, PCI, 48 GB GDDR6) was used for this experiment. We measured an average run-time of 0.56 s (=seconds) per question for the QUD stage, and an average run-time of 11.53 s for the OTX stage, leading to an average run-time of 12.09 s. The median run-time is much lower with 3.57 s. Table 7.13 shows the average run-time per operator. Unsurprisingly, the RETRIEVE and EXTRACT operators, that are based on LMs and run on thousands of inputs for each question, are the most computationally expensive.

Anecdotal examples. Table 7.15 shows anecdotal examples for operator trees generated by our QUD stage.

Table 7.13: Average operator run-times.

Operator	Average run-time
RETRIEVE	2.61 s
EXTRACT	5.93 s
JOIN	2.29 s
GROUP_BY	4.82×10^{-3} s
FILTER	1.65×10^{-2} s
MAP	3.44×10^{-1} s
APPLY	4.82×10^{-3} s
UNNEST	3.36×10^{-1} s
ARGMIN	3.36×10^{-1} s
ARGMAX	8.61×10^{-5} s
MIN	1.74×10^{-4} s
MAX	1.33×10^{-4} s
SUM	1.20×10^{-4} s
AVG	1.13×10^{-4} s

Table 7.14: Results on TIMELINEQA. The metric is Hit@1, and Rlx-Hit@1 is in parentheses.

Model → Method ↓	GPT4o ($\gg 100$ B)	LLAMA3.3 (70B)	SFT (1B)
RAG	0.321 (0.36)	0.369 (0.41)	0.214 (0.22)
CODEGEN	0.262 (0.33)	0.187 (0.22)	0.135 (0.17)
REQAP (ours)	0.357 (0.39)	0.286 (0.31)	0.313 (0.35)

7.6.3 Experiment with TIMELINEQA

Setup. Although TIMELINEQA is deemed insufficient for our setting, we obtained insight on how REQAP performs on these questions. For this purpose, we used their 42 templates for complex questions, and adapted them to our realistic user data. We ran the questions for each of the 6 test personas in PERQA. SFT variants utilize the models trained on PERQA, as the limited number of questions in TIMELINEQA does not allow for training.

Results. Table 7.14 shows the results on TIMELINEQA. RAG shows stronger performance than on the more challenging questions in PERQA. However, REQAP substantially outperforms both RAG and CODEGEN, with the small model (SFT) and with GPT4o.

7.7 User Study for Evaluation

The PERQA benchmark is designed with synthetic personas to avoid privacy issues. The goal of this user study is two-fold:

- (i) analyze whether the questions in PERQA are comparable with real user questions, and
- (ii) investigate whether a small-scale variant of REQAP, trained with PERQA data and run on end user devices, can deal with such realistic information needs and user data.

7.7.1 Setup

We conducted our user study with 20 undergraduate students. First, we instructed participants to export their personal data from various services like Apple, Netflix and Spotify. Then, users were provided with Docker images¹¹ for running REQAP on their own laptops and exports, but without giving us access to their personal data. The Docker images had disabled Internet connectivity, to guarantee the impossibility of data leaks.

Question creation. Each participant of our user study created 100 questions, that capture their real information needs on their personal data, first. We provided as little instructions as possible, to avoid biasing them towards a specific type of question. For evaluation, the participants then chose 20 questions that are answerable with their personal data, the sources we support, and the data in current exports, assuming a hypothetically perfect QA system. For example, the Spotify exports for music streams would not include the genre or album of the songs, which appeared in many user questions. Similarly, we observed questions about watch lists on Netflix, like “*What is the name of the movies or series that has been in the user Emma’s lists for the longest time?*”, that are not covered in current data exports. All 2,005 questions (some participants provided more than 100) collected in our user study, including the ones not used for evaluation, are available at <https://reqap.mpi-inf.mpg.de>.

REQAP model sizes. When conducting the user study, we used a preliminary version of the QUD model. Specifically, we used CODET5-LARGE [221] with 738M parameters, fine-tuned as outlined in Section 7.3.1¹². We later found variants based on LLAMA-1B to perform substantially better (on PERQA), while demonstrating comparable or faster run-times. Notably, compared to Seq2seq LMs, decoder-only models (i.e., GLMs) can be run in a quantized manner much more easily. We hypothesize that the usage of LLAMA-1B might have further improved the results of the user study. For the RETRIEVE operator, we used the model of size S (16M), and for the EXTRACT operator we used XS (16M).

Payment. We hired local students. Participants were paid \$15 per hour for the user study, which is well above the local minimum wage.

7.7.2 Results

Real user questions and PERQA questions are similar. We ran our QUD module on user and benchmark questions, and compared the structure of the resulting operator trees, abstracting away the individual constants. For 376 out of 400 (=20 users×20 questions) user questions (94%), we were able to map them to an isomorphic operator tree in PERQA. This indicates that the questions in PERQA indeed capture the complexities of real-life information needs.

REQAP performs well in the wild. For each of their 20 selected questions, users ran REQAP on their own device and data. We asked them whether the provided answer is *correct*, *almost correct* (with minor drifts tolerated by the user), or *incorrect*. By design, REQAP allows users to identify which of their events were used for answering, aiding them to understand the correctness of answers. Users judged about 28% of the answers for their own questions as fully correct, and 41% as almost correct.

Users also evaluated 20 PERQA questions against their personal data, with allowance for minor adjustments to adapt them to their data (e.g., for dates to ensure they fall into the user’s

¹¹<https://docker.com>

¹²<https://huggingface.co/Salesforce/codet5-large>

time frame). In this setting, REQAP achieves even better performance: 45% for Hit@1 (and 60% for Rlx-Hit@1).

Anecdotal failure case. QA over heterogeneous personal data often requires aggregating or ordering information from hundreds of events. For such questions, even minor errors in one of the operators can lead to an incorrect answer – as observed in PERQA and in the user study – making the task a real challenge even with the latest advances in NLP.

We report one such anecdote from the user study (we asked the respective user for consent on reporting this anecdote): for the user query “*most expensive purchase*”, REQAP derived the correct answer among the top few results (based on the available user data). However, a mail from a mailing list, mentioning that an apartment was available for a rent of 730€, was incorrectly classified as relevant by the RETRIEVE operator. The corresponding value, 730€ was then found to be the highest, leading to an incorrect answer. Note that such a case would yield a Hit@1 and Rlx-Hit@1 of 0 on PERQA.

7.8 Demonstration

We provide an interactive demo for REQAP at <https://reqap.mpi-inf.mpg.de/demo>, which runs on synthetic user data from the PERQA benchmark and illustrates the answering process of REQAP. Users can trace back answers to the corresponding events. The following considers the example question:

q: How often did I eat Italian food after a yoga workout?

which is in-line with the preferences of one of the artificial personas used for this demo (unlike our earlier running example on playing football).

7.8.1 Personal User Data

User data. For the demo, we utilize five personas from the PERQA dataset, and selected 4,000 user events to enable browsing and searching all of them. These synthetic personas are hand-crafted, with details on their family, career, personal milestones and preferences (cf. Section 7.4). For running REQAP locally on their own data, users can download our Docker images¹³.

Data exports. REQAP loads data exports from the respective services. Our current implementation supports data exports from Amazon Marketplace, Spotify, Netflix, Apple, Google, as well as mails in the “*mbox*” format and calendar data in the “*ics*” format. Note that it is trivial to extend our implementation with additional services, such as streaming data from Amazon Prime. This simply requires providing the data in the format expected by REQAP, i.e. as sets of key-value pairs.

User data browser. The user can browse through their data with an interface similar to the one shown for related events in Figure 7.7. Our demo also shows the handcrafted persona information, for better understanding of the large-scale personal data. This is not used within the REQAP algorithm, as such structured personal information is unlikely to be readily available in reality.

¹³<https://github.com/PhilippChr/ReqAP-docker>

Source: Calendar (Event 1 / 3)	Source: Calendar (Event 2 / 3)
Start date: 2021-09-11 Start time: 18:00:00 End date: 2021-09-11 End time: 19:00:00	Start date: 2021-09-11 Start time: 16:30:00 End date: 2021-09-11 End time: 17:00:00
Description: Italian food with Mum, Dad and Kalle Location: L'Antica Locanda Summary: Family dinner	Description: Location: Summary: Yoga

Figure 7.6: Relevant user events for the example question “*How often did I eat Italian food after a yoga workout?*”, as shown in our REQAP demo.

7.8.2 REQAP Inference

REQAP configuration. In our demo interface, the user can select the answering sources. For the sake of the demo, questions can be run on data of various PERQA personas. The default reference date (2024-11-25) considers the last date of any user event in the PERQA dataset, to avoid empty answers for temporal constraints like “... *this year*”. The interface also enables configuring the QUD generation, which could be disabled in the case of lay users.

Answer tracing. Figure 7.7 shows an example of how the answer derivation can be traced by the end user, for the question “*How often did I eat Italian food after a yoga workout?*”. In each decomposition step, the user can see the operator, the sub-question(s) and related user events. For the first operator (APPLY), there is only a single result. This result consists of three events: a calendar entry about a visit of an Italian restaurant, a calendar entry with `summary=“yoga”`, and the social media post shown in Figure 7.7. Figure 7.6 displays the other two entries. The two events on yoga are duplicate entries, discussing the same real-life event, as identified in step 5 of our RETRIEVE operator (cf. Section 7.3.2.1). For such join results and duplicated events, we allow for investigating the entries by clicking the left/right arrows, similar to a slideshow. By enabling browsing of relevant user events, our interface helps users verify the answer correctness.

Below these related events, the interface illustrates the decomposition of the question, with a JOIN and an EXTRACT operator. For resolving the user condition of only considering yoga workouts, REQAP applies a FILTER to prune out other workouts, based on string matching in the `workout_type` key of results. This key (`workout_type`) is not readily accessible for most sources (e.g., the calendar and social media entries in Figure 7.6). Hence, the subsequent operator resolving “*my workouts with workout type*” applies an EXTRACT operator to add this structured key and enable the FILTER operation.

To avoid overwhelming users, only the crisp answer, the first operator and its relevant events are shown initially. The decomposition steps and details on relevant events can be conveniently accessed by clicking collapsible items.

Debugging. The interface is also useful for developers, as it enables viewing both correct and failed operator trees. Further, per-event details for the RETRIEVE and EXTRACT operators are visible. For the RETRIEVE operator, the demo shows how the event was classified (via a pattern or via per-event classification), and the scores obtained by sparse retrieval and the classifier. For the EXTRACT operator, users can explore the extracted key-value pairs for each event.

"How often did I eat Italian food after a yoga workout?"

Answer:
1

Computation: (0 s)
APPLY(
 l=**QUD**("instances of eating Italian food after a yoga workout"),
 fct=len
)

Show Related User Events (1)

End datetime	Start datetime	Workout type	User event(s)
2021-09-11 17:00:00	2021-09-11 16:30:00	yoga	Source: Social media (Event 3 / 3) Start date: 2021-09-11 Start time: 17:21:29 End date: End time: Text: Zen mode activated 🧘. Wrapped up a 39-minute yoga session, feeling centered and refreshed. Heart rate stayed in check, averaging 85.08 bpm. Namaste, indeed 🙏 #YogaLove #FitnessJourney #MindfulMoments

Showing 1 to 1 of 1 entry

"instances of eating Italian food after a yoga workout"

Computation: (0.001 s)
JOIN(
 l1=**QUD**("instances of eating Italian food with start date and time"),
 l2=**QUD**("my yoga workouts with end date and time"),
 condition="l1.start_date == l2.end_date and l1.start_time >= l2.end_time"
)

Show Related User Events (1)

"instances of eating Italian food with start date and time"

"my yoga workouts with end date and time"

Computation: (0.022 s)
EXTRACT(
 l=**QUD**("my yoga workouts"),
 attr_names=["end_date", "end_time"],
 attr_types=[date.fromisoformat, time.fromisoformat]
)

Show Related User Events (276)

"my yoga workouts"

Computation: (0 s)
FILTER(
 l=**QUD**("my workouts with workout type"),
 filter=lambda attr: "yoga" in attr["workout_type"].**lower**()
)

Show Related User Events (276)

"my workouts with workout type"

Figure 7.7: Screenshot of the REQAP demo interface for the question "How often did I eat Italian food after a yoga workout?", showing the decomposition steps, operators and related events, for tracing answer derivation.

7.8.3 Implementation

The REQAP backend is implemented using Python, with `Flask` for hosting API endpoints. The frontend is also based on `Flask`, and makes use of HTML, CSS and JavaScript. For the general design we leverage `Bootstrap`¹⁴, and `DataTables`¹⁵ for our searchable tables.

Besides our public code for the REQAP¹⁶ methodology and our online demo¹⁷, we provide a Docker image and instructions to process personal data from popular services for input in REQAP, and Docker images for frontend and backend¹⁸.

7.8.4 Usability

We utilized a version of the REQAP interface for our local user study (Section 7.7), enabling the users to answer their questions against their own personal data. Our study had 20 local undergraduate students interact with REQAP, each answering 20 questions (=total of 400 judgments). Users were provided with Docker containers to run our system on their own devices, this way eliminating data leakage risks. We asked users about their (i) *certainty* of the answer correctness, and (ii) their *understanding* of the answer derivation process. Users provided scores on a scale from 0 – 10 in both cases. We found that users generally understood whether the answer is correct or not with a high certainty of 8.6 (out of 10) on average. Their understanding of the individual steps of the answer derivation was slightly lower with 7.0 (out of 10).

7.9 Related Work

Personal data management. This theme was initially motivated by the tasks of desktop search [47] and document search [50]. With the increasing abundance of personal data on mobile phones and cloud storage, the topic has been revived [73]. Recent works target simple retrieval over personal data [71, 171]. The closest work to ours is TIMELINEQA [206], which conducts experiments with standard RAG and CODEGEN approaches – we show that REQAP substantially outperforms these state-of-the-art baselines.

Orthogonally, there are concerns about platform providers “knowing everything” about users, and the need for humans to stay in full control of their personal data [102, 136]. Our approach follows this rationale.

Data verbalization. A popular way of using GLMs for multi-source QA is to verbalize all data and treat it uniformly as if were a collection of natural-language sentences [154]. This approach has been pursued for the case of text and tables [4, 245], and for combining knowledge graphs with other sources [33, 34, 160, 252]. We compare REQAP against the verbalization paradigm, which is also utilized in previous chapters of this dissertation, in our experiments.

Question-to-query translation. This paradigm is rooted in early works on semantic parsing (e.g., [10, 235, 244]). It has been reshaped and boosted with the advent of large-scale GLMs. State-of-the-art methods, often referred to as Text2SQL or NL2SQL [106, 127], leverage GLM pre-training with code repositories (e.g., GitHub) and fine-tune the model with question-query pairs [53, 59, 65, 118, 159]. Importantly, these methods require *schema* information about the tables, column headers and column types of the underlying sources. For complex schemas or

¹⁴<https://getbootstrap.com>

¹⁵<https://datatables.net>

¹⁶<https://reqap.mpi-inf.mpg.de>

¹⁷<https://reqap.mpi-inf.mpg.de/demo>

¹⁸<https://github.com/PhilippChr/ReQAP-docker>

when only partial information (e.g., without expressive typing) is provided, the performance drops sharply.

Question decomposition. This emerging approach decomposes natural-language questions in a step-wise manner. It resembles the semantic parsing paradigm, but a key difference is that question decomposition is agnostic of the underlying data sources. Instead, it aims at understanding the user’s intent and deriving a suitable representation. By decomposing a user question, REQAP aims to understand the user’s intent and derive a suitable representation. This has been pursued for a variety of specialized settings, including our FAITH project (Chapter 5) for temporal questions, and in SUQL [125] and Saeed et al. [176] for delegating sub-tasks to a GLM. Also, enhanced prompting techniques like chain-of-thought [222] and tree-of-thoughts [242] have been proposed. While these approaches enable GLMs to decompose tasks of certain kinds, they exhibit similar drawbacks as RAG, and do not easily generalize to QA over heterogeneous data sources. All these works are fairly different from our setting.

7.10 Conclusion

This chapter introduced a new approach, REQAP, for QA over heterogeneous personal data, spanning structured tables and unstructured text. REQAP recursively decomposes a user question into an operator tree, which can be efficiently executed on user devices. In our experiments, REQAP outperforms methods based on verbalization (RAG) or translation (CODEGEN), and our small-model configurations achieve competitive performance, thus reconciling the goals of expressive QA and lightweight footprint. Our framework allows for flexible configurations, such as invoking the question decomposition on a cloud-based server with a large LM (without seeing any user data) and executing operator trees with a small model on the user’s mobile.

Compared to previous work in this dissertation, REQAP enables targeting much more complex questions, involving advanced operations (such as aggregation, grouping or joins) with large sets of items (items are user events in this context), often of the order of hundreds or thousands. This is in contrast to the other chapters, that mostly target questions for which combining information from a few (ten or less) evidence pieces is sufficient.

Table 7.15: Anecdotal examples for operator trees generated by REQAP.

Anecdotal Example 1 – “How much money did I spend on online purchases in March 2022?”
(0) $QUD("How\ much\ money\ did\ I\ spend\ on\ online\ purchases\ in\ March\ 2022?")$
(1) $SUM(l=QUD("my\ online\ purchases\ in\ March\ 2022\ with\ amounts"),\ attr_name="amount_spent")$
(2) $EXTRACT(l=QUD("my\ online\ purchases\ in\ March\ 2022"),\ attr_names=["amount_spent"],\ attr_types=[float])$
(3) $FILTER(l=QUD("my\ online\ purchases\ with\ date"),$ $filter=\lambda attr: attr["purchase_date"].year == 2022\ and\ attr["purchase_date"].month == 3)$
(4) $EXTRACT(l=QUD("my\ online\ purchases"),\ attr_names=["purchase_date"],\ attr_types=[date.fromisoformat])$
(5) $RETRIEVE(query="my\ online\ purchases")$
Anecdotal Example 2 – “First football training after I started as Engineer – when was it?”
(0) $QUD("First\ football\ training\ after\ I\ started\ as\ Engineer\ -\ when\ was\ it?")$
(1) $MIN(l=QUD("football\ training\ sessions\ after\ I\ started\ as\ Engineer"),\ attr_name="start_datetime")$
(2) $FILTER(l=QUD("my\ football\ training\ sessions\ with\ datetime"),$ $filter=\lambda attr: attr["start_datetime"] >= QUD("first\ start\ datetime\ as\ Engineer").result)$
(3.1) $EXTRACT(l=QUD("my\ football\ training\ sessions"),\ attr_names=["start_datetime"],$ $attr_types=[datetime.fromtimestamp])$
(4.1) $RETRIEVE(query="I\ played\ football")$
(3.2) $MIN(l=QUD("start\ datetime\ as\ Engineer"),\ attr_name="start_datetime")$
(4.2) $EXTRACT(l=QUD("I\ started\ as\ Engineer"),\ attr_names=["start_datetime"],$ $attr_types=[datetime.fromtimestamp])$
(5.2) $RETRIEVE(query="I\ started\ as\ Engineer")$
Anecdotal Example 3 – “which restaurants did we visit when in Bali, Indonesia”
(0) $QUD("Which\ doctor's\ appointment\ was\ the\ earliest\ in\ the\ day?")$
(1) $EXTRACT(l=QUD("restaurants\ we\ visited\ in\ Bali,\ Indonesia"),\ attr_names=["restaurant_name"],$ $attr_types=[str])$
(2) $JOIN(i1=QUD("restaurants\ we\ visited\ with\ date"),$ $i2=QUD("we\ were\ in\ Bali,\ Indonesia\ with\ start\ and\ end\ date"),$ $condition="i1.visit_date >= i2.start_date\ and\ i1.visit_date <= i2.end_date")$
(3.1) $EXTRACT(l=QUD("restaurants\ I\ visited"),\ attr_names=["visit_date",\ "restaurant_name"],$ $attr_types=[date.fromisoformat,\ str])$
(4.1) $RETRIEVE(query="restaurants\ I\ visited")$
(3.2) $EXTRACT(l=QUD("I\ was\ in\ Bali,\ Indonesia"),\ attr_names=["start_date",\ "end_date"],$ $attr_types=[date.fromisoformat,\ date.fromisoformat])$
(4.2) $RETRIEVE(query="I\ was\ in\ Bali,\ Indonesia")$
Anecdotal Example 4 – “Which doctor’s appointment was the earliest in the day?”
(0) $QUD("Which\ doctor's\ appointment\ was\ the\ earliest\ in\ the\ day?")$
(1) $ARGMIN(l=QUD("my\ doctor's\ appointments\ with\ start\ time"),\ arg_attr_name="start_time",$ $val_attr_name="appointment_details")$
(2) $EXTRACT(l=QUD("my\ doctor's\ appointments"),\ attr_names=["start_time",\ "appointment_details"],$ $attr_types=[time.fromisoformat,\ str])$
(3) $RETRIEVE(query="my\ doctor's\ appointments")$
Anecdotal Example 5 – “how many products did I buy online in the last 6 months?”
(0) $QUD("how\ many\ products\ did\ I\ buy\ online\ in\ the\ last\ 6\ months?")$
(1) $SUM(l=QUD("products\ bought\ online\ in\ the\ last\ 6\ months"),\ attr_name="quantity")$
(2) $FILTER(l=QUD("products\ bought\ online\ with\ purchase\ date"),$ $filter=\lambda attr: attr["purchase_date"] >= (date.today() - relativedelta(months=6)))$
(3) $EXTRACT(l=QUD("products\ bought\ online"),\ attr_names=["purchase_date",\ "quantity"],$ $attr_types=[date.fromisoformat,\ int])$
(4) $RETRIEVE(query="I\ bought\ a\ product\ online")$

Table 7.16: Subset of ICL examples used for the GLM-based version of our QUD stage. We handcrafted 40 such examples, and retrieve 8 of them *dynamically*, per step, during inference (via BM25).

ICL Example 1 – “On which day did I listen to music the most?”
<p>User: Starting with new question. Input: QUD("On which day did I listen to music the most?") Assistant: ARCMAX(l= QUD("number of songs I listened per day?") , arg_attr_name="num_songs", val_attr_name="start_date") User: Input: QUD("number of songs I listened per day?") Assistant: MAP(l= QUD("my songs I listened to grouped by day") , fct=len, res_name="num_songs") User: Input: QUD("my songs I listened to grouped by day") Assistant: GROUP_BY(l= QUD("instances I listened to music with date") , attr_names=["start_date"]) User: Input: QUD("instances I listened to music with date") Assistant: EXTRACT(l= QUD("I listened to music") , attr_names=["start_date"], attr_types=[date.fromisoformat]) User: Input: QUD("I listened to music") Assistant: RETRIEVE(query="I listened to music")</p>
ICL Example 2 – “how often did I meet with both my parents in the evening?”
<p>User: Starting with new question. Input: QUD("how often did I meet with both my parents in the evening?") Assistant: APPLY(l= QUD("I met with both my parents in the evening") , fct=len) User: Input: QUD("I met with both my parents in the evening") Assistant: FILTER(l= QUD("instances I met with both my parents") , filter=lambda attr: attr["start_time"].hour >= 18 and attr["start_time"].hour < 24) User: Input: QUD("instances I met with both my parents") Assistant: JOIN(l1= QUD("instances I met with my mum") , l2= QUD("instances I met with my dad") , condition="i1.start_datetime <= i2.end_datetime and i2.start_datetime <= i1.end_datetime") User: Input: QUD("instances I met with my mum") Assistant: RETRIEVE(query="I met with my mum") User: Input: QUD("instances I met with my dad") Assistant: RETRIEVE(query="I met with my dad")</p>
ICL Example 3 – “how much money did I spend online the last three years?”
<p>User: Starting with new question. Input: QUD("how much money did I spend online the last three years?") Assistant: SUM(l= QUD("my online purchases in the last three years with amounts") , attr_name="amount_spent") User: Input: QUD("my online purchases in the last three years with amounts") Assistant: EXTRACT(l= QUD("my online purchases in the last three years") , attr_names=["amount_spent"], attr_types=[float]) User: Input: QUD("my online purchases in the last three years") Assistant: FILTER(l= QUD("my online purchases with date") , filter=lambda attr: attr["purchase_date"] >=(date.today() - relativedelta(years=3))) User: Input: QUD("my online purchases with date") Assistant: EXTRACT(l= QUD("my online purchases") , attr_names=["purchase_date"], attr_types=[date.fromisoformat]) User: Input: QUD("my online purchases") Assistant: RETRIEVE(query="my online purchases")</p>

8

Conclusion

Contents

8.1	Summary	133
8.2	Lessons Learned	135
8.3	Limitations	138
8.4	Outlook	139

8.1 Summary

The following summarizes the contributions of this dissertation. An overview is given in Table 8.1.

In **Chapter 3**, we introduce the 3-stage pipeline for QA over heterogeneous sources, that is built upon in subsequent Chapters 4, 5 and 6. We develop CONVINSE, the first approach for ConvQA over heterogeneous sources. CONVINSE constructs a structured representation (SR), which makes the intent of an incomplete follow-up question explicit, and captures the important cues in categorized slots. Evidence is then obtained from heterogeneous sources, by top- k entity linking (via CLOCQ [31]), and subsequent look-up of related KB-facts and retrieval of text, tables and infoboxes from the Wikipedia pages of linked entities. All evidence is verbalized for unified treatment in the answering step, which relies on FID [85]. We further construct CONVMIX, the first benchmark for ConvQA that requires tapping into a mixture of sources. Experiments demonstrate the benefits of the SR generation compared to baselines for question completion.

In **Chapter 4**, we propose a graph-based answering approach that constructs a unified evidence graph which seamlessly combines information from heterogeneous sources. This graph is then reduced in size via iterative inference of GNNs, relying on both encodings by small-scale LMs and structural cues among evidence pieces. A crucial component of the GNNs is our novel attention mechanism that gives more weight to question-relevant information during its message passing steps. The answer is then predicted from a small graph with few evidence pieces only, which serves as explanatory evidence for the answer. Our resulting end-to-end QA method, named EXPLAIGNN, demonstrates substantial answering improvements over CONVINSE, as a result of the enhanced answering stage. Notably, the GNNs in EXPLAIGNN are also decisive parts in our follow-up works FAITH (Chapter 5) and QUASAR (Chapter 6).

Table 8.1: Chapter-wise summary of the contributions made with this dissertation.

Chapter	Contributions
Chapter 3	<ul style="list-style-type: none"> Proposing CONVINSE, the first end-to-end method for conversational QA (ConvQA) that integrates heterogeneous sources for enhanced answer coverage. Introducing a 3-stage pipeline for QA over heterogeneous sources, consisting of (i) a question understanding phase, (ii) an evidence retrieval phase, and (iii) an answering step for deriving the answer from this evidence. Collecting CONVMIX, the first benchmark for ConvQA over heterogeneous sources, comprising 3,000 real-user conversations with 16,000 questions.
Chapter 4	<ul style="list-style-type: none"> Proposing EXPLAIGNN, a new method for ConvQA over heterogeneous sources with a focus on computing answers that are explainable to end users. Devising a framework for iterative application of GNN inference to shrink evidence graphs until the answer and its explanatory evidence are obtained. Developing a novel attention mechanism which ensures that only question-relevant information is spread during the message passing steps of the GNNs.
Chapter 5	<ul style="list-style-type: none"> Proposing FAITH, the first temporal QA system that taps into heterogeneous information sources, and provides faithful answers with explanatory evidence. Introducing a mechanism that transforms implicit temporal constraints into explicit conditions, by recursively calling the QA system itself. Constructing TIQ, a benchmark consisting of 10,000 implicit questions, based on an automatic pipeline for creating difficult temporal questions.
Chapter 6	<ul style="list-style-type: none"> Proposing QUASAR, a unified system architecture for answering complex questions over heterogeneous sources, with strong points regarding both effectiveness (i.e., answer quality) and efficiency (i.e., computational cost). Demonstrating the benefits of iterative GNN re-ranking on the answer presence. Releasing COMPMIX, a benchmark for QA over heterogeneous sources, with 9K questions that exhibit complex phenomena such as comparisons or aggregation.
Chapter 7	<ul style="list-style-type: none"> Proposing REQAP, a novel approach for complex QA over heterogeneous personal data, that is empowered by small-scale LMs suitable for on-device inference. Devising a recursive question understanding mechanism that can solve user questions involving complexities such as aggregations, grouping, or temporal conditions, based on step-by-step decomposition of the question complexity. Constructing PERQA, a large-scale benchmark for personal QA with realistic user data and questions, that is constructed based on handcrafted personas.

In **Chapter 5**, we devise a new approach, FAITH, for *faithfully* answering temporal questions, with focus on the challenging case of implicit time constraints. For modeling implicit temporal constraints, FAITH generates intermediate questions and answers them in recursive calls to the entire QA system, making the time constraint explicit. The temporal information need is then captured in an SR with dedicated slots for time-related cues. After evidence retrieval, explicit rules are utilized for pruning out information inconsistent with the time constraints. The answer is obtained via the GNNs of EXPLAIGNN. For evaluation, we construct the TIQ dataset with 10,000 implicit questions. Experiments show that FAITH substantially outperforms the State of the Art on questions with implicit time constraints. On other temporal questions, our method performs almost on par, but adds the benefit of reliably matching the temporal conditions.

In **Chapter 6**, we introduce QUASAR, a unified approach for complex QA that follows the same 3-stage pipeline as before. However, QUASAR employs a RAG-based answering step for enhanced answering performance: the iterative GNNs proposed in EXPLAIGNN are utilized for identifying the most relevant evidence pieces, and these are provided as input for a generative LM. The GLM enables combining information from few evidence pieces in the input, enabling answering of more advanced questions. For experiments, we release COMPMIX, a large-scale benchmark for QA over heterogeneous sources, that collates the completed questions from our CONVMIX dataset. We also compare on a subset of complex questions from the CRAG dataset

and on temporal questions in TIMEQUESTIONS. Our results show that QUASAR performs on par or outperforms baselines that incur much higher computational costs, across all benchmarks.

In **Chapter 7**, we propose REQAP, a novel approach for answering complex questions over personal data that naturally appears in heterogeneous form. REQAP targets questions involving advanced operations such as aggregation, joining or grouping with hundreds or thousands of relevant information snippets. For processing such questions, REQAP first constructs an operator tree by recursive decomposition of the information need. This operator tree is then executed, enabling users to trace back their answers to the sources. The individual operators tap into both structured sources (e.g., streaming data or workouts) and unstructured sources (e.g., mails, calendar, notes), enabled by novel operators that build upon the verbalization paradigm. Experiments on our new benchmark, PERQA, show the substantial benefits of REQAP compared to baselines based on (i) verbalization and RAG, or (ii) code generation.

8.2 Lessons Learned

Integration of heterogeneous sources is vital. The overarching goal of this dissertation was to integrate data sources of different types into QA systems for enhanced answer coverage. Throughout the chapters, we observed consistent and substantial improvements of QA performance when integrating heterogeneous sources. The corresponding experimental results are covered in:

- Table 3.5 and Table 3.6 for CONVINSE,
- Table 4.3 for EXPLAIGNN,
- Table 5.6 for FAITH,
- Table 6.3 and Table 6.7 for QUASAR, and
- Table 7.10 for REQAP.

This substantiates the key lesson learned in this dissertation: integrating heterogeneous sources improves both the answer coverage and answer accuracy (via multi-source corroboration) for QA, and thus enables major improvements of the end-to-end answering performance for QA systems.

The 3-stage pipeline excels across QA tasks. The second key takeaway is that our 3-stage pipeline introduced in Chapter 3, consisting of

- (i) a question understanding (QU),
- (ii) an evidence retrieval and scoring (ERS), and
- (iii) a heterogeneous answering (HA) stage

is applicable to a variety of QA tasks, including conversational QA, temporal QA and complex QA. The key novelty is the structured representation (SR) of the question intent, which is adaptable to different scenarios for question understanding, including incomplete follow-up questions in conversations with implicit intent and temporal questions with time constraints. The ERS stage enables integration of heterogeneous sources, by unifying retrieval results via verbalization. Our methods, built upon this pipeline, demonstrate strong performance across the board.

Intermediate results in symbolic space for traceable answers. Another key design principle in all our methods is the traceability of answers. Existing work on QA, including recent large-scale GLMs [143], often employs end-to-end black-box neural architectures for generating the answer. In contrast, all our approaches compute intermediate results in symbolic space, as well as explanatory evidence for supporting the answer. This includes:

- the SR within CONVINSE, EXPLAIGNN, FAITH and QUASAR,
- the conversational flow graph devised in CONVINSE and applicable for EXPLAIGNN,

- the evidence used for answer prediction in EXPLAIGNN, FAITH and QUASAR, and
- the operator tree and its computed event sets in REQAP.

We verified that our answer explanations are understandable by end users in EXPLAIGNN (Section 4.5) and REQAP (Section 7.8.4). Further, these intermediate results enable deriving more faithful answers for questions with specific constraints, as demonstrated for FAITH (Section 5.6.2).

Combinations of small-scale models perform on par with billion-scale GLMs. Another benefit of computing intermediate results is that the task complexity is naturally decomposed. The individual sub-tasks, that are of a simpler nature, can then be solved via smaller model architectures, leading to reduced computational costs. Example sizes of models used throughout this dissertation:

- the BART model for generating the SR has 139M parameters (used in CONVINSE, EXPLAIGNN, FAITH and QUASAR),
- the GNNs within EXPLAIGNN and FAITH (based on DISTILROBERTA) have 94M parameters,
- the GNNs within QUASAR (based on smaller ELMs) have 22M and 26M parameters,
- the cross-encoder within FAITH (based on DISTILROBERTA) has 82M parameters,
- the cross-encoders in the RETRIEVE operator of REQAP have between 4M and 33M parameters, and the Seq2seq LM in the EXTRACT operator has between 16M and 139M parameters.

This is in stark contrast to recent GLMs which would often consist of >100B parameters. Note that in some cases our models necessitate processing larger sets of inputs (e.g., the cross-encoders in FAITH and REQAP or the GNNs in EXPLAIGNN). Through batching (for cross-encoders) and matrix-based computations (for GNNs) the computational costs and memory consumption are still substantially lower than for typical large-scale GLMs.

Weak supervision and in-context learning are useful techniques. For enabling this strong performance with relatively small-scale models, the individual components are fine-tuned on task-specific data. In general, we refrain from collecting such annotations via crowdsourcing, since this is cost-prohibitive, not easily scalable, and would often require expert knowledge. Instead, we explore two different techniques in this dissertation:

- weak supervision, and
- in-context learning (ICL).

Weak supervision is used for heuristic annotation of SRs in CONVINSE, EXPLAIGNN, FAITH and QUASAR, by identifying relevant mentions in the input that yield the retrieval of answering evidence. Also, the annotation of such answering evidence (utilized within EXPLAIGNN, FAITH and QUASAR) is based on the proxy that answering evidence mentions the ground-truth answer.

With the emergence of ICL with recent GLMs [14], we investigated this paradigm for annotating data. ICL is generally more flexible than our weak supervision approaches that are mostly implemented via rule-based heuristics. We utilize ICL for annotating intermediate questions to resolve implicit time constraints (within FAITH), and for deriving candidate operator trees (within REQAP).

In the REQAP project, we also demonstrate how the two paradigms can be combined: candidate operator trees are obtained via ICL, but we only retain the ones leading to the correct ground-truth answer, which is a typical heuristic in weak supervision. Similarly, we generate ground-truth queries using ICL, but utilize heuristics for annotating relevant events, for training the RETRIEVE operator (Section 7.3.2.1). This combination is generally promising, as ICL allows for flexible annotations, while weak supervision heuristics improve the correctness and efficiency of annotations (e.g., for annotating ground-truth evidence, weak supervision could derive

candidates pruning the search space [31] for ICL).

Combining LMs with structure enhances performance. Another strength of our approaches is the judicious combination of language models and their natural language understanding capabilities, with structural cues and structured processing techniques:

- Our frame-like SR (introduced in Chapter 3 and used in Chapters 4, 5 and 6) imposes structure upon the plain textual form of the information need. This structure helps with evidence retrieval (cf. Table 4.7), LM-based answer generation (cf. Table 3.11), enables our rule-based temporal pruning (in Section 5.3.2), and assists LM-based encodings for our GNNs (in Section 4.2.3).
- Within the GNNs introduced in Chapter 4 (and utilized in Chapters 4, 5, and 6), the initial encodings for entity and evidence nodes are obtained via encoder LMs, and in the message passing steps the structural relationships between these nodes are utilized for improving the informativeness of these encodings.
- In the REQAP methodology, we use a GLM for generating an operator tree that follows Python syntax. The operator tree itself mostly consists of operators that execute code for structured processing of event sets. For enabling this behavior, the **EXTRACT** operator (Section 7.3.2.2), implemented by a fine-tuned LM, transforms unstructured information covered in plain text into structured key-value pairs.

These examples contrast the prevalent approach of treating and concatenating all information as plain token sequences, for LM-based generation. Such verbalization can be a useful technique for implementing sub-tasks in a pipeline, as in the HA stage in CONVINSE (Section 3.2.3), or for independent processing of information pieces, as in the GNN encodings of EXPLAINNN (Section 4.2.3) or the **EXTRACT** operator in REQAP (Section 7.3.2.2) – but following an end-to-end approach based on verbalization loses insightful structural cues and implies total reliance on the reasoning capabilities of LMs (e.g., for advanced operations such as aggregation or grouping).

Decomposition helps with complex information needs. Connected to earlier discussions in this section, we found that decomposition generally helps dealing with complex questions:

- This holds true for the decomposition of the end-to-end QA task into three sub-tasks within a pipeline (as employed in Chapters 3, 4, 5, and 6).
- However, we also observe that decomposition enables dealing with complex temporal questions specifying implicit time constraints (Section 5.3.1). In this case, answering the generated intermediate questions is simpler than answering the entire implicit question in one shot.
- This idea of decomposition is further extended in REQAP (Section 7.3.1), which recursively decomposes questions in multiple steps until the resulting sub-question can be answered by a simple call to the **RETRIEVE** operator.

In all cases, decomposition contributes substantially to the performance of our approaches:

- Table 6.6 compares results of our decomposed pipeline versus end-to-end answer generation by large-scale GLMs.
- Table 5.4 illustrates the performance benefits of our decomposition-based approach in FAITH versus state-of-the-art methods for temporal QA without such decomposition.
- Table 7.7 shows results of REQAP with recursive decomposition versus a variant of REQAP with one-shot generation of operator trees.

8.3 Limitations

Restriction to Wikimedia content. Our QA systems for answering real-world knowledge (Chapters 3-6) all utilize information from Wikipedia and Wikidata for grounding their answers. Integrating other (web) content, such as news pages, online forums or even local documents, could enable further improvements of the answer coverage. However, non-Wikimedia sources are typically harder to process: Wikipedia content follows a specific structure, but other content would require well-designed crawling and scraping techniques for processing the plain HTML code. Note that existing public libraries could be utilized for processing general web content, such as `BeautifulSoup`¹ or `unstructured`².

Beyond these technical challenges of dealing with web content outside Wikipedia/Wikidata, such content would also introduce more noise into the QA pipeline: information in Wikipedia is often nicely structured and polished, whereas open-web text and tables might contain typos and ungrammatical sentences. Even though we found our approaches to be relatively robust to ad-hoc formulations on the question-side, as present in our CONVMIX (Section 3.3) and COMPMIX (Section 6.3) benchmarks, it is unclear how the appearance of these phenomena on the evidence-side would affect our QA systems.

Trust in data sources. Connecting to the previous limitation, in our experiments, we assumed all sources to be trustworthy and unbiased. With our focus on Wikidata and Wikipedia, this assumption has been well justified. In the wild, however, input data for QA systems likely exhibits a wide spectrum of quality issues, in terms of stale information, biased positions, or simply false statements. Identifying trustworthy and up-to-date evidence and dealing with conflicting data, has been explored in other contexts (e.g., for KB curation [46]), but remains a major challenge for QA in general. Especially the appearance of conflicting information could be critical.

While we did not investigate this problem in the context of this dissertation, we observed few cases of minor inconsistencies between Wikipedia and Wikidata when developing the demo for the EXPLAINNN project. An example is the duration of the movie *Angels&Demons*: the Wikipedia infobox mentions a running time of “138 minutes”, while Wikidata provides a duration of “146 minutes”. For such cases, EXPLAINNN would rank the different (contradicting) information pieces as relevant, and provide them as explanatory evidence to the user (with one chosen answer). The user would then be able to detect conflicts in the evidence.

Targeting QA tasks independently. In the context of this dissertation, we target different QA tasks and their complexities in separation: Chapters 3 and 4 target conversational QA, Chapter 5 targets temporal QA, and Chapter 7 aims to answer questions over personal data. Chapter 6 targets both complex QA and temporal QA (which is one type of complex QA), but trains independent models. Supporting these different question types in a unified QA system requires further research. Simple solutions could be to (i) employ a question classifier for routing questions to task-specific QA systems, or (ii) training a joint question understanding module on the union of task-specific training data. The second solution might benefit from transfer learning, and is more general as it could implicitly route questions (e.g., targeting personal data) by generating outputs specific to the downstream QA system (e.g., an operator tree for personal data and otherwise an SR).

Long-form answers. Another limitation of this dissertation is the focus on crisp and entity-level answers. While this might be the desirable mode of interaction for most users, and could even enable integration of QA systems as tools for GLMs [184], some users might prefer additional

¹<https://pypi.org/project/beautifulsoup4/>

²<https://github.com/Unstructured-I0/unstructured>

context, and specific questions inherently require longer answers [194]. For example, information needs could be open-ended (e.g., for the question “*How did Langdon save the world?*” a single entity would not be sufficient as answer), or ambiguous (e.g., the question “*how long is Angels&Demons?*” could be about the movie or the novel), requiring passage-level answers. Retrieving relevant information for such *long-form answers* is already challenging when solely operating over text [36], and would be even harder when tapping into heterogeneous sources. Further, crisp and entity-level answers can be traced back easily, as they would be exactly mentioned in text (in Chapter 6), or directly predicted as relevant in the form of a graph node (in Chapters 4 and 5). However, long-form answers require some generative element that adapts the retrieved evidence for directly targeting the specific information need of the user – this makes it harder to understand the grounding evidence for individual parts of the answer [36].

Further modalities. Finally, this dissertation does not consider other modalities, beyond text, tables and KBs, such as audio, images or videos. Information from these modalities could further improve the answer coverage, and enable answering completely different types of questions (e.g., “*Which actors are shown on the movie cover of Angels&Demons?*”). A simple approach would be to integrate these sources by utilizing a vision language model for verbalizing the content [121, 126]. This might oversimplify the problem though, and needs further investigation.

Visual content, including images and videos, would also be useful for QA over personal data. In this context, even other types of user data, such as geo-locations, interactions with their personal devices (clicking, typing, ...), or contextual information (the current time/weekday, climate, device in-use, ...) could convey important information for dealing with user requests.

8.4 Outlook

The following presents different avenues for future research revolving around the work presented in this dissertation (beyond targeting its limitations discussed in Section 8.3).

Putting it all together. Our approach in Chapter 7 for QA over personal data differs from the pipeline pursued in Chapters 3-6. This is motivated by the question and result type targeted in Chapter 7, which often requires computing sets of relevant events, and executing advanced operations (e.g., grouping, aggregation, joining or filtering) over these sets to compute an answer. For answering a question like “*my total distance run in 2025?*”, the QA system has to retrieve *all* relevant user events – as soon as one evidence piece is missing, the answer will deviate from the ground-truth. In contrast, questions like “*the main character in Angels&Demons?*” often require combining *few* (3-5) evidence pieces (that jointly implicate that Robert Langdon is the *main* character of the book), but not hundreds or thousands. Combining these paradigms in one unified approach would be an interesting direction for future work: current LM-based operators in REQAP (RETRIEVE and EXTRACT) process each user event individually, without combining information from multiple pieces (as in EXPLAINNN or QUASAR). A potential solution could incorporate a new operator, that receives a question (and, optionally, a set of relevant events) as input, and proceeds as outlined in Chapter 4 (on EXPLAINNN) or Chapter 6 (on QUASAR). This would enable more accurate answering of questions like “*Which hotel did I stay at in Barcelona?*”, by leveraging answer redundancy among different evidence pieces (cf. Section 3.5.1). In contrast, the EXTRACT operator in REQAP could potentially be distracted by noise, such as advertisement mails or messages from friends about hotels in Barcelona.

Similarly, REQAP operates solely over personal data, without access to background knowledge (outside the processed data itself). Such knowledge could be incorporated by calls to a new EXPLAINNN or QUASAR operator, for answering questions such as “*how many kilometers did*

I run since the COVID outbreak?”. Importantly, such operators could be separated from the processing of user data and executed in the cloud, as no personal information is accessed.

Finally, investigating a REQAP-like system for answering analytical questions over non-personal data could be an interesting extension. Queries like “*number of non-penalty career goals by Messi?*” or “*CO2 emissions in Europe in 2024?*”, require integration of many sources of evidence for providing an accurate answer. Connected to the previous discussions, such questions require combining information from multiple evidence pieces (e.g., information for each individual European country). Further, numerical values from multiple sources for similar statistics might need consolidation [64], and often conversion and normalization of units is required [78].

Seamless integration of tools. Future research could also aim to integrate a larger set of tools [184], moving towards AI systems that are actual personal assistants, capable of performing non-trivial actions and look-ups for end users, that require knowledge of their context and personal information. Consider the question:

q: what’s the next bus to my work place?

While this question looks simple on surface, answering the question requires:

- access to the user’s current location,
- knowledge of the user’s work place address, and
- access to an up-to-date local bus schedule.

These individual steps could be empowered by calling specific tools. For the example, a potential solution, generated by a dedicated GLM, could look like:

```
BUS (
  start = LOCATION(“user’s current location”),
  destination = REQAP(“user’s work place”)
)
```

where `LOCATION(“user’s current location”)` accesses the current location of the user (i.e., the location of the user device in use), `REQAP(“user’s work place”)` utilizes our REQAP approach to obtain the work place address, and `BUS` makes an API call to obtain live information. This plan could be derived similar to operator trees in REQAP (by decomposition of the question).

Note that the security of such systems would be of utmost importance [67] – imagine a system that has access to a banking tool. It is absolutely necessary that no external factor (such as an incoming mail) can trigger an undesired money transaction.

Towards proactive systems. Another interesting avenue for future research is the proactivity of (QA) systems. After asking several questions revolving around Dan Brown novels, the user might ask for similar books from other authors. The system would then be supposed to provide recommendations [12], ideally based on cues from the ongoing conversation [208]. A smart system could also automatically detect that the user could be interested in other books, based on *implicit* cues, and *proactively* provide recommendations. Similarly, in the context of personal data, the system could detect user patterns (e.g., that the user typically eats Italian food after playing football), and automatically suggest to perform specific actions (e.g., suggesting a reservation at the user’s favorite Italian restaurant, when observing a calendar entry about playing football).

In these scenarios, it would be critical to avoid overwhelming the end user, by providing too many of these proactive suggestions. The system should comply with the user’s preferences, which requires judicious personalization.

Self-assessment. Finally, a smart QA system should be able to identify incorrect answers automatically, and correct them in an autonomous manner.

In the context of Chapters 3-6, this would involve identifying SRs that do not capture the information need well, lead to retrieval of irrelevant evidence, or lead to incorrect answers. Upon detecting such an instance, the QA system could generate an alternative SR, and verify whether the resulting answer is faithful to the information need. Similarly, the evidence retrieval or answer derivation could be enhanced.

In the context of Chapter 7, such self-assessment could be conducted on the operator-level. After execution of an operator, a dedicated model could scrutinize its result (e.g., the resulting set of user events), and decide whether this result is as expected. For example, a RETRIEVE operator call could have an incorrect query (like *"I worked"*), that was supposed to identify user workouts. After self-assessment, such errors could then be automatically corrected. In the example, the RETRIEVE query would be changed to *"I worked out"*.

How to devise such mechanisms for self-assessment is subject for future research.

List of Resources

- **Chapter 3**

- The CONVINSE Methodology – Code: <https://convinse.mpi-inf.mpg.de>
- The CONVINSE Methodology – Demo: <https://convinse.mpi-inf.mpg.de>
- The CONVMIX Benchmark – Data: <https://convinse.mpi-inf.mpg.de>

- **Chapter 4**

- The EXPLAIGNN Methodology – Code: <https://explaignn.mpi-inf.mpg.de>
- The EXPLAIGNN Methodology – Demo: <https://explaignn.mpi-inf.mpg.de>

- **Chapter 5**

- The FAITH Methodology – Code: <https://faith.mpi-inf.mpg.de>
- The TIQ Benchmark – Data: <https://faith.mpi-inf.mpg.de>

- **Chapter 6**

- The QUASAR Methodology – Data: <https://qa.mpi-inf.mpg.de/quasar>
- The COMPMIX Benchmark – Data: <https://qa.mpi-inf.mpg.de/compmix>

- **Chapter 7**

- The REQAP Methodology – Code: <https://reqap.mpi-inf.mpg.de>
- The REQAP Methodology – Demo: <https://reqap.mpi-inf.mpg.de/demo>
- The PERQA Benchmark – Data: <https://reqap.mpi-inf.mpg.de>

List of Figures

3.1	CONVINSE: Overview of the CONVINSE methodology	22
3.2	CONVINSE: Example conversational flow graph	24
3.3	CONVINSE: Evidence retrieval from heterogeneous sources	24
3.4	CONVINSE: Screenshot of the CONVINSE demo	37
4.1	EXPLAIGNN: Overview of the GNN-based answering stage of EXPLAIGNN	41
4.2	EXPLAIGNN: Example evidence graph and its iterative reduction	43
4.3	EXPLAIGNN: Training of and inference with iterative GNNs	46
4.4	EXPLAIGNN: Screenshot of the EXPLAIGNN demo	59
5.1	FAITH: Overview of the FAITH methodology	65
5.2	FAITH: Resolving of implicit temporal constraints	66
5.3	FAITH: Construction of the TIQ benchmark	70
5.4	FAITH: Input-source combinations in the TIQ benchmark	72
5.5	FAITH: P@1 with top- k answers for intermediate questions	80
6.1	QUASAR: Overview of the QUASAR methodology	88
6.2	QUASAR: Answer-type frequencies in the COMPMIX benchmark	92
6.3	QUASAR: P@1 by the number of evidence pieces	98
7.1	REQAP: Example user data of heterogeneous form	106
7.2	REQAP: Overview of an example inference of the REQAP methodology	107
7.3	REQAP: GLM instructions for generating operator trees	109
7.4	REQAP: Example canonicalized and verbalized user events in PERQA	114
7.5	REQAP: GLM instructions for generating complex questions	115
7.6	REQAP: Example user events in our REQAP demo	126
7.7	REQAP: Screenshot of the REQAP demo	127

List of Tables

1.1	Overview of this dissertation’s contributions.	5
2.1	Example verbalized evidence from different input sources	17
3.1	CONVINSE: Comparison of question understanding approaches for ConvQA	21
3.2	CONVINSE: Example verbalized evidence from different input sources	25
3.3	CONVINSE: Basic statistics for the CONVMIX benchmark	27
3.4	CONVINSE: Representative conversations from the CONVMIX benchmark	28
3.5	CONVINSE: Answer presence on CONVMIX after the ERS stage	32
3.6	CONVINSE: Main results on the CONVMIX benchmark	32
3.7	CONVINSE: P@1 with predicted answers for previous turns	33
3.8	CONVINSE: P@1 over turns	34
3.9	CONVINSE: Average QU output length	34
3.10	CONVINSE: P@1 by domain and information source	35
3.11	CONVINSE: Ablation studies	35
3.12	CONVINSE: Anecdotal examples for generated SRs	36
4.1	EXPLAIGNN: Main results on the CONVMIX benchmark	49
4.2	EXPLAIGNN: Answering performance with predicted answers in the history	50
4.3	EXPLAIGNN: Effect of integrating heterogeneous sources	50
4.4	EXPLAIGNN: Effect of varying the multi-task learning weights	51
4.5	EXPLAIGNN: Effect of varying the GNN iterations	51
4.6	EXPLAIGNN: Robustness test on the CONVQUESTIONS benchmark	52
4.7	EXPLAIGNN: Ablation studies	52
4.8	EXPLAIGNN: Effect of varying the evidence graph size	53
4.9	EXPLAIGNN: Effect of varying the CLOCQ- k parameter	54
4.10	EXPLAIGNN: Effect of varying the maximum number of positive evidence pieces .	54
4.11	EXPLAIGNN: Effect of varying the encoder LM	55
4.12	EXPLAIGNN: Confusion matrix for the user study	55
4.13	EXPLAIGNN: User study examples with correct answers by EXPLAIGNN	56
4.14	EXPLAIGNN: User study examples with incorrect answers by EXPLAIGNN	57
5.1	FAITH: Comparison of benchmarks for temporal QA	69
5.2	FAITH: Basic statistics for the TIQ benchmark	72
5.3	FAITH: Representative questions from the TIQ benchmark	74
5.4	FAITH: Main results on the TIQ and TIMEQUESTIONS benchmarks	75
5.5	FAITH: Comparison of the faithfulness of FAITH and Un-FAITH	76

List of Tables

5.6	FAITH: Effect of integrating heterogeneous sources	78
5.7	FAITH: Ablation studies	78
5.8	FAITH: Anecdotal example for derived SRs on the TIQ benchmark	79
5.9	FAITH: Anecdotal example for derived SRs on the TIMEQUESTIONS benchmark	80
5.10	FAITH: Performance on questions from different source combinations	81
5.11	FAITH: Prompt for rephrasing pseudo-questions	83
5.12	FAITH: Prompts for generating intermediate questions	84
6.1	QUASAR: Comparison of benchmarks for QA over heterogeneous sources	91
6.2	QUASAR: Basic statistics for the COMPMIX benchmark	91
6.3	QUASAR: Answer coverage across information sources	93
6.4	QUASAR: Answer redundancy across information sources	94
6.5	QUASAR: Representative questions from the COMPMIX benchmark	95
6.6	QUASAR: Main results on COMPMIX, CRAG and TIMEQUESTIONS	97
6.7	QUASAR: Effect of integrating heterogeneous sources	97
6.8	QUASAR: Effect of unified versus per-source evidence scoring	98
6.9	QUASAR: Effect of varying the ES strategy	99
6.10	QUASAR: Anecdotal examples for generated SRs	99
6.11	QUASAR: Performance of a selective variant on COMPMIX	100
6.12	QUASAR: Performance of a selective variant on TIMEQUESTIONS	100
7.1	REQAP: Operators with their input types	109
7.2	REQAP: Basic statistics for the PERQA dataset	116
7.3	REQAP: Representative questions in the PERQA benchmark	117
7.4	REQAP: Representative user data in the PERQA benchmark	118
7.5	REQAP: Main results on the PERQA benchmark	119
7.6	REQAP: Performance by complexity types of questions	120
7.7	REQAP: Ablation studies	120
7.8	REQAP: Base models used for the model-size analysis	121
7.9	REQAP: Effect of varying the model sizes of individual modules	121
7.10	REQAP: Effect of integrating heterogeneous sources	121
7.11	REQAP: Results for test personas 1-3 in the PERQA benchmark	122
7.12	REQAP: Results for test personas 4-6 in the PERQA benchmark	122
7.13	REQAP: Average operator run-times	123
7.14	REQAP: Results on the TIMELINEQA benchmark	123
7.15	REQAP: Anecdotal examples for generated operator trees	130
7.16	REQAP: Subset of ICL examples for the QUD stage	131
8.1	Chapter-wise summary of this dissertation’s contributions	134

Bibliography

- [1] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. [Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases](#). In *WWW*, 2018.
- [2] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. [Automated Template Generation for Question Answering over Knowledge Graphs](#). In *WWW*, 2017.
- [3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. [DBpedia: A Nucleus for a Web of Open Data](#). In *ISWC*, 2007.
- [4] Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. [Transformers for Tabular Data Representation: A Survey of Models and Applications](#). *Transactions of the Association for Computational Linguistics*, 2023.
- [5] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. [SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval](#). *arXiv*, 2020.
- [6] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. [MS MARCO: A Human-Generated Machine Reading Comprehension Dataset](#). *arXiv*, 2016.
- [7] Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. [Modern Baselines for SPARQL Semantic Parsing](#). In *SIGIR*, 2022.
- [8] Hannah Bast and Elmar Haussmann. [More Accurate Question Answering on Freebase](#). In *CIKM*, 2015.
- [9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. [A Neural Probabilistic Language Model](#). *Journal of Machine Learning Research*, 2003.
- [10] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. [Semantic Parsing on Freebase from Question-Answer Pairs](#). In *EMNLP*, 2013.
- [11] Nikita Bhutani, Xinyi Zheng, and H. V. Jagadish. [Learning to Answer Complex Questions over Knowledge Bases with Query Composition](#). In *CIKM*, 2019.
- [12] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. [Recommender Systems Survey](#). *Knowledge-Based Systems*, 2013.

- [13] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. [Large-scale Simple Question Answering with Memory Networks](#). *arXiv*, 2015.
- [14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. [Language Models are Few-Shot Learners](#). In *NeurIPS*, 2020.
- [15] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. [WebTables: Exploring the Power of Tables on the Web](#). In *VLDB*, 2008.
- [16] Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. [Temporal Knowledge Graph Completion: A Survey](#). In *IJCAI*, 2023.
- [17] Qingqing Cai and Alexander Yates. [Large-scale Semantic Parsing via Schema Matching and Lexicon Extension](#). In *ACL*, 2013.
- [18] Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. [SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL](#). In *NeurIPS*, 2021.
- [19] Ricardo Campos, Gaël Dias, Alípio M Jorge, and Adam Jatowt. [Survey of Temporal Information Retrieval and Related Applications](#). *ACM Computing Surveys*, 2014.
- [20] Kaushik Chakrabarti, Zhimin Chen, Siamak Shakeri, and Guihong Cao. [Open Domain Question Answering Using Web Tables](#). *arXiv*, 2020.
- [21] Angel X. Chang and Christopher D. Manning. [SUTime: A Library for Recognizing and Normalizing Time Expressions](#). In *LREC*, 2012.
- [22] Shubham Chatterjee and Laura Dietz. [BERT-ER: Query-specific BERT Entity Representations for Entity Ranking](#). In *SIGIR*, 2022.
- [23] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. [Reading Wikipedia to Answer Open-Domain Questions](#). In *ACL*, 2017.
- [24] Wenhui Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. [Open Question Answering over Tables and Text](#). In *ICLR*, 2021.
- [25] Wenhui Chen, Xinyi Wang, and William Yang Wang. [A Dataset for Answering Time-Sensitive Questions](#). In *NeurIPS*, 2021.
- [26] Wenhui Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. [HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data](#). In *EMNLP*, 2020.
- [27] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. [GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension](#). In *IJCAI*, 2020.
- [28] Zhiyu Chen, Wenhui Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. [FinQA: A Dataset of Numerical Reasoning over Financial Data](#). In *EMNLP*, 2021.
- [29] Ziyang Chen, Xiang Zhao, Jinzhi Liao, Xinyi Li, and Evangelos Kanoulas. [Temporal Knowledge Graph Question Answering via Subgraph Reasoning](#). *Knowledge-Based Systems*, 2022.

-
- [30] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. [QuAC: Question Answering in Context](#). In *EMNLP*, 2018.
- [31] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. [Beyond NED: Fast and Effective Search Space Reduction for Complex Question Answering over Knowledge Bases](#). In *WSDM*, 2022.
- [32] Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. [Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion](#). In *CIKM*, 2019.
- [33] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. [Conversational Question Answering on Heterogeneous Sources](#). In *SIGIR*, 2022.
- [34] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. [Explainable Conversational Question Answering over Heterogeneous Sources via Iterative Graph Neural Networks](#). In *SIGIR*, 2023.
- [35] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. [CompMix: A Benchmark for Heterogeneous Question Answering](#). In *WWW*, 2024.
- [36] Philipp Christmann, Svitlana Vakulenko, Ionut Sorodoc, Bill Byrne, and Adrià de Gispert. [Retrieving Contextual Information for Long-Form Question Answering using Weak Supervision](#). In *EMNLP*, 2024.
- [37] Philipp Christmann and Gerhard Weikum. [RAG-based Question Answering over Heterogeneous Data and Text](#). *IEEE Data Engineering Bulletin*, 2024.
- [38] Philipp Christmann and Gerhard Weikum. [Recursive Question Understanding for Complex Question Answering over Heterogeneous Personal Data](#). In *ACL*, 2025.
- [39] Philipp Christmann and Gerhard Weikum. [The ReQAP System for Question Answering over Personal Information](#). In *CIKM*, 2025.
- [40] Hervé Déjean, Stéphane Clinchant, and Thibault Formal. [A Thorough Comparison of Cross-Encoders and LLMs for Reranking SPLADE](#). *arXiv*, 2024.
- [41] Yang Deng, Wenqiang Lei, Wenxuan Zhang, Wai Lam, and Tat-Seng Chua. [PACIFIC: Towards Proactive Conversational Question Answering over Tabular and Textual Data in Finance](#). In *EMNLP*, 2022.
- [42] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). In *NAACL*, 2019.
- [43] Bhuvan Dhingra, Jeremy Cole, Julian Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William Cohen. [Time-Aware Language Models as Temporal Knowledge Bases](#). *Transactions of the Association for Computational Linguistics*, 2022.
- [44] Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. [Towards a Question Answering System over the Semantic Web](#). *Semantic Web Journal*, 2020.
- [45] Wentao Ding, Hao Chen, Huayu Li, and Yuzhong Qu. [Semantic Framework based Query Generation for Temporal Question Answering over Knowledge Graphs](#). In *EMNLP*, 2022.

- [46] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. [Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources](#). In *VLDB*, 2015.
- [47] Xin Luna Dong and Alon Y. Halevy. [A Platform for Personal Information Management and Integration](#). In *CIDR*, 2005.
- [48] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. [LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia](#). In *ISWC*, 2019.
- [49] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. [Web Question Answering: Is More Always Better?](#) In *SIGIR*, 2002.
- [50] Susan T. Dumais, Edward Cutrell, Jonathan J. Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. [Stuff I've Seen: A System for Personal Information Retrieval and Re-Use](#). In *SIGIR*, 2003.
- [51] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. [Faith and Fate: Limits of Transformers on Compositionality](#). *NeurIPS*, 2023.
- [52] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. [Can You Unpack That? Learning to Rewrite Questions-in-Context](#). In *EMNLP-IJCNLP*, 2019.
- [53] Ju Fan, Zihui Gu, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Samuel Madden, Xiaoyong Du, and Nan Tang. [Combining Small Language Models and Large Language Models for Zero-Shot NL2SQL](#). In *VLDB*, 2024.
- [54] Yue Feng, Zhen Han, Mingming Sun, and Ping Li. [Multi-Hop Open-Domain Question Answering over Structured and Unstructured Knowledge](#). In *NAACL*, 2022.
- [55] Paolo Ferragina and Ugo Scaiella. [TAGME: On-the-fly Annotation of Short Text Fragments \(by Wikipedia Entities\)](#). In *CIKM*, 2010.
- [56] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. [Building Watson: An Overview of the DeepQA Project](#). *AI Magazine*, 2010.
- [57] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. [From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective](#). In *SIGIR*, 2022.
- [58] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. [SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking](#). In *SIGIR*, 2021.
- [59] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. [Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation](#). In *VLDB*, 2024.
- [60] Shen Gao, Yuchi Zhang, Yongliang Wang, Yang Dong, Xiuying Chen, Dongyan Zhao, and Rui Yan. [HeteroQA: Learning towards Question-and-Answering through Multiple Information Sources via Heterogeneous Graph Modeling](#). In *WSDM*, 2022.

-
- [61] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. [Enabling Large Language Models to Generate Text with Citations](#). In *EMNLP*, 2023.
- [62] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. [Retrieval-Augmented Generation for Large Language Models: A Survey](#). *arXiv*, 2023.
- [63] Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. [Learning Sequence Encoders for Temporal Knowledge Graph Completion](#). In *EMNLP*, 2018.
- [64] Shrestha Ghosh, Simon Razniewski, and Gerhard Weikum. [Answering Count Questions with Structured Answers from Text](#). *Journal of Web Semantics*, 2023.
- [65] Zheng Gong and Ying Sun. [Graph Reasoning Enhanced Language Models for Text-to-SQL](#). In *SIGIR*, 2024.
- [66] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. [Baseball: An Automatic Question-Answerer](#). In *IRE-AIEE-ACM (Western)*, 1961.
- [67] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. [Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection](#). In *AISec@CCS*, 2023.
- [68] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. [Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base](#). In *NeurIPS*, 2018.
- [69] Manish Gupta and Michael Bendersky. [Information Retrieval with Verbose Queries](#). In *SIGIR*, 2015.
- [70] Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. [TempTabQA: Temporal Question Answering for Semi-Structured Tables](#). In *EMNLP*, 2023.
- [71] Cathal Gurrin, Björn Þór Jónsson, Duc-Tien Dang-Nguyen, Graham Healy, Jakub Lokoc, Liting Zhou, Luca Rossetto, Minh-Triet Tran, Wolfgang Hürst, Werner Bailer, and Klaus Schoeffmann. [Introduction to the Sixth Annual Lifelog Search Challenge, LSC’23](#). In *ICMR*, 2023.
- [72] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. [REALM: Retrieval Augmented Language Model Pre-Training](#). In *ICML*, 2020.
- [73] Alon Halevy, Yuliang Li, and Wang-Chiew Tan. [Personal Manifold: Management of Personal Data in the Age of Large Language Models](#). In *ICDE*, 2024.
- [74] Darryl Hannan, Akshay Jain, and Mohit Bansal. [ManyModalQA: Modality Disambiguation and QA over Diverse Inputs](#). In *AAAI*, 2020.
- [75] Xuehai He and Xin Wang. [Multimodal Graph Transformer for Multimodal Question Answering](#). In *EACL*, 2023.
- [76] Jonathan Herzig, Thomas Mueller, Syrine Krichene, and Julian Eisenschlos. [Open Domain Question Answering over Tables via Dense Retrieval](#). In *NAACL*, 2021.
- [77] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. [TaPas: Weakly Supervised Table Parsing via Pre-training](#). In *ACL*, 2020.

- [78] Vinh Thinh Ho, Yusra Ibrahim, Koninika Pal, Klaus Berberich, and Gerhard Weikum. [Qsearch: Answering Quantity Queries from Text](#). In *ISWC*, 2019.
- [79] Johannes Hoffart, Mohamed Amir Yosef, Iliaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. [Robust Disambiguation of Named Entities in Text](#). In *EMNLP*, 2011.
- [80] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. [The Curious Case of Neural Text Degeneration](#). In *ICLR*, 2020.
- [81] Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. [Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes](#). In *ACL*, 2023.
- [82] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. [FlowQA: Grasping Flow in History for Conversational Machine Comprehension](#). In *ICLR*, 2018.
- [83] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. [Search-based Neural Structured Learning for Sequential Question Answering](#). In *ACL*, 2017.
- [84] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. [Unsupervised Dense Information Retrieval with Contrastive Learning](#). *Transactions on Machine Learning Research*, 2022.
- [85] Gautier Izacard and Édouard Grave. [Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering](#). In *EACL*, 2021.
- [86] Parag Jain and Mirella Lapata. [Integrating Large Language Models with Graph-based Reasoning for Conversational Question Answering](#). *arXiv*, 2024.
- [87] Sarthak Jain and Byron C Wallace. [Attention is not Explanation](#). In *NAACL-HLT*, 2019.
- [88] Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. [Tables as Semi-structured Knowledge for Question Answering](#). In *ACL*, 2016.
- [89] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. [TempQuestions: A Benchmark for Temporal Question Answering](#). In *HQA@WWW*, 2018.
- [90] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. [TEQUILA: Temporal Question Answering over Knowledge Bases](#). In *CIKM*, 2018.
- [91] Zhen Jia, Philipp Christmann, and Gerhard Weikum. [Faithful Temporal Question Answering over Heterogeneous Sources](#). In *WWW*, 2024.
- [92] Zhen Jia, Philipp Christmann, and Gerhard Weikum. [TIQ: A Benchmark for Temporal Question Answering with Implicit Time Constraints](#). In *TempWeb@WWW*, 2024.
- [93] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. [Complex Temporal Question Answering on Knowledge Graphs](#). In *CIKM*, 2021.
- [94] Songlin Jiao, Zhenfang Zhu, Wenqing Wu, Zicheng Zuo, Jiangtao Qi, Wenling Wang, Guangyuan Zhang, and Peiyu Liu. [An Improving Reasoning Network for Complex Question Answering over Temporal Knowledge Graphs](#). In *Applied Intelligence*, 2022.

-
- [95] Jeff Johnson, Matthijs Douze, and Hervé Jégou. [Billion-Scale Similarity Search with GPUs](#). *IEEE Transactions on Big Data*, 2019.
- [96] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. [TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension](#). In *ACL*, 2017.
- [97] Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. [Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks](#). In *EACL*, 2021.
- [98] Endri Kacupaj, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. [Contrastive Representation Learning for Conversational Question Answering over Knowledge Graphs](#). In *CIKM*, 2022.
- [99] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. [Conversational Question Answering over Passages by Leveraging Word Proximity Networks](#). In *SIGIR*, 2020.
- [100] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. [Reinforcement Learning from Reformulations in Conversational Question Answering over Knowledge Graphs](#). In *SIGIR*, 2021.
- [101] Magdalena Kaiser and Gerhard Weikum. [Preference-based Learning with Retrieval Augmented Generation for Conversational Question Answering](#). In *WWW*, 2025.
- [102] Varvara Kalokyri, Alexander Borgida, and Amélie Marian. [YourDigitalSelf: A Personal Digital Trace Integration Tool](#). In *CIKM*, 2018.
- [103] Ehsan Kamaloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. [Evaluating Open-Domain Question Answering in the Era of Large Language Models](#). In *ACL*, 2023.
- [104] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. [Large Language Models Struggle to Learn Long-Tail Knowledge](#). In *ICML*, 2023.
- [105] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. [Dense Passage Retrieval for Open-Domain Question Answering](#). In *EMNLP*, 2020.
- [106] George Katsogiannis-Meimarakis and Georgia Koutrika. [A Survey on Deep Learning Approaches for Text-to-SQL](#). *The VLDB Journal*, 2023.
- [107] Xirui Ke, Jing Zhang, Xin Lv, Yiqi Xu, Shulin Cao, Cuiping Li, Hong Chen, and Juanzi Li. [Knowledge-augmented Self-training of A Question Rewriter for Conversational Knowledge Base Question Answering](#). In *EMNLP*, 2022.
- [108] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. [Question Answering via Integer Programming over Semi-Structured Knowledge](#). In *IJCAI*, 2016.
- [109] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang-Dong Yoo. [Gaining Extra Supervision via Multi-task learning for Multi-Modal Video Question Answering](#). In *IJCNN*, 2019.
- [110] Thomas N Kipf and Max Welling. [Semi-Supervised Classification with Graph Convolutional Networks](#). *ICLR*, 2017.

- [111] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, and Kenton Lee. [Natural Questions: A Benchmark for Question Answering Research](#). *Transactions of the Association for Computational Linguistics*, 2019.
- [112] Yunshi Lan and Jing Jiang. [Modeling Transitions of Focal Entities for Conversational Knowledge Base Question Answering](#). In *ACL*, 2021.
- [113] Julien Leblay and Melisachew Wudage Chekol. [Deriving Validity Time in Knowledge Graph](#). In *TempWeb@WWW*, 2018.
- [114] Dongkyu Lee, Chandana Satya Prakash, Jack FitzGerald, and Jens Lehmann. [MATTER: Memory-Augmented Transformer Using Heterogeneous Knowledge Sources](#). In *ACL*, 2024.
- [115] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. [BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension](#). In *ACL*, 2020.
- [116] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks](#). In *NeurIPS*, 2020.
- [117] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. [Efficient One-Pass End-to-End Entity Linking for Questions](#). In *EMNLP*, 2020.
- [118] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. [The Dawn of Natural Language to SQL: Are We Fully Ready?](#) In *VLDB*, 2024.
- [119] Meiwen Li, Tianyu Cai, Lingyan Wu, Li Chen, and Shenggen Ju. [Integrating Syntax Tree and Graph Neural Network for Conversational Question Answering over Heterogeneous Sources](#). In *NLPCC*, 2024.
- [120] Xiao Li, Yawei Sun, and Gong Cheng. [TSQA: Tabular Scenario Based Question Answering](#). In *AAAI*, 2021.
- [121] Qi Zhi Lim, Chin Poo Lee, Kian Ming Lim, and Ahmad Kamsani Samingan. [UniRaG: Unification, Retrieval, and Generation for Multimodal Question Answering With Pre-Trained Language Models](#). *IEEE Access*, 2024.
- [122] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. [Pretrained Transformers for Text Ranking: BERT and Beyond](#). *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers, 2021.
- [123] Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, D’Autume Cyprien De Masson, Tim Scholtes, Manzil Zaheer, Susannah Young, et al. [StreamingQA: A Benchmark for Adaptation to New Knowledge over Time in Question Answering Models](#). In *ICML*, 2022.
- [124] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. [TAPEX: Table Pre-training via Learning a Neural SQL Executor](#). In *ICLR*, 2022.
- [125] Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina Semnani, Chen Yu, and Monica Lam. [SUQL: Conversational Search over Structured and Unstructured Data with Large Language Models](#). In *NAACL*, 2024.

-
- [126] Weihao Liu, Fangyu Lei, Tongxu Luo, Jiahe Lei, Shizhu He, Jun Zhao, and Kang Liu. [MMHQA-ICL: Multimodal In-context Learning for Hybrid Question Answering over Text, Tables and Images](#). *arXiv*, 2023.
- [127] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. [A Survey of NL2SQL with Large Language Models: Where are we, and where are we going?](#) *arXiv*, 2024.
- [128] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#). In *ICLR*, 2020.
- [129] Shaonan Long, Jinzhi Liao, Shiyu Yang, Xiang Zhao, and Xuemin Lin. [Complex Question Answering Over Temporal Knowledge Graphs](#). In *WISE*, 2022.
- [130] Jack Longwell, Mahdiyar Ali Akbar Alavi, Fattane Zarrinkalam, and Faezeh Ensan. [Triple Augmented Generative Language Models for SPARQL Query Generation from Natural Language Questions](#). In *SIGIR-AP*, 2024.
- [131] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. [Evaluating Question Answering over Linked Data](#). *Journal of Web Semantics*, 2013.
- [132] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. [Answering Complex Questions by Joining Multi-Document Evidence with Quasi Knowledge Graphs](#). In *SIGIR*, 2019.
- [133] Haohao Luo, Ying Shen, and Yang Deng. [Unifying Text, Tables, and Images for Multimodal Question Answering](#). In *EMNLP*, 2023.
- [134] Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. [Open Domain Question Answering with A Unified Knowledge Interface](#). In *ACL*, 2022.
- [135] Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. [Open-domain Question Answering via Chain of Reasoning over Heterogeneous Knowledge](#). In *EMNLP*, 2022.
- [136] Essam Mansour, Andrei Vlad Samba, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. [A Demonstration of the Solid Platform for Social Web Applications](#). In *WWW*, 2016.
- [137] Pierre Marion, Paweł Krzysztof Nowak, and Francesco Piccinno. [Structured Context and High-Coverage Grammar for Conversational Question Answering over Knowledge Graphs](#). In *EMNLP*, 2021.
- [138] Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N Ioannidis, Adesoji Adeshina, Phillip R Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. [TempoQR: Temporal Question Reasoning over Knowledge Graphs](#). In *AAAI*, 2022.
- [139] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. [On Faithfulness and Factuality in Abstractive Summarization](#). In *ACL*, 2020.
- [140] Johannes Messner, Ralph Abboud, and Ismail Ilkan Ceylan. [Temporal Knowledge Graph Completion Using Box Embeddings](#). In *AAAI*, 2022.

- [141] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. [Key-Value Memory Networks for Directly Reading Documents](#). In *EMNLP*, 2016.
- [142] Dehai Min, Zhiyang Xu, Guilin Qi, Lifu Huang, and Chenyu You. [UniHGKR: Unified Instruction-aware Heterogeneous Knowledge Retrievers](#). In *NAACL*, 2025.
- [143] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. [Large Language Models: A Survey](#). *arXiv*, 2024.
- [144] Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. [Answering Conversational Questions on Structured Data without Logical Forms](#). In *EMNLP-IJCNLP*, 2019.
- [145] Kai Nakamura, Sharon Levy, Yi-Lin Tuan, Wenhua Chen, and William Yang Wang. [HybridDialogue: An Information-Seeking Dialogue Dataset Grounded on Tabular and Textual Data](#). In *ACL*, 2022.
- [146] Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Iqbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Penedu, et al. [SYGMA: A System for Generalizable Modular Question Answering over Knowledge Bases](#). In *EMNLP*, 2022.
- [147] Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Iqbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Penedu, et al. [A Benchmark for Generalizable and Interpretable Temporal Question Answering over Knowledge Bases](#). *arXiv*, 2022.
- [148] Jianmo Ni, Jiacheng Li, and Julian McAuley. [Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects](#). In *EMNLP-IJCNLP*, 2019.
- [149] Jianmo Ni, Larry Muhlstain, and Julian McAuley. [Modeling Heart Rate and Activity Data for Personalized Fitness Recommendation](#). In *WWW*, 2019.
- [150] Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. [TORQUE: A Reading Comprehension Dataset of Temporal Ordering Questions](#). In *EMNLP*, 2020.
- [151] Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. [Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction](#). In *ACL*, 2019.
- [152] Rodrigo Nogueira and Kyunghyun Cho. [Passage Re-ranking with BERT](#). *arXiv*, 2019.
- [153] OpenAI. [GPT-4 Technical Report](#). *arXiv*, 2023.
- [154] Barlas Oğuz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. [UniK-QA: Unified Representations of Structured and Unstructured Knowledge for Open-Domain Question Answering](#). In *NAACL*, 2022.
- [155] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. [Training Language Models to Follow Instructions with Human Feedback](#). In *NeurIPS*, 2022.

-
- [156] Namyong Park, Fuchen Liu, Purvanshi Mehta, Dana Cristofor, Christos Faloutsos, and Yuxiao Dong. [EvoKG: Jointly Modeling Event Time and Network Structure for Reasoning over Temporal Knowledge Graphs](#). In *WSDM*, 2022.
- [157] Panupong Pasupat and Percy Liang. [Compositional Semantic Parsing on Semi-Structured Tables](#). In *ACL*, 2015.
- [158] Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. [Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs](#). In *ESWC*, 2021.
- [159] Mohammadreza Pourreza and Davood Rafei. [DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction](#). In *NeurIPS*, 2023.
- [160] Soumajit Pramanik, Jesujoba Alabi, Rishiraj Saha Roy, and Gerhard Weikum. [UNIQRN: Unified Question Answering over RDF Knowledge Graphs and Natural Language Text](#). *Journal of Web Semantics*, 2024.
- [161] Minghui Qiu, Xinjing Huang, Cen Chen, Feng Ji, Chen Qu, Wei Wei, Jun Huang, and Yin Zhang. [Reinforced History Backtracking for Conversational Question Answering](#). In *AAAI*, 2021.
- [162] Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W Bruce Croft, and Mohit Iyyer. [Open-Retrieval Conversational Question Answering](#). In *SIGIR*, 2020.
- [163] Chen Qu, Liu Yang, Minghui Qiu, W Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. [BERT with History Answer Embedding for Conversational Question Answering](#). In *SIGIR*, 2019.
- [164] Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. [Attentive History Selection for Conversational Question Answering](#). In *CIKM*, 2019.
- [165] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *Journal of Machine Learning Research*, 2020.
- [166] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#). In *EMNLP*, 2016.
- [167] Gonçalo Raposo, Rui Ribeiro, Bruno Martins, and Luísa Coheur. [Question Rewriting? Assessing Its Importance for Conversational Question Answering](#). In *ECIR*, 2022.
- [168] Deepak Ravichandran and Eduard Hovy. [Learning Surface Text Patterns for a Question Answering System](#). In *ACL*, 2002.
- [169] Siva Reddy, Danqi Chen, and Christopher Manning. [CoQA: A Conversational Question Answering Challenge](#). *Transactions of the Association for Computational Linguistics*, 2019.
- [170] Nils Reimers and Iryna Gurevych. [Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks](#). In *EMNLP-IJCNLP*, 2019.
- [171] Ricardo Ribeiro, Alina Trifan, and António JR Neves. [Lifelog Retrieval From Daily Digital Data: Narrative Review](#). *JMIR mHealth and uHealth*, 2022.

- [172] Stephen Robertson and Hugo Zaragoza. [The Probabilistic Relevance Framework: BM25 and Beyond](#). *Foundations and Trends in Information Retrieval*, 2009.
- [173] Anna Rogers, Matt Gardner, and Isabelle Augenstein. [QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension](#). *ACM Computing Surveys*, 2023.
- [174] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, et al. [Recipes for Building an Open-Domain Chatbot](#). In *EACL*, 2021.
- [175] Rishiraj Saha Roy and Avishek Anand. [Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections](#). Springer, 2022.
- [176] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. [Querying Large Language Models with SQL](#). In *EDBT*, 2024.
- [177] Amrita Saha, Vardaan Pahuja, Mitesh Khapra, Karthik Sankaranarayanan, and Sarath Chandar. [Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph](#). In *AAAI*, 2018.
- [178] Rishiraj Saha Roy, Joel Schlotthauer, Chris Hinze, Andreas Foltyn, Luzian Hahn, and Fabian Kuech. [Evidence Contextualization and Counterfactual Attribution for Conversational QA over Heterogeneous Data with RAG Systems](#). In *WSDM*, 2025.
- [179] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. [DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter](#). In *EMC2@NeurIPS*, 2019.
- [180] Denis Savenkov and Eugene Agichtein. [When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data](#). In *SIGIR*, 2016.
- [181] Uma Sawant, Saurabh Garg, Soumen Chakrabarti, and Ganesh Ramakrishnan. [Neural Architecture for Question Answering Using a Knowledge Graph and Web Corpus](#). *Information Retrieval Journal*, 2019.
- [182] Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. [Question Answering Over Temporal Knowledge Graphs](#). In *ACL*, 2021.
- [183] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. [Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings](#). In *ACL*, 2020.
- [184] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. [Toolformer: Language Models Can Teach Themselves to Use Tools](#). In *NeurIPS*, 2023.
- [185] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. [Bidirectional Attention Flow for Machine Comprehension](#). In *ICLR*, 2017.
- [186] Chao Shang, Guangtao Wang, Peng Qi, and Jing Huang. [Improving Time Sensitivity for Question Answering over Temporal Knowledge Graphs](#). In *ACL*, 2022.
- [187] Aditya Sharma, Apoorv Saxena, Chitranshu Gupta, Seyed Mehran Kazemi, Partha Talukdar, and Soumen Chakrabarti. [TwiRGCN: Temporally Weighted Graph Convolution for Question Answering over Temporal Knowledge Graphs](#). In *EACL*, 2023.

-
- [188] Saeedeh Shekarpour, Abhishek Nadgeri, and Kuldeep Singh. [QA2Explanation: Generating and Evaluating Explanations for Question Answering Systems over Knowledge Graph](#). In *IntEx-SemPar@EMNLP*, 2020.
- [189] Dan Shen and Dietrich Klakow. [Exploring Correlation of Dependency Relation Paths for Answer Extraction](#). In *COLING*, 2006.
- [190] Tao Shen, Xiubo Geng, Qin Tao, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. [Multi-Task Learning for Conversational Question Answering over a Large-Scale Knowledge Base](#). In *EMNLP-IJCNLP*, 2019.
- [191] Xiaoyu Shen, Gianni Barlacchi, Marco Del Tredici, Weiwei Cheng, Bill Byrne, and Adrià de Gispert. [Product Answer Generation from Heterogeneous Sources: A New Benchmark and Best Practices](#). In *ECNLP@ACL*, 2022.
- [192] Chenglei Si, Chen Zhao, and Jordan Boyd-Graber. [What’s in a Name? Answer Equivalence For Open-Domain Question Answering](#). In *EMNLP*, 2021.
- [193] Amit Singhal. [Introducing the Knowledge Graph: Things, not Strings](#). *Google Blog*, 2012.
- [194] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. [ASQA: Factoid Questions Meet Long-Form Answers](#). In *EMNLP*, 2022.
- [195] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. [YAGO: A Core of Semantic Knowledge](#). In *WWW*, 2007.
- [196] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. [PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text](#). In *EMNLP-IJCNLP*, 2019.
- [197] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. [Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text](#). In *EMNLP*, 2018.
- [198] Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. [Table Cell Search for Question Answering](#). In *WWW*, 2016.
- [199] Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. [Head-to-Tail: How Knowledgeable are Large Language Models \(LLMs\)? A.K.A. Will LLMs Replace Knowledge Graphs?](#) In *NAACL*, 2024.
- [200] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. [Sequence to Sequence Learning with Neural Networks](#). In *NeurIPS*, 2014.
- [201] Alona Sydorova, Nina Poerner, and Benjamin Roth. [Interpretable Question Answering on Knowledge Bases and Text](#). In *ACL*, 2019.
- [202] Tilahun Abedissa Taffa and Ricardo Usbeck. [Bridge-Generate: Scholarly Hybrid Question Answering](#). In *WWW*, 2025.
- [203] Alon Talmor and Jonathan Berant. [The Web as a Knowledge-Base for Answering Complex Questions](#). In *NAACL-HLT*, 2018.
- [204] Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. [MultiModalQA: Complex Question Answering over Text, Tables and Images](#). In *ICLR*, 2021.

- [205] Qingyu Tan, Hwee Tou Ng, and Lidong Bing. [Towards Benchmarking and Improving the Temporal Reasoning Capability of Large Language Models](#). In *ACL*, 2023.
- [206] Wang-Chiew Tan, Jane Dwivedi-Yu, Yuliang Li, Lambert Mathias, Marzieh Saeidi, Jing Nathan Yan, and Alon Y. Halevy. [TimelineQA: A Benchmark for Question Answering over Timelines](#). In *ACL*, 2023.
- [207] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. [From Natural Language Processing to Neural Databases](#). In *VLDB*, 2021.
- [208] Ghazaleh H Torbati, Andrew Yates, and Gerhard Weikum. [You Get What You Chat: Using Conversations to Personalize Search-based Recommendations](#). In *ECIR*, 2021.
- [209] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. [LLaMA: Open and Efficient Foundation Language Models](#). *arXiv*, 2023.
- [210] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. [Template-based Question Answering over RDF Data](#). In *WWW*, 2012.
- [211] Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. [Question Answering over Linked Data \(QALD-4\)](#). In *CLEF*, 2014.
- [212] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. [7th Open Challenge on Question Answering over Linked Data \(QALD-7\)](#). In *Semantic Web Challenges*, 2017.
- [213] Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. [Message Passing for Complex Question Answering over Knowledge Graphs](#). In *CIKM*, 2019.
- [214] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. [Question Rewriting for Conversational Question Answering](#). In *WSDM*, 2021.
- [215] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. [REL: An Entity Linker Standing on the Shoulders of Giants](#). In *SIGIR*, 2020.
- [216] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. [Graph Attention Networks](#). In *ICLR*, 2018.
- [217] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. [Recovering Semantics of Tables on the Web](#). In *VLDB*, 2011.
- [218] Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. [Artificial Artificial Artificial Intelligence: Crowd Workers Widely Use Large Language Models for Text Production Tasks](#). *arXiv*, 2023.
- [219] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas, and Maarten de Rijke. [Query Resolution for Conversational Search with Limited Supervision](#). In *SIGIR*, 2020.
- [220] Denny Vrandečić and Markus Krötzsch. [Wikidata: A Free Collaborative Knowledge Base](#). *Communications of the ACM*, 2014.

-
- [221] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. [CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation](#). In *EMNLP*, 2021.
- [222] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#). In *NeurIPS*, 2022.
- [223] Gerhard Weikum, Luna Dong, Simon Razniewski, and Fabian Suchanek. [Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases](#). *Foundations and Trends in Databases*, 2021.
- [224] William A. Woods. [Progress in Natural Language Understanding: An Application to Lunar Geology](#). In *AFIPS*, 1973.
- [225] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. [TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion](#). In *EMNLP*, 2020.
- [226] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vasilis Ioannidis, Karthik Subbian, James Y Zou, and Jure Leskovec. [STARK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases](#). In *NeurIPS*, 2024.
- [227] Wenqing Wu, Zhenfang Zhu, Qiang Lu, Dianyuan Zhang, and Qiangqiang Guo. [Introducing External Knowledge to Answer Questions with Implicit Temporal Constraints over Knowledge Base](#). *Future Internet*, 2020.
- [228] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. [A Comprehensive Survey on Graph Neural Networks](#). *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [229] Yao Xiao, Guangyou Zhou, and Jin Liu. [Modeling Temporal-Sensitive Information for Complex Question Answering over Knowledge Graphs](#). In *NLPCC*, 2022.
- [230] Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri, and Jens Lehmann. [Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings](#). In *NAACL*, 2021.
- [231] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. [Hybrid Question Answering over Knowledge Base and Free Text](#). In *COLING*, 2016.
- [232] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. [Question Answering on Freebase via Relation Extraction and Textual Evidence](#). In *ACL*, 2016.
- [233] Xiaojun Xu, Chang Liu, and Dawn Song. [SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning](#). *arXiv*, 2017.
- [234] Zihan Xu, Jiangang Zhu, Ling Geng, Yang Yang, Bojia Lin, and Daxin Jiang. [Learning to Generate Reformulation Actions for Scalable Conversational Query Understanding](#). In *CIKM*, 2020.
- [235] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. [Natural Language Questions for the Web of Data](#). In *EMNLP*, 2012.
- [236] Peilin Yang, Hui Fang, and Jimmy Lin. [Anserini: Enabling the Use of Lucene for Information Retrieval Research](#). In *SIGIR*, 2017.

- [237] Qian Yang, Qian Chen, Wen Wang, Baotian Hu, and Min Zhang. [Enhancing Multimodal Multi-hop Question Answering via Structured Knowledge and Unified Retrieval-Generation](#). In *MM*, 2023.
- [238] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. [End-to-End Open-Domain Question Answering with BERTserini](#). In *NAACL-HLT*, 2019.
- [239] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran Jiang, Ziyu Jiang, et al. [CRAG - Comprehensive RAG Benchmark](#). In *NeurIPS*, 2024.
- [240] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. [HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering](#). In *EMNLP*, 2018.
- [241] Junping Yao, Yijing Wang, Xiaojun Li, Cong Yuan, and Kaiyuan Cheng. [TERQA: Question Answering over Knowledge Graph Considering Precise Dependencies of Temporal Information on Vectors](#). *Displays*, 2022.
- [242] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. [Tree of Thoughts: Deliberate Problem Solving with Large Language Models](#). In *NeurIPS*, 2023.
- [243] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. [QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering](#). In *NAACL*, 2021.
- [244] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. [Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base](#). In *ACL-IJCNLP*, 2015.
- [245] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. [TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data](#). In *ACL*, 2020.
- [246] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. [QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension](#). In *ICLR*, 2018.
- [247] Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. [KG-FiD: Infusing Knowledge Graph in Fusion-in-Decoder for Open-Domain Question Answering](#). In *ACL*, 2022.
- [248] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. [Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task](#). In *EMNLP*, 2018.
- [249] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. [Explainability in Graph Neural Networks: A Taxonomic Survey](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [250] Hamed Zamani, Johanne R Trippas, Jeff Dalton, and Filip Radlinski. [Conversational Information Seeking](#). *Foundations and Trends in Information Retrieval*, 2023.

-
- [251] John M. Zelle and Raymond J. Mooney. [Learning to Parse Database Queries Using Inductive Logic Programming](#). In *AAAI*, 1996.
- [252] Heidi C. Zhang, Sina J. Semnani, Farhad Ghassemi, Jialiang Xu, Shicheng Liu, and Monica S. Lam. [SPAGHETTI: Open-Domain Question Answering from Heterogeneous Data Sources with Retrieval and Semantic Parsing](#). In *ACL*, 2024.
- [253] Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. [Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering](#). In *ACL*, 2022.
- [254] Ruiqiang Zhang, Yuki Konda, Anlei Dong, Pranam Kolari, Yi Chang, and Zhaohui Zheng. [Learning Recurrent Event Queries for Web Search](#). In *EMNLP*, 2010.
- [255] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. [Variational Reasoning for Question Answering with Knowledge Graph](#). In *AAAI*, 2018.
- [256] Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. [Answering Any-hop Open-domain Questions with Iterative Document Reranking](#). In *SIGIR*, 2021.
- [257] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. [A Survey of Large Language Models](#). *arXiv*, 2023.
- [258] Wenting Zhao, Zhongfen Deng, Shweta Yadav, and Philip S Yu. [Heterogeneous Knowledge Grounding for Medical Question Answering with Retrieval Augmented Large Language Model](#). In *WWW*, 2024.
- [259] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. [TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance](#). In *ACL*, 2021.
- [260] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. [Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering](#). *arXiv*, 2021.
- [261] David Ziegler, Abdalghani Abujabal, Rishiraj Saha Roy, and Gerhard Weikum. [Efficiency-aware Answering of Compositional Questions using Answer Type Prediction](#). In *IJCNLP*, 2017.