Best of Both Worlds: Towards Improving Temporal Knowledge Base Question Answering via Targeted Fact Extraction

Nithish Kannen[♣§*], **Udit Sharma**[†], **Sumit Neelam**[†], **Dinesh Khandelwal**[†]
Shajith Ikbal[†*], **Hima Karanam**[†*], **L Venkata Subramaniam**[†]

[†] IBM Research, India [♣] Amazon Alexa AI, UK

Abstract

Temporal question answering (QA) is a special category of complex question answering task that requires reasoning over facts asserting time intervals of events. Previous works have predominately relied on Knowledge Base Question Answering (KBQA) for temporal QA. One of the major challenges faced by these systems is their inability to retrieve all relevant facts due to factors such as incomplete KB and entity/relation linking errors (Patidar et al., 2022). A failure to fetch even a single fact will block KBQA from computing the answer. Such cases of KB incompleteness are even more profound in the temporal context. To address this issue, we explore an interesting direction where a *targeted temporal fact extraction* technique is used to assist KBQA whenever it fails to retrieve temporal facts from the KB. We model the extraction problem as an open-domain question answering task using off-the-shelf language models. This way, we target to extract from textual resources those facts that failed to get retrieved from the KB. Experimental results on two temporal QA benchmarks show promising **~30%** & **~10%** relative improvements in answer accuracies without any additional training cost.

1 Introduction

Complex Question Answering involves the integration of multiple facts identified and extracted from disjoint pieces of information (Vakulenko et al., 2019; Saxena et al., 2020; Fu et al., 2020; Neelam et al., 2022a). Two critical components in systems trying to achieve this are: 1) Knowledge Source - to retrieve relevant facts, and 2) Reasoning - to reason over relevant facts to derive the final answer. Both

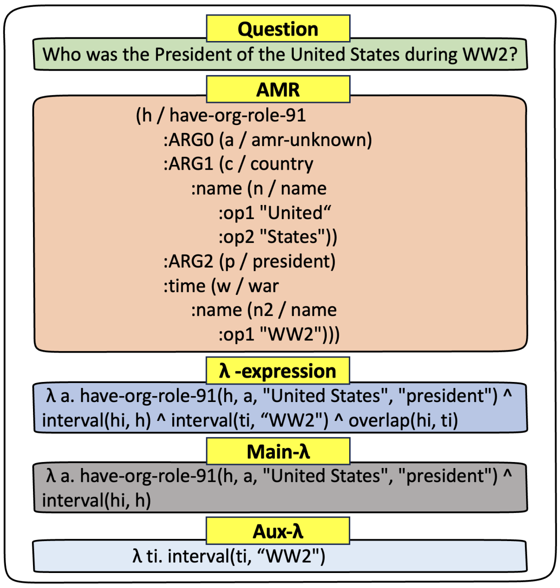

Figure 1: Example question, its AMR and λ-expressions

NLP and Semantic Web communities have shown immense interest in this problem lately. Work in the NLP community has primarily focused on using textual data as a knowledge source, and typically uses deep-neural models to represent knowledge and for reasoning. While those approaches achieve impressive accuracies (Zhang et al., 2021, 2020), they are typically limited in their reasoning capabilities. Although reasoning is starting to receive attention in NLP community (Wei et al., 2023), reasoning over large amounts of unstructured knowledge in text is still a challenge. Work in the Semantic Web community has primarily focused on Knowledge Base Question Answering (KBQA), where a KB is used to retrieve facts. While better-equipped for complex reasoning (Bhutani et al., 2020; Wu et al., 2021), they typically suffer from incomplete knowledge (Patidar et al., 2022).

In this paper, we present a novel combination of successful elements from past approaches, i.e., KB-based and text-corpus based, for complex QA, in a way to overcome their individual limitations.

[§] This work was done when Nithish Kannen was at IBM Research, India as a Summer Intern. At that time he was also an (integrated bachelor's+master's) Electrical Engineering Student at IIT Kharagpur, India.

[*] Correspondence e-mails: nithishkannen@gmail.com, shajmoha@in.ibm.com, hkaranam@in.ibm.com

Our combination strategy is to use a targeted extraction technique that would assist KBQA whenever it fails. KBQA failure is when it fails to retrieve relevant facts needed to answer the question from the KB because of reasons such as *incomplete KB* and *inaccurate entity/relation linking*. Identification of these specific points of KBQA failure is a critical step in our approach, which is enabled by decomposition of the question's logical representation obtained by semantic parsing. The targeted extraction technique compensates for those KB failures by extracting facts from textual resources in an open-domain QA fashion. Concretely, we make effective utilization of the KB (reliable but not exhaustive) and the textual resources (vast but noisy), essentially combining the best of both worlds.

In this work, we focus on *temporal questions*. They additionally involve reasoning over temporal facts, i.e., assertions on points and intervals in time of events[1]. For example, to answer *Who was the President of the United States during World War 2?*, we need to know the temporal facts of both *World War 2* and the list of all *US presidents*.

The main contributions of our work are: 1) A novel combination of *KBQA* and *textual-extraction* for temporal question answering. 2) λ-calculus based semantic representation to identify KBQA gaps. 3) An open-domain QA style approach for targeted extraction of temporal facts from textual resources using off-the-shelf models. We show that it is possible to achieve significant improvements on two temporal QA benchmarks even without any task-specific training and believe the promising initial results will foster research in this direction.

Related Work: GRAFT-Net (Sun et al., 2018) and PullNet (Sun et al., 2019; Xiong et al., 2019) utilize both KB and textual resources but do not address temporal reasoning. Unlike approaches using end-to-end neural models, we adopt a modular approach as in (Neelam et al., 2022a) because of the flexibility it offers to combine textual extraction with the KBQA, with additional benefits such as interpretability and easy domain-adaptation. Another main difference in our textual extraction approach is the usage of LMs off-the-shelves, hence bypassing the need for domain-specific training and datasets that are hard to obtain. A detailed literature review is presented in A.6.

[1]In this paper, we consider entities and facts (represented as triples in KBs) with associated time intervals as events. For example, {*World War 2*, (start time:1939, end time:1945)}.

2 Our Approach

Figure 2 shows a block diagram of our proposed approach, with two groups of modules: 1) Upper line of *KBQA pipeline* modules, and 2) Lower line of *Extraction pipeline* modules for targeted fact extraction. Our strategy is to bank on the *KBQA pipeline*, owing to its reliability, and trigger *Extraction pipeline* only to aid KBQA when it fails.

2.1 KBQA Pipeline

For *KBQA pipeline*, we adopt a modular design as in (Neelam et al., 2022a; Kapanipathi et al., 2021), with two high-level modules: (1) **Question Understanding** to derive logical semantic representation of the NL question and to further decompose it. (2) **KB Linking and Answering** to ground the Entity and Relation mentions in the logical representation onto the KB. Both these modules in our approach are similar to (Neelam et al., 2022a) except for the event specific decomposition described next.

We use λ-expression constructed from Abstract Meaning Representation (AMR)(Banarescu et al., 2013) for logical semantic representation of the questions (Neelam et al., 2022a). Figure 1 gives an illustration of NL question, its AMR and λ-expression. As shown, the λ-expression compactly represents the mentions of events in the question as its sub-components, facts about those events needed from the knowledge source, and the reasoning steps needed to derive the final answer. We decompose λ-expression into two components to help localize the points of KB failures: 1) *Main-λ*: Part of the λ-expression related to the unknown variable, i.e., main event being questioned. In Figure 1, a is the unknown variable, whose value if found is the answer. 2) *Aux-λ*: part of the λ-expression not related to the unknown variable, but related to the other events in the question as shown in Figure 1 . This part adds temporal constraints on the answer candidates. We use a simple rule-based approach to perform this decomposition, where unknown variable is used as anchor to segregate the respective components. We provide additional details in A.1.

2.2 Targeted Temporal Fact Extraction

We know a KBQA failure has occurred when the complete lambda expression fails to return an answer from the KB. The first critical step in our approach is to localize points of KBQA failure within the lambda expression. For this, we examine answers derived independently for the aux-λ

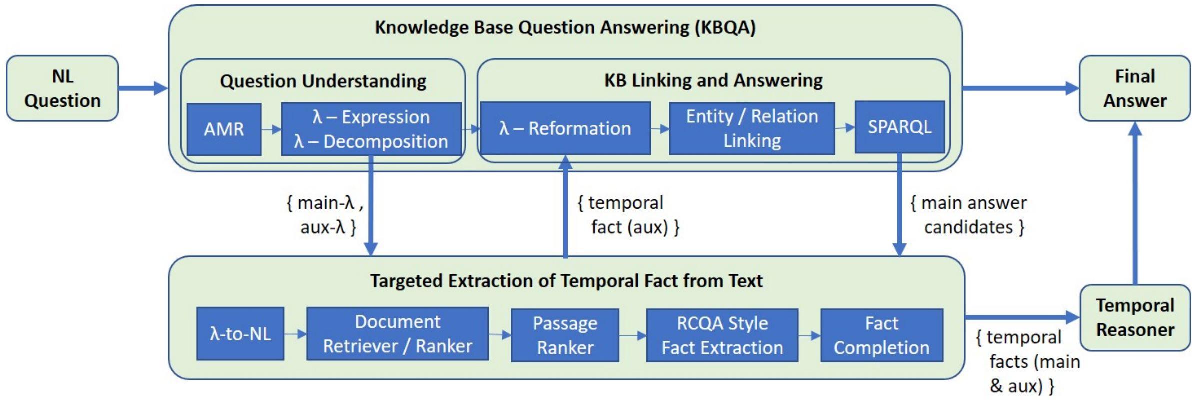

Figure 2: An illustration of the proposed approach. Upper line of modules correspond to the KBQA pipeline, while lower line of modules are related to targeted fact extraction from textual resources.

and main-λ using the KBQA pipeline in order to classify the failure into one of the below cases[2]:

- *Aux Failure*: Temporal fact corresponding to the temporal constraint (aux-λ) is missing in the KB. For example, in Figure 1, when time interval of *World War 2* is missing in the KB.
- *Main Failure*. Temporal fact(s) corresponding to the unknown variable is missing in the KB. For example, in Figure 1, when time interval of *Franklin D. Roosevelt* (the US president during WW2) as president in office is missing.

The goal now is to extract the identified missing facts from textual resources using the extraction pipeline (described next). In this way, we assist KBQA using *targeted fact extraction*, which has several advantages: 1) A focused extraction from text (guided by KB) makes textual fact extraction, which is usually noisy, reliable. 2) The complementing strengths of textual resources and KBs are leveraged to overcome their individual limitations.

Aux Failure. Answer for aux-λ is a time interval, i.e., composed of time intervals of all the events part of aux-λ, that imposes temporal constraints on the answer candidates. We extract the KB missing fact through the extraction pipeline and construct a reformed λ-expression, which is simply replacing the auxiliary part of the original λ-expression with the extracted time interval. This will enable answer derivation without fetching facts related to aux-λ from KB. We show an example of this in A.3.1.

Main Failure: Here we attempt to extract missing facts corresponding to the main-λ. However, we do not fully rely on textual resources for main-λ, because it represents event with unknown variable. We take that part of the main λ-expression that would fetch the answer candidates from the KB (leaving out the temporal fact specific components), and pass that onto the *KBQA pipeline*. For each answer candidate obtained, we extract the temporal information via the extraction pipeline as in A.3.2.

Temporal Reasoning. Upon successful gathering of all the temporal facts (i.e., time intervals), either from the KB or from text, we use *Temporal Reasoning* module to select the final answer from among the answer candidates of the main-λ. For example in Figure 1, λ-expression component overlap(hi, ti) corresponds to the temporal reasoning step, which essentially represents the overlap between the time intervals ti (of *Ww2*) and hi (of answer candidates of the main-λ). Candidates that comply with the reasoning condition are chosen as the final answer. Other temporal reasoning categories handled include *before*, *after*, *now* etc,. Algorithm 1 describes our overall flow sequence.

2.3 Extraction pipeline

There are 3 scenarios where we may look to extract facts from textual resources: 1) KBQA fails for aux-λ because of Linking failure, 2) KBQA fails for aux-λ because of missing temporal fact, and 3) KBQA fails for main-λ because of missing temporal fact. We pose extraction as an open-domain QA problem where: 1) for each missing fact, the corresponding NL query is generated, 2) top-k passages relevant to the NL query are retrieved from textual resources, and 3) a Reading Comprehension QA (RCQA) style answer derivation is performed by treating NL query as the question and top-k retrieved passages as the context. We show that even a naive pipeline with pre-trained LMs as the back-

[2]Note that there could also be a combination of failures. We use the flow sequence in Algorithm 1 to handle all cases.

bone demonstrates significant improvements and argue that an improved (domain-trained) pipeline will only further boost performance.

λ to NL. For example, for the aux-λ in Figure 1, the equivalent NL query is *When was WW2?*. We use simple rules to convert λ of the missing facts into its corresponding NL query. Since we deal with temporal facts, all the queries start with *When*, and we add *was* or *did* depending on whether the event being considered is entity-based or triple-based.

Document retrieval. For each NL query, we perform document retrieval in 2 steps: first, get a list of relevant documents and then choose a top-k scored list of passages from them. (1) Entities of the question are extracted using BLINK (Wu et al., 2019) and Wikipedia pages of all the entities are collected allowing minimal lexical variations[3]. We also use NL text query generated from λ-expression to search on MediaWiki API[4]. (2) We use Siamese-BERT networks (Reimers and Gurevych, 2019) to rank passages within the retrieved documents. The Bi-Encoder picks out top-50 relevant passages based on passage-query similarity and the Cross-Encoder re-ranks them. We use public checkpoints[5] trained on the MS-MARCO dataset (Bajaj et al., 2018).

QA based fact extraction. We use top-3 ranked passages as context to derive answer for the NL query in Reading Comprehension QA (RCQA) style. We use BERT[6] trained on the SQUAD (Rajpurkar et al., 2016)) dataset as the QA model. Note that RCQA style extraction gives one answer, which is sufficient for point-in-time extraction. For time intervals (start and end times) we further take the sentence identified by the RCQA model and use its AMR tree to obtain time interval by examining sibling nodes to that containing the RCQA answer.

3 Experimental Setup

We used Wikidata as KB and Wikipedia as textual resource, and evaluate on two aspects: 1) Temporal QA performance of the overall system and 2) Targeted temporal fact extraction performance. We experiment with 2 datasets: 1) *TempQA-WD* (Neelam et al., 2022b) with 839 questions and 2) *TimeQuestions* (Jia et al., 2021) (Train-9708, Test-3237).
Baselines: 1) *Only-KB* - answers only from KBQA, 2) *KB+TemporalText* - extracting temporal facts only from text and the remaining facts from KB, and 3) *Open-Domain-QA* - A state-of-the-art open-domain-QA system called RAG (Lewis et al., 2020) that answers purely from text. RAG consists of a retriever based on Dense Passage retrieval (DPR) and a generator based on BART jointly trained. We refer the readers to the original paper for more details on this baseline.

3.1 Implementation Details

Our system pipeline is implemented using Flow Compiler[7] (Chakravarti et al., 2019) that stitches together the gRPC services of the individual modules. λ-expressions are defined using ANTLR grammer. SPARQL queries are run on public Wikidata end point[8]. We reuse the KBQA pipeline implementation of (Neelam et al., 2022b).

4 Results and Discussion

Table 1 shows performance comparison of our approach (*KB+Text*) against the baselines on *TempQA-WD* dataset. *KB+Text* achieves an improvement of 0.095 in F1 score (∼30% relative improvement) over *Only-KB*, demonstrating the effectiveness of our approach in making a targeted utilization of the textual resources to assist KBQA. Comparison to *KB+TemporalText* illustrates the reliability of facts obtained from KB, whenever available. Performance of *Open-Domain QA* is inferior to *Only-KB*, illustrating the superior reasoning capability of KBQA systems.

System	Precision	Recall	F1
Only-KB	0.320	0.329	0.321
KB+TemporalText	0.224	0.227	0.212
Open-Domain QA (RAG)	0.291	0.240	0.252
KB+Text (proposed)	0.423	0.434	0.416

Table 1: Performance comparison on *TempQA-WD*.

Since extraction pipeline is a critical component in our system, we also evaluate its independent accuracy using a small set of 3709 NL queries (added in supplementary material) generated from our system for facts existing in KB. The extraction pipeline performs better than a state-of-the-art open domain QA system as shown in Table 3.

Due to discrepancies discussed in A.4, we could not use our KBQA pipeline to evaluate on *TimeQuestions* dataset. Alternatively, we used the end-

[3]https://pypi.org/project/fuzzywuzzy/
[4]https://www.mediawiki.org/wiki/Download
[5]https://www.sbert.net/
[6]https://huggingface.co/
[7]https://github.com/IBM/flow-compiler
[8]https://query.wikidata.org/

KBQA

Question: Who was the President during the Dawes Plan?
λ-expression: `λ a. have-org-role-91(h, a, "president") ^ interval(hi, h) ^ type(tp2,"Dawes", "plan") ^ interval(p2i, "Dawes") ^ overlap(hi, p2i)`
Is Answerable from KBQA?: No
KBQA Issue: Auxiliary Fact Missing (temporal information of Dawes Plan)

Missing Fact Extraction

Retrieved textual info: The Dawes Plan as proposed by the Dawes Committee, chaired by Charles G. Dawes was a plan in **1924** that successfully resolved the issue of World War I reparations that Germany had to pay.
Auxiliary temporal fact: 1924
Reformed λ-expression: `λ a. interval(hi, h) ^ have-org-role-91(h, a, "president") ^ interval(p2i, date("dd-mm-1924")) ^ overlap(hi, p2i)`
Answer with reformed λ-expression (from KB): Calvin Coolidge

Figure 3: Illustration of a working example showing the KBQA failure occurring due to missing auxiliary fact that is substituted by temporal fact extraction and finally reforming the lambda expression by hard-coding the missing fact.

Error $\rightarrow$	Incorrect extraction	KBQA error	Parsing error	Miscellaneous
Count $\rightarrow$	21	9	8	12

Table 2: Error analysis on 50 randomly selected test instances incorrectly answered by our system.

System	Precision	Recall	F1
Open-Domain QA (RAG)	0.051	0.048	0.049
KB+Text Extraction	0.171	0.163	0.165

Table 3: Evaluation of *Extraction pipeline*.

to-end trained *EXAQT* (Jia et al., 2021) system as the baseline KBQA system that was purpose-built for the specific dataset. With this, we then built an equivalent of our proposed approach called *EXAQT+Text*, by running our *Extraction pipeline* wherever *EXAQT* fails to answer. The performance of *EXAQT+Text* is better for all threshold values with improvement reaching upto 10% as shown in Table 4 . This demonstrates that our approach of targeted extraction backing up KBQA is resilient to changes in underlying KBQA. In order to benefit future work, we conduct an error analysis by manually examining 50 randomly selected test samples from TempQA-WD that were incorrectly answered by out proposed system. We classify the error into following types: 1) *Incorrect extraction*: the text pipeline extracted incorrect facts, 2) *KBQA error*: KBQA produced erroneous answers after missing fact was correctly extracted from text, 3) *Parsing error*: incorrect parsing of the question, 4) *Miscellaneous*: consists of miscellaneous issues like lexical and date representation variations. Table 2 shows the results of the experiment and further highlights the scope of improvement in the extraction pipeline. Figure 3 is an illustration of how a KB Aux-Failure is handled in our pipeline. The missing temporal fact '1924' is extracted from text and the λ-expression is reformed.

Code and experimental setup used for our experiments is at https://github.com/IBM/tempqa-wd/tree/main/targeted-extraction

System $\rightarrow$	EXAQT			EXAQT+Text		
Threshold$\downarrow$	P	R	F1	P	R	F1
0.1	0.396	0.574	0.435	0.406	0.586	0.446
0.3	0.446	0.551	0.465	0.464	0.570	0.483
0.5	0.466	0.520	0.469	0.496	0.553	0.499
0.7	0.468	0.490	0.460	0.505	0.531	0.497
0.9	0.458	0.451	0.440	0.506	0.502	0.486

Table 4: Performance on TimeQuestions test set.

5 Conclusion

In this paper, we propose an approach to combine the knowledge resources of KB (structured) and text (unstructured) for temporal QA by using textual resources to aid wherever KBQA fails. In our approach, a semantic representation of the question as λ-expression is used to represent the set of facts and reasoning required to answer a question. Those facts that failed to get fetched from KBQA are extracted from the textual resources using RCQA style fact extraction. This way, we perform targeted extraction of temporal facts to compensate for KBQA failures. The results of experimental evaluation on two temporal QA benchmark show the effectiveness of our approach even without any additional training cost. We additionally conduct error analysis to highlight the scope of improvement in order to guide future work.

6 Limitations

The proposed approach focuses only on temporal reasoning and work is required to generalize it across other reasoning types. In this work we addressed temporal QA as we identified missing KB facts as a major pain point in the temporal context (adding temporal facts to KBs has started gaining momentum only in recent times). The proposed approach will have to be modified to handle more complex questions that can feature multiple aux-λ (temporal constraints). Our λ decomposition algorithm that is hand-crafted works well for temporal questions, but may have to be tested for other reasoning tasks with appropriate modifications. It would be interesting to come up with learning-based methods for the same. The *Extraction pipeline* is triggered only when KBQA fails to return answer. However KBQA approaches can also produce wrong answers and we can potentially develop methods to predict such errors. To aid future research, we provide a detailed error analysis in Appendix highlighting areas that require improvement. Our *Extraction pipeline* has a large scope of improvement. The document retrieval approach is purely based on lexical overlap and considering query semantics would improve it. Our QA and ranker LMs can be finetuned for the domain in-focus. Alternatively, training end-to-end neural extraction modules remains to be investigated. Performance metrics used to evaluate KBQA systems do not account for mismatches arising because of lexical variations or date representation variations. Using a better metric can provide a better picture of the overall performance.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1191–1200. ACM.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 178–186. The Association for Computer Linguistics.

Nikita Bhutani, Xinyi Zheng, Kun Qian, Yunyao Li, and H. Jagadish. 2020. Answering complex questions by combining information from curated and extracted knowledge bases. In *Proceedings of the First Workshop on Natural Language Interfaces*, pages 1–10, Online. Association for Computational Linguistics.

Rishav Chakravarti, Cezar Pendus, Andrzej Sakrajda, Anthony Ferritto, Lin Pan, Michael Glass, Vittorio Castelli, J William Murdock, Radu Florian, Salim Roukos, and Avi Sil. 2019. CFO: A framework for building production NLP systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 31–36, Hong Kong, China. Association for Computational Linguistics.

Tarcísio Souza Costa, Simon Gottschalk, and Elena Demidova. 2020. Event-qa: A dataset for event-centric question answering over knowledge graphs. *CoRR*, abs/2004.11861.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. *CoRR*, abs/2007.13069.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018a. Tempquestions: A benchmark for temporal question answering. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1057–1062. ACM.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018b. TEQUILA: temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1807–1810. ACM.

Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 792–802.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander G. Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois P. S. Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G. P. Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3884–3894. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N. Ioannidis, Soji Adeshina, Phillip R. Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2021. Tempoqr: Temporal question reasoning over knowledge graphs.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G P Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme LimaRyan Riegel, Francois Luus, and L Venkata Subramaniam. 2021. Sygma: System for generalizable modular question answering overknowledge bases.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G P Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme LimaRyan Riegel, Francois Luus, and L Venkata Subramaniam. 2022a. Sygma: System for generalizable modular question answering overknowledge bases. In *Findings of the Empirical Methods in Natural Language Processing: EMNLP 2022*.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, et al. 2022b. A benchmark for generalizable and interpretable temporal question answering over knowledge bases. *arXiv preprint arXiv:2201.05793*.

Mayur Patidar, Avinash Singh, Prayushi Faldu, Lovekesh Vig, Indrajit Bhattacharya, and Mausam. 2022. Do i have the knowledge to answer? investigating answerability of knowledge base questions.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6663–6676, Online. Association for Computational Linguistics.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.

Jiaxin Shi, Shulin Cao, Liangming Pan, Yutong Xiang, Lei Hou, Juanzi Li, Hanwang Zhang, and Bin He. 2020. Kqa pro: A large-scale dataset with interpretable programs and accurate sparqls for complex question answering over knowledge base.

Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2380–2390. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4231–4242. Association for Computational Linguistics.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *Proceedings of the 16th International Semantic Web Conference (ISWC)*, pages 210–218. Springer.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1431–1440. ACM.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Trans. Assoc. Comput. Linguistics*, 8:183–198.

Dekun Wu, Nana Nosirova, Hui Jiang, and Mingbin Xu. 2019. A general fofe-net framework for simple and effective question answering over knowledge bases. *CoRR*, abs/1903.12356.

Peiyun Wu, Yunjie Wu, Linjuan Wu, Xiaowang Zhang, and Zhiyong Feng. 2021. Modeling global semantics for question answering over knowledge bases. *CoRR*, abs/2101.01510.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Improving question answering over incomplete kbs with knowledge-aware reader. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4258–4264. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. Sg-net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9636–9643. AAAI Press.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2021. Retrospective reader for machine reading comprehension. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14506–14514. AAAI Press.

A Appendix

A.1 Details of KBQA pipeline (Neelam et al., 2021)

Here we describe the modules in the KBQA pipeline that is adopted from (Neelam et al., 2021).

A.1.1 Question Understanding

The goal of *Question Understanding* is to 1) transform NL questions into corresponding λ-expressions that logically represent the set of event-specific facts needed from the KB and the reasoning needed to be performed to derive the answer and 2) further perform event-specific decomposition. We use method as in (Neelam et al., 2022a) to construct λ-expressions of the questions from their AMR (Abstract Meaning Representation) (Banarescu et al., 2013). AMR encodes meaning of the sentence into a rooted directed acyclic graph where nodes and edges represent concepts and relations respectively. Such a representation is useful because event-specific decomposition of the question is represented to some extent as sub-paths and sub-graphs in the AMR graph. Figure 1 shows an illustration of AMR and λ-expression for example question. This example illustrates how λ-expression compactly represents, the mentions of events in the question (as its sub-components), facts about those events (that need to be fetched from the knowledge source), and the reasoning steps (that need to be performed to derive the final answer).

λ-expression constructed for the question is further processed to decompose into components:

1. *Main-λ*: Part of the λ-expression related to the unknown variable, i.e., main event being questioned. For example in Figure 1, a is the unknown variable, whose value if found is answer to the question.
2. *Aux-λ*: part of the λ-expression not related to the unknown variable, but related to the rest of the events mentioned in the question. This part serves the purpose of adding temporal constraint to the candidate answer values for the unknown variable.

We use a rule based approach to perform decomposition, that simply uses unknown variable as anchor to segregate the respective components. Note that the decomposed λ-expressions play a critical role in our approach to identify the points of failures of the *KBQA pipeline* and to further decide on the use of *Extraction pipeline*.

A.1.2 KB Linking and Answering

This is essentially a step to ground the Entity and Relation mentions of λ-expression to the KB, i.e., map the elements of λ-expression onto the corresponding KB elements. Relation mentions are the predicates (for example, *have-org-role* in Figure 1) and entity mentions are the arguments (for example, *United States*, *president*, and *Ww2* in Figure 1). The goal of linking is to map, for example *Ww2* to a node in KB corresponding to *World War 2* (Wikidata id wd:Q362). After linking, we generate corresponding SPARQL queries that when executed on the KB endpoint would fetch the intended KB facts. Our approach to linking and SPARQL generation is similar to that of (Neelam et al., 2022a).

A.2 Targeted Extraction Algorithm

Algorithm 1 Algorithm for the overall approach illustrated in Figure 2 and its flow sequence.

```
lambda = GetLambda(question)
ans_list = GetKBAnswer(lambda)
if ans_list is empty then                          ▷ KBQA Failure
    main, aux = Decompose_Lambda(lambda)
    ans_list = GetKBAnswer(aux)
    if ans_list is empty then                      ▷ Aux Failure
        fact = Extract_From_Text(aux)
        aux_fact = fact                            ▷ for later use
        reformed_lambda =
                ReformLambda(fact, lambda)
        ans_list =
                GetKBAnswer(reformed_lambda)
        if ans_list is not empty then
            return                                 ▷ Ans Found
        end if
    end if
    ans_list = GetKBAnswer(main)
    candidate_facts = []
    for candidate in ans_list do
        fact = Extract_From_Text(candidate)
        candidate_facts.append(fact, candidate)
    end for
    ans =
      TemporalReasoner(candidate_facts, auxfact)
    return                                         ▷ Ans Found
else
    return                  ▷ Ans Found, No missing fact in KB
end if
```

A.3 Details of Targeted Extraction from Text

Our goal is to use textual resources to assist KBQA failures, which can happen for two reasons:

1. *Linking failure* - when KB linking step fails to successfully map mentions in the λ-expression to the corresponding KB entities and relations. For example, in Figure 1 when mention *Ww2* fails to get linked to the *World War 2* node in the KB.

2. *Missing facts* - KBs are known to be incomplete, and hence may fail to fetch a specific fact, simply because it is not present in the KB. For example, if temporal information corresponding to *World War 2* is not present in the KB, attempt to fetch time interval corresponding to λ-expression part interval(ti, “Ww2") would fail.

λ-expression specifies all facts that need to be fetched from the KB. A failure to fetch even a single fact would block KBQA from computing the final answer. To handle failures we need to know the specific facts that failed to get fetched from the KB, so that we can look for them in the textual resources. For this purpose, we categorize KBQA failure as below, based on the decomposed λ-expressions where failure happens into **Aux Failure** and **Main Failure**. We present the flow sequence in Algorithm 1.

A.3.1 Aux Failure

This issue arises due to missing temporal fact corresponding to the aux-λ in Upon successful extraction of time interval for aux-λ, from textual resources, we construct a reformed λ-expression from the original λ-expression by simply replacing auxiliary part with the time interval of aux-λ. For example, λ-expression in Figure 1 will be reformed as:

λ a. have-org-role-91(h, a, "United States", "president") $\wedge$ interval(hi, h) $\wedge$ overlap(hi, (interval_start:1939-09-01, interval_end:1945-09-02)).

Note that this reformed λ-expression does not contain aux-λ. Its NL equivalent is *Who was the President of the United States during period from 1st September 1939 to 2nd September 1945?* Thus if reformed λ-expression is passed onto the KBQA pipeline (instead of original λ-expression), it should result in the same answer, but without the need to fetch facts related to aux-λ from the KB.

A.3.2 Main Failure

For example, in Figure 1 main-λ corresponds to the list of all the US presidents and their time intervals in office as the president. We make an assumption that we can always get the list of all answer candidates from the KB itself, but may need to look into textual resources only for temporal fact about them. For example, in Figure 1 we take that part of the main λ-expression that would fetch the answer candidates from the KB (leaving out the temporal fact specific components), i.e.,

λ a. have-org-role-91(h, a, "United States", "president")

and pass that onto the *KBQA pipeline*. Then for each answer candidate obtained we try to extract time interval from the textual resource. For example, if *Franklin D. Roosenvelt* is one of the answer candidates, we try to extract the time interval of *Franklin D. Roosenvelt* being the US president from the textual resources.

Note that we resort to extraction from textual resources only for those facts that failed to get fetched from the KB. We believe this approach of targeted extraction from the text is likely to be more accurate than unrestricted extraction, because we are looking to extract facts with a set of known variables and only one unknown variable.

A.4 Discrepancies in Evaluating on TimeQuestions

We could not directly evaluate our approach on *TimeQuestions* because a few discrepancies obseverd in that datasets such as, invalid answers (with change in time) and incorrect porting of the answers from old datasets without verifying validity on Wikidata. It is because *TimeQuestions* (Jia et al., 2021) is built from older datasets on different KBs, by mapping only the final answer entities onto the corresponding Wikidata entities and it seems while mapping the validity of the answer is not verified carefully. Hence, we resort to using EXAQT (Jia et al., 2021) built on *TimeQuestions* as the base KBQA pipeline and build an equivalent of our proposed approach called EXAQT+Text for comparison.

A.5 Implementation Details

Our system pipeline is implemented using Flow Compiler[9] (Chakravarti et al., 2019) that stitches together the gRPC services of the individual modules. λ-expressions are defined using ANTLR grammer. SPARQL queries are run on public Wikidata end point[10]. We reuse the KBQA pipeline implementation of (Neelam et al., 2022b).

A.6 Related Work

Although Complex KBQA has been an active research topic (Vakulenko et al., 2019; Saxena et al., 2020; Wu et al., 2021; Shi et al., 2020), there has been very limited research focused on Temporal

[9]https://github.com/IBM/flow-compiler
[10]https://query.wikidata.org/

KBQA. Temporal Questions require identification of time intervals of events and temporal reasoning.

A.6.1 Temporal KBQA Datasets:

TempQuestions (Jia et al., 2018a) is one of the first publicly available temporal KBQA dataset consisting of 1271 questions. However, this dataset was annotated over FreeBase, which is no longer maintained and was officially discontinued in 2014. SYGMA (Neelam et al., 2022a) introduced a subset of TempQuestions that can be answered over wikidata called TempQA-WD. TimeQuestions(Jia et al., 2021) is another temporal QA dataset that is curated from existing QA datasets and mapping the answers to Wikidata. We use TempQA-WD, and TimeQuestions data sets to evaluate our approach. CRONQUESTION (Saxena et al., 2021) is another temporal KBQA dataset that uses its own KB drawn from Wikidata. Event-QA dataset (Costa et al., 2020) is based on Event-KG, curated from DBpedia, Wikidata and YAGO. Since these datasets are generated in a template based manner using existing facts from the KBs, they do not represent the real world challenge of incomplete KBs. One of the main goals of our approach is to handle the issue of incomplete KBs.

A.6.2 Temporal KBQA Systems:

TEQUILA (Jia et al., 2018b) is one of the first attempts to address temporal question answering over KBs. It used an existing KBQA engine (Abujabal et al., 2017) to answer individual sub-questions and perform a temporal reasoning over the answers to derive the final answer. SYGMA (Neelam et al., 2022a) is another system that works on a Wikidata and uses λ-expressions to represent the facts and their temporal reasoning operators. EXAQT (Jia et al., 2021) is another temporal KBQA system that uses entity and temporal embeddings to generate final answers. TempoQR (Mavromatis et al., 2021) is another system that tries to improve on top of CRONQA (Saxena et al., 2021) and introduces temporal KB-completion aspect into temporal Questions answering. We did not consider these two systems as they work on curated subset of wikidata which has all the temporal facts to answer the given dataset. TEQUILA uses a pre-specified set of temporal signals (10 signal words) to decompose questions into sub-questions at sentence level in a rule-based manner. Instead, we follow the approach similar to SYGMA that use a sophisticated semantic parsing approach involving AMR (Abstract Meaning Representation) (Banarescu et al., 2013) and λ-calculus (Zettlemoyer and Collins, 2012) to get logical representations of the questions. This enables decomposition of the questions at semantic level and is likely robust to linguistic variations as well.

A.7 KB + Text for QA

There have been past work exploring effectiveness of using KB and text resources for complex QA (Sun et al., 2018; Xiong et al., 2019). However, none of them address the temporal context addressed in this work. Prior work using a combination of KB and text have largely been based on end-to-end neural models. GRAFT-Net (Sun et al., 2018) constructs a sub-graph from KB and text corpora using an early fusion technique. The task of QA is then reduced to binary classification over the nodes of this sub-graph. PullNet (Sun et al., 2019) proposes to build sub-graph through an iterative process(Xiong et al., 2019), utilise a graph-attention based KB reader and knowledge-aware text reader.

All these methods are based on end-to-end neural models that require large amount of training data and offer little interpretability, which is essential to evaluate intermediate stages of complex QA systems. Additionally, labeling large amounts of data for KBQA is hard (Trivedi et al., 2017). In this work, we extend modular approach described in (Neelam et al., 2022a), additionally incorporating it with a targeted extraction pipeline. We made this choice as this particular approach integrates multiple, reusable modules that are pre-trained for their specific individual tasks (semantic parsers, entity and relational linkers, rankers and re-rankers and reading comprehension model) thus offering interpretability and flexibility for optimal combination of textual extraction with KBQA. Additionally, this does not require a large amount of domain-specific training data.

A.7.1 Question Decomposition

Our work uses a form of logical query decomposition, based on λ-expression of the NL question, to help effectively combine the KB with the text resources. Some of the past work in the literature on QA have also explored question decomposition. BREAK IT down (Wolfson et al., 2020) is a popular benchmark data that captures complex question as an ordered list of tasks, that when executed in sequence will derive the final answer. It introduced question decomposition meaning representation (QDMR) to represent decomposed questions in an intermediate form resembling SQL. TEQUILA (Jia et al., 2018b) used temporal signal (words) based question decomposition to turn natural language questions into sub questions. STAG (Yih et al., 2015) defines core inferential chain and constraints which are analogous to the main and aux defined in our work. However, it's important to note that STAGG doesn't execute explicit decomposition of the lambda in the manner we do.

KBQA

Question: Who was the President during the Dawes Plan?
λ-expression: `λ a. have-org-role-91(h, a, "president") ^ interval(hi, h) ^ type(tp2,"Dawes", "plan") ^ interval(p2i, "Dawes") ^ overlap(hi, p2i)`
Is Answerable from KBQA?: No
KBQA Issue: Auxiliary Fact Missing (temporal information of Dawes Plan)

Missing Fact Extraction

Retrieved textual info: The Dawes Plan as proposed by the Dawes Committee, chaired by Charles G. Dawes was a plan in **1924** that successfully resolved the issue of World War I reparations that Germany had to pay.
Auxiliary temporal fact: 1924
Reformed λ-expression: `λ a. interval(hi, h) ^ have-org-role-91(h, a, "president") ^ interval(p2i, date("dd-mm-1924")) ^ overlap(hi, p2i)`
Answer with reformed λ-expression (from KB): Calvin Coolidge

Figure 4: Illustration of a working example showing the KBQA failure occurring due to missing auxiliary fact that is substituted by temporal fact extraction and finally reforming the lambda expression by hard-coding the missing fact.

Example
Missing auxiliary fact
Question: What was Franklin Roosevelt's position during World War II before pearl harbor? **Ground Truth Answer**: President of the United States **Answered from KB?**: False **Lambda** : lambda a. position-01(p2, "Franklin Roosevelt", a) ∧ interval(p2i, p2) ∧ war(w, "World War II") ∧ interval(wi, w) ∧ interval(bi, "Pearl Harbor") ∧ before(wi, bi) ∧ interval(wi, w) ∧ overlap(p2i, wi) **Aux lambda** : lambda wi. war(w, "World War II") ∧ interval(wi, w) ∧ interval(bi, "Pearl Harbor") ∧ before(wi, bi) ∧ interval(wi, w) **Main lambda** : lambda a. interval(p2i, p2) ∧ position-01(p2, "Franklin Roosevelt", a) **Is Auxiliary answered from KB?**: False **Auxiliary relevant passages** (text extraction): 1) "Japan's attack on Pearl Harbor took place on December 7, 1941. The U.S. military suffered 18 ships damaged or sunk, and 2,400 people were killed. Its most significant consequence was the entrance of the United States into World War II. The US had previously been officially neutral but subsequently entered the Pacific War, the Battle of the Atlantic and the European theatre of war. Following the attack, the US interned 120,000 Japanese Americans, 11,000 German Americans, and 3,000 Italian Americans.", 2) "On the morning of December 7, 1941, the Japanese struck the U.S. naval base at Pearl Harbor with a surprise attack, knocking out the main American battleship fleet and killing 2,403 American servicemen and civilians. Scholars have all rejected the conspiracy thesis that Roosevelt, or any other high government officials, knew in advance about the Japanese attack on Pearl Harbor. The Japanese had kept their secrets closely guarded, and while senior American officials were aware that war was imminent, they did not expect an attack on Pearl Harbor." **Auxiliary answer fact**: "Japan's attack on Pearl Harbor took place on December 7, 1941." **Auxiliary interval**: 1941 **Reformed lambda**: lambda a. interval(p2i, p2) ∧ position-01(p2, "Franklin Roosevelt", a) ∧ interval(wi, date("7-12-1941")) ∧ overlap(p2i, wi) **System Answer**: President of the United States

Table 5: Examples of System Answers

KBQA

Question: Who was Governor of Arkansas when Deewangee was released?
λ-expression: `λ a. have-org-role-91(h, a, "Arkansas", "governor") ^ interval(hi, h) ^ release-01(r, "Deewangee") ^interval(ri, r)^ overlap(hi, ri)`
Is Answerable from KBQA?: No
KBQA Issue: Main Fact Missing (temporal information of answer candidate(s))
Main Candidates from KB: Bill Clinton, Jim Guy Tucker, Mike Huckabee, Mike Beebe
Auxiliary temporal constraint:: 2002 (overlap)

Fact Extraction

Targeted extraction of temporal fact from text: Bill Clinton (1983 - 1992), Jim Guy Tucker (1992, 1996), Mike Huckabee (1996-2007), Mike Beebe (2007 - 2015)
Temporal Reasoning: Candidate Temporal Interval ^ Overlap (2002)
Answer: Mike Huckabee

Figure 5: Illustration of a working example showing the KBQA failure occurring due to missing main fact. Temporal information for each of the extracted main (answer) candidates are extracted from textual resources followed by temporal reasoning to retrieve valid main answer.

Example
Missing main fact
Question: Who was the first man on the moon in 1969? **Ground Truth Answer**: Neil Armstrong **Answered from KB?**: False **Lambda** : argmin(lambda m. man(mu, m, "moon"), lambda m. lambda mi. interval(di, date("dd-mm-1969")) ∧ overlap(mi, di) ∧ interval(mi, mu), 0, 1) **Aux lambda** : - **Main lambda** : lambda m. interval(mi, mu) ∧ man(mu, m, "moon")" **Is Auxiliary answered from KB?**: False **Main relevant passages** (text extraction) : 1) "On July 20, 1969, Armstrong and Apollo 11 Lunar Module LM pilot Buzz Aldrin became the first people to land on the Moon, and the next day they spent two and a half hours outside the Lunar Module Eagle spacecraft while Michael Collins remained in lunar orbit in the Apollo Command Module Columbia. When Armstrong first stepped onto the lunar surface, he famously said: T̈hat's one small step for [a] man, one giant leap for mankind.İt was broadcast live to an estimated 530 million viewers worldwide. Apollo 11 effectively proved US victory in the Space Race, by fulfilling a national goal proposed in 1961 by President John F. Kennedy öf landing a man on the Moon and returning him safely to the Earthb̈efore the end of the decade. Along with Collins and Aldrin, Armstrong was awarded the Presidential Medal of Freedom by President Richard Nixon. President Jimmy Carter presented him with the Congressional Space Medal of Honor in 1978, and Armstrong and his former crewmates received a Congressional Gold Medal in 2009." 2) "Apollo 11 July 162̌01324, 1969 was the spaceflight that first landed humans on the Moon. Commander Neil Armstrong and lunar module pilot Buzz Aldrin formed the American crew that landed the Apollo Lunar Module Eagle on July 20, 1969, at 20:17 UTC. Armstrong became the first person to step onto the lunar surface six hours and 39 minutes later on July 21 at 02:56 UTC; Aldrin joined him 19 minutes later. They spent about two and a quarter hours together outside the spacecraft, and collected 47.5 pounds 21.5 kg of lunar material to bring back to Earth. Command module pilot Michael Collins flew the Command Module Columbia alone in lunar orbit while they were on the Moon's surface. Armstrong and Aldrin spent 21 hours, 36 minutes on the lunar surface, at a site they had named Tranquility Base upon landing, before lifting off to rejoin Columbia in lunar orbit." **Main answer fact**: "On July 20, 1969, Armstrong and Apollo 11 Lunar Module LM pilot Buzz Aldrin became the first people to land on the Moon" **System Answer**: Neil Armstrong, Buzz Aldrin

Table 6: Examples of System Answers

Question	System Path	Ground Answer	System Answer	Comments
Who won best actor when Alfred Junge won best art direction?	Aux Failure (text extraction)	Ronald Colman	Yul Brynner	**Top Retrieved Fact**: In addition to the same producer, director and star, the first two films in the trilogy had the same cinematographer F. A. Freddie Young, composer Mikl Rufzsa, art director Alfred Junge and costume designer Roger Furse. The costumes for this film were executed by Elizabeth Haffenden. In 1955, she would take over from Furse as costume designer for the final film in the trilogy, Quentin Durward. Alfred Junge remained as art director. **Issue**: The issue here is incorrect Auxiliary fact extraction by the ectraction pipeline. As can be seen, an irrelevant passage has been detected as the fact containing Auxiliary interval. Hence the extracted Auxiliary fact was 1957 as opposed to the correct date 1947. As a result the reformed lambda consisted of erroneous temporal constraint due to to which the KBQA pipeline returned an incorrect answer as expected.
Which team did Wayne Rooney play for before joining Manchester United?	Aux Failure (text extraction)	Everton F.C.	England	**Top Retrieved Fact**: Wayne Mark Rooney born 24 October 1985 is an English professional football manager and former player. He is the manager of EFL Championship club Derby County, for whom he previously served as interim player-manager. He spent much of his playing career as a forward while also being used in various midfield roles. Widely considered to be one of the best players of his generation, Rooney is the record goalscorer for both the England national team and Manchester United.Rooney joined the Everton youth team at the age of nine and made his professional debut for the club in 2002 at the age of 16. He spent two seasons at the Merseyside club, before moving to Manchester United for 325.6 million in the 2004 summer transfer window where he won 16 trophies and became the only English player, alongside teammate Michael Carrick, to win the Premier League, FA Cup, UEFA Champions League, League Cup, UEFA Europa League, and FIFA Club World Cup.

				Issue: The Auxiliary fact in this case is correctly detected: 2004. However when the reformed lambda with the auxiliary fact hard-coded was evaluated on the KB, an adjacent entity also complying with the constraints was picked as the final answer. In this case Wayne Rooney was also a player of England before (as well as after) he joined Manchester United. However the answer expected was his previous club: Everton F.C. This is due to an inherent issue with the KBQA pipeline.
When did the industrial revolution in Europe began?	Main Failure (text extraction)	1791	late 18th century	**Top Retrieved Fact**: Industrial growth Capitalism has been dominant in the Western world since the end of feudalism. From Britain, it gradually spread throughout Europe. The Industrial Revolution started in Europe, specifically the United Kingdom, in the late 18th century, and the 19th century saw Western Europe industrialise. Economies were disrupted by World War I but by the beginning of World War II they had recovered and were having to compete with the growing economic strength of the United States. World War II, again, damaged much of Europe's industries. **Issue**: The textual resource does not contain the exact date mention of the missing temporal fact. So the targeted extraction pipeline retrieves the most appropriate sequence of tokens "late 18th century" from the text as the temporal fact.
Who is the Governor of Arizona in 2009?	Main Failure (text extraction)	Jan Brewer & Janet Napolitano	Jan Brewer	**Top Retrieved Fact**: Governor Jan Brewer assumed office in 2009 after Janet Napolitano had her nomination by Barack Obama for Secretary of Homeland Security confirmed by the United States Senate. Arizona has had four female governors, more than any other state. **Issue**: The top retrieved relevant fact had explicit mention of Jan Brewer assuming office in 2009. However, it lacks explicit mention of Jane Napolitano's end date. The end date being 2009 had to be implicitly reasoned out which is not trivial for the text based fact extractor. Hence the overlapping dates in office (2009) of the two Governors was missed out.

Table 7: Detailed error analysis of a few incorrectly answered questions