TimelineQA: A Benchmark for Question Answering over Timelines

Wang-Chiew Tan, Jane Dwivedi-Yu, Yuliang Li,
Lambert Mathias, Marzieh Saeidi*, Jing Nathan Yan+, and Alon Y. Halevy
Meta Cornell University+
{wangchiew,janeyu,yuliangli,lambert,ayh}@meta.com
marzieh.saeidi@googlemail.com* jy858@cornell.edu+

Abstract

Lifelogs are descriptions of experiences that a person had during their life. Lifelogs are created by fusing data from the multitude of digital services, such as online photos, maps, shopping and content streaming services. Question answering over lifelogs can offer personal assistants a critical resource when they try to provide advice in context. However, obtaining answers to questions over lifelogs is beyond the current state of the art of question answering techniques for a variety of reasons, the most pronounced of which is that lifelogs combine free text with some degree of structure such as temporal and geographical information.

We create and publicly release TimelineQA[1], a benchmark for accelerating progress on querying lifelogs. TimelineQA generates lifelogs of imaginary people. The episodes in the lifelog range from major life episodes such as high school graduation to those that occur on a daily basis such as going for a run. We describe a set of experiments on TimelineQA with several state-of-the-art QA models. Our experiments reveal that for atomic queries, an extractive QA system significantly out-performs a state-of-the-art retrieval-augmented QA system. For multi-hop queries involving aggregates, we show that the best result is obtained with a state-of-the-art table QA technique, assuming the ground truth set of episodes for deriving the answer is available.

1 Introduction

The promise of augmented reality (AR) glasses has renewed interest in building personal assistants that are capable of being with us at all times of the day. In order for such assistants to be useful, they need to have detailed knowledge about the user, including their past experiences, preferences, habits and goals in the spirit of systems such as Memex (Bush, 1945) and MyLifeBits (Gemmell et al., 2006). A lot of that knowledge already is implicitly present in the digital data that people generate by interacting with a myriad of online services such as photos, maps, health apps, shopping and content streaming. A lifelog is a private and secure database that contains a set of episodes from the user's past that are gleaned from these data sources and in the future from smart glasses. The lifelog is completely under the control of the user, and only they can decide if and when to share fragments of it as they see beneficial. For example, they may share past dining experiences with an assistant when trying to choose an item from a menu, or past movie preferences with a friend when trying to decide which movie to watch together.

In addition to issues relating to privacy, lifelogs raise two main classes of challenges. The first is to infer meaningful episodes from the raw data. For example, such an inference module would take as input a set of photos and output an episode such as *visited Venice for 7 days*, or *celebrated a birthday party with friends*. The second challenge, which is the subject of this paper is to answer questions over the lifelog, such as *when did I go to Tokyo*, *what did I eat on my second night in Paris*, or *how many times did I go to the dentist last year*.

Question answering is challenging because the data contains a combination of text and structure. The episodes themselves are described as text (and may also contain images and video), but each episode is associated with a time and location. For example, in order to answer a query such as *where did I take my mom when she visited Seattle*, the system first needs to figure out when mom visited Seattle and then look for episodes within that time interval. Other questions may require counting or reasoning over sets of episodes, similar to challenges raised in (Thorne et al., 2021).

This paper describes TimelineQA, a benchmark for querying lifelogs. The benchmark includes a

[1]Code and data available at https://github.com/facebookresearch/TimelineQA

Findings of the Association for Computational Linguistics: ACL 2023, pages 77–91

generator that produces lifelogs for imaginary people with different personas (e.g., age, gender, education and family status). Each lifelog includes episodes drawn from a variety of activities, ranging from significant activities (e.g., going on a trip or getting married) to more daily activities (e.g., cooking dinner or going to the doctor). For each lifelog, the benchmark creates a set of question/answer pairs, specified in English.

Naturally, real lifelogs are complex and extremely diverse and are challenging to generate synthetically. Our main contribution is a benchmark for QA systems over lifelog data of different sizes. The goal of the benchmark is not to represent people's lives in their full complexity or diversity, but to offer a sufficiently rich set of lifelogs that already exposes the challenges involved in question answering (QA). We show some snippets of our generated lifelogs Section 4.1. As our QA techniques improve, the benchmark will be enriched to include more real life complexities.

We describe a set of experiments demonstrating that current SOTA QA techniques fall short of adequate performance on lifelogs. We experimented with extractive (Karpukhin et al., 2020) and RAG (Lewis et al., 2020b) QA systems on atomic queries. Somewhat surprisingly, even after fine-tuning, the generative RAG QA system still lags behind the extractive system for question-answering. In addition, we ran a Tapex (Liu et al., 2022), a table QA model and BART (Lewis et al., 2020a) for complex queries over TimelineQA. Our experiments reveal that the best performing system, Tapex, only scores 59.0%, assuming that the subset of episodes needed to compute the answer is known.

2 Related work

The idea of creating a repository that captures all the knowledge about a person's life dates back to Vannevar Bush's vision of the Memex System (Bush, 1945). Gemmell et al. (2006) describes the MyLifeBits System that implemented the vision with the technology available in the late 1990's, and they used simple keyword search with the help of an SQL database to query its contents. Alam et al. (2022) describes a more recent project on creating lifelogs, and the Solid Project (Mansour et al. (2016)) takes an even more radical approach, suggesting that all of the user's data be stored in a *data pod* and that applications be redesigned to access it from the pod. Since the early years, the promise of personal agents has increased since data storage has become cheaper and ubiquitous, we anyway generate many more digital breadcrumbs with services we use on a daily basis, and AI techniques have become much better at analyzing text and image content.

The design of our benchmark was inspired by the Clevr benchmark for evaluating visual query answering systems (Johnson et al., 2017). Like Clevr, we design a space of possible questions that can be asked and then generate synthetic datasets where we know the answer to each questions posed.

There is a rich body of work on query answering. The ones closest to our work are on multi-hop queries (Mavi et al., 2022) and neural databases (Thorne et al., 2021). In addition to queries that can be answered from a single episode in a lifelog, TimelineQA includes more complex queries that require combining information from multiple episodes in a lifelog. This is similar to work on QA over long documents (Khashabi et al., 2018). However, the length of a lifelog is typically much greater than any existing benchmark or experimental dataset to the best of our knowledge. A typical lifelog can contain between 15M to 78M entries on average, where each entry contains about 8–9 tokens on average. Furthermore, TimelineQA queries can also contain aggregates (e.g., max, sum, average). Neural databases considers the problem of answering aggregate queries over text data of arbitrary size, but it does not address the temporal aspects that are critical to queries over lifelogs.

3 Lifelogs

A lifelog includes any kind of experience that a user recorded digitally (see Figure 1). We model experiences as *episodes* in the lifelog, and every episode is associated with a start/end time and start/end location, if those are known. Episodes are captured via photos or videos, smart watches (e.g., exercise and sleep tracking), mapping services (e.g., routes and visits), documents that have been explicitly stored (e.g., passport), or notes that the user takes describing their subjective experiences. A lifelog is completely private and accessible only to the user. She can share slices of her lifelog if and when there's value in doing so (e.g., getting better service from a sales person).

Episodes are typically activities that the user was involved in, such as celebrating a holiday, going on a trip, going for a run or a bike ride, physi-

cal therapy, seeing fireworks or watching a movie. Episodes in the lifelog can either be done by the owner of the lifelog or by someone in their family or circle of acquaintances, e.g., mom moving to Seattle, sister getting married, having one's air-conditioning fixed, or being told something by a friend. In addition to time and location, episodes may have attributes, such as who was involved, the distance and speed of a run, or the name of a product that was purchased. Some of these attributes may be modeled explicitly in the lifelog if they're easy to extract, and others may remain in the raw text or image and found at query time.

Lifelogs are meant to be built with as little friction as possible from users. Hence, as shown in Figure 1, the data is imported from the external services into the lifelog as raw data. Some raw data already describes episodes (e.g., purchase or content consumption episodes). Other episodes are then inferred by analyzing and fusing multiple pieces of raw data (e.g., a trip, or a meal with friends). Of course, the inference step is a best-effort one, which means that some questions may still be impossible to answer and in some cases the QA system will point the user back to data that contains the answer (e.g., what did we eat on my daughter's birthday). Questions are answered based on the text and structured data describing all the episodes in the lifelog.

Our work concerns question answering after the inference of episodes has been done. Hence, formally a lifelog is a collection of episodes, each one associated with their start/end time and location: time-and-space boxed episodes. Each episode contains some text and possibly pointers to external raw data. Note that episodes can be nested within other episodes.

3.1 A classification of questions

To understand the breadth and types of questions users may want to ask of lifelogs, we crowdsourced the task of writing down questions over their potential lifelogs to 7 people. We also asked for the categories of their questions. We obtained a total of about 600 questions. We analyzed the categories and organized them into 13 topics (e.g., life milestones, travel, daily activities) as described in Table 8 in the appendix. After this, we asked (again) each contributor to write a few questions they would ask on each of the 13 topics.

Based on a qualitative analysis of these questions, we observe that the queries can be classfied as follows. We use the term query and question interchangeably.

Atomic questions: An atomic query, which is the most common type, asks for some attribute of an episode. Examples include:

- When did my mom have a knee operation?
- What's the name of the company that repaired my A/C?
- What's the name of my daughter's first-grade teacher?

An atomic query is one that can be answered by a *single* episode. The answer to an atomic query can either be directly explicit in the text of the episode (e.g., when), or requires inference from the text (e.g., who fixed the A/C). For example, if an episode describes "*08/01/2022: John was here. He fixed the AC this morning.*", then the respective answers to the questions are "08/01/2022" and "John". In principle, an answer may also be a link to a photo that may contain the information asked by the user, though TimelineQA is currently limited to questions that can be answered after the inference of episodes is done. Finally, some answers may require a bit of derivation. For example, when is my sister's 40th birthday could be derived from the episode describing her birth.

Complex queries – multi-hop: The answer to a multi-hop query is formed by combining data from multiple episodes. Hence, oftentimes, multi-hop queries require identifying a set of episodes in the timeline. For example, *Where did we eat great Indian food on our way to Vancouver?* would require identifying episodes involving the trek to Vancouver and eating Indian food. Other examples of multi-hope queries are:

- What places did I visit when my mom came to visit Seattle?
- Show me photos of the car damage I had after the accident

Complex queries – aggregates: These questions (known as *aggregation queries* in SQL) consider a set of episodes and compute an aggregate on them. For example:

- How many times did I visit the dentist this year?
- How many miles did I bike this year?

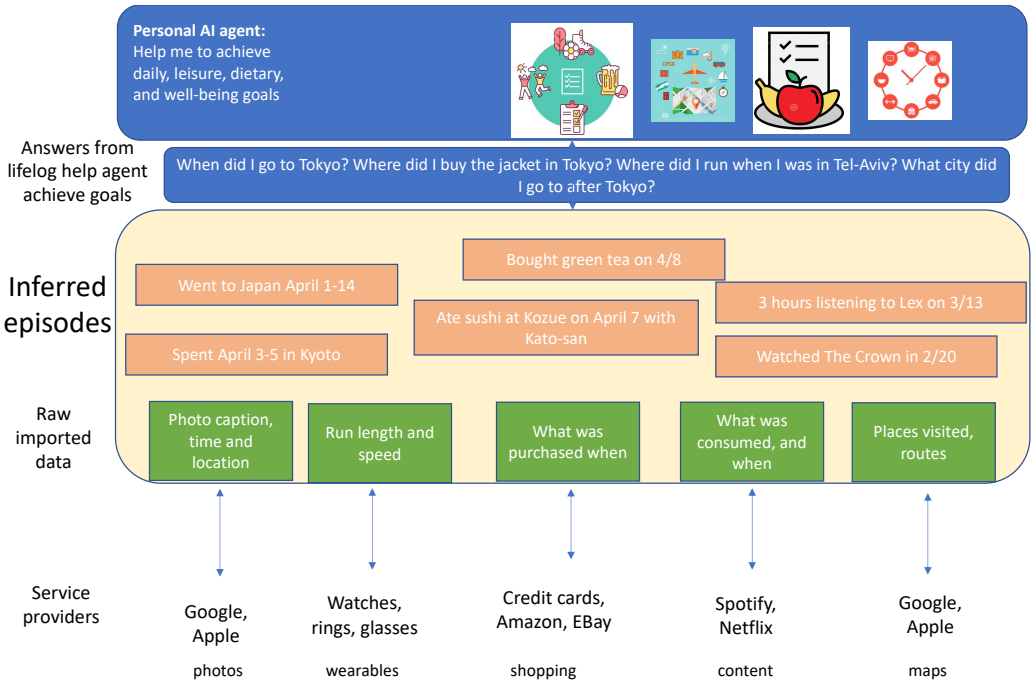

Figure 1: Lifelogs import meta-data from a set of external services. A set of inference models deduces higher level episodes from the raw imported data. Episodes have a start and end time, and often a location. Question answering uses the raw and the deduced data.

In some cases, the aggregation may be combined with another condition, such as *How many calories did I burn on my last two rides?* or *When did I last ride 40 miles or more in a day?*

Temporal queries: Because of the nature of personal timelines, many of the questions that arise are temporal ones. In addition to atomic and complex queries, we identified temporal queries that may be atomic or complex. Examples of atomic queries that are temporal are those whose answer is the time of an episode, such as "*When did I pay my car insurance?*" In general, temporal questions may require more sophisticated reasoning about time, such as finding the length of a life event or the time between episodes, e.g., *How long was my break between leaving my last job and starting my current job?* or reasoning about the sequence of occurrence "*Did I go to Spain before Italy?*". In our crowdsourced query collection, temporal queries were mostly atomic *when* queries or implicit subgoals of more complex queries, e.g., "*when was the last time I visited the dentist?*"

3.2 The goals of the benchmark

The above classification of questions highlights some of the challenges that will arise in query answering over lifelogs. The first challenge is typical for query answering—the disparity between the terms that are used in the query versus the language used in the lifelog itself. For example, a user might ask when they had a drink with a particular friend, while the lifelog may say that they went to a bar before dinner. In the lifelog context the challenge can also require multi-modal reasoning, because the only item in the lifelog might be a photo from a bar. As another example, users may refer to more aggregate terms than what's in the lifelog. For example, the user may ask how much they spent on utilities last month, while the lifelog has individual utility bills, but the system may not be aware of which bills are considered utility bills. We expect that query answering over lifelogs will benefit from advances in the broader field of query answering and therefore this is not a focal point of our benchmark.

The second set of challenges involves the interplay between the structure that the lifelog supports and the linguistic reasoning. For example, the lifelog may store the duration of every exercise you made, but answering the query on how long did you exercise every day for the past month is more challenging. Another complex example is in the context of multi-hop questions. If a user asks when was the first time she traveled to Tokyo, the system needs to find all instances of the user's travel to Tokyo and then return the first one. Reasoning

about such temporal relations is an area of weakness for QA algorithms today. This aspect of query answering is critical to lifelogs and therefore we design our benchmark to evaluate these challenges.

Specifically, we would like our benchmark to push the limits on the interaction between structure and language in query answering. To that end, our benchmark is designed to be able to vary a few variables, including the complexity of the questions, the size and contents of the lifelogs, and the types of data that are in the lifelog, including the complexity of life episodes the user has, how verbose the user is (i.e., do they log only their major experiences or also many minutiae episodes).

4 Creating lifelogs in TimelineQA

Since we believe that TimelineQA is the first in a series of lifelog benchmarks, we explain here in some detail how it is built. A lifelog is a set of episodes in the life of a person. Our goal is to create lifelogs that contain a good range of experiences that a person may have in life and sufficient to begin benchmarking the performance of QA systems on lifelogs. To collect a broad set of typical episodes, we started with a detailed set of episode categories described in Coelition[2], a site that *provides technology and expert advice for data collected about people on the Internet of Things*, and distilled them into the categories shown in Table 1. The categories of episodes range from life episodes (e.g., being born, going to college), episodes that happen a few times a year (e.g., trips) to those that happen on a weekly or daily basis (e.g., meeting friends or cooking). The timescales and examples in Table 1 coincided broadly with the types and categories of questions we obtained from our crowdsourced task. See Table 8 in Appendix A.1.

Creating a persona The process of building a lifelog begins with creating a persona which includes the skeletal details of a person's life, including when and where they were born, their gender, their educational and professional history, their family members and some of their preferences and hobbies. We first generate a birthdate, which must be between 18-75 years old at the time of generation. We randomly select a gender and a name from a dictionary of names. We then proceed to create their educational and professional history, family members, preferences and hobbies. These are generated via a model that depends on several probability distributions of episodes. We note that while the personas we create are quite varied, we do not claim that they represent a diversity in any social sense. The diversity we do build in is limited: age, gender, locations, professions. Clearly, in order to achieve robust query answering on lifelogs we need to consider many other kinds of diversity (culture, non-typical episodes and scenarios), but we believe that the benchmark as is already poses many important challenges.

Creating episodes Once a persona is created, we begin creating episodes starting from the day the person was 18 years old to the present year. We first create episodes in the lifelog for life events, such as birth, educational phases, starting and ending jobs, marriage(s) and having children. We then proceed to generate episodes at different levels of granularity based on the timescales (annual, monthly, weekly, daily) as shown in Table 1. For example, for annual episode types, we create annual health checkups episodes and yearly trips. For monthly episodes, we generate pet grooming episodes and some examples of weekly and daily episodes are baking/cooking, grocery shopping, catching up with friends or news. These episodes are generated as described in Table 1. These episodes are generated based on a predefined probability distribution which can be modified.

Some of the episodes we create are *super episodes*, which involves sub-episodes that depict events of finer granularity. For example, a multi-day travel or trip episode will be broken down to movements between different destinations, and the itinerary for every single day and special episodes that happened in each day. The descriptions of episodes are generated by instantiating templates that we specify. Every episode is associated with a set of alternative templates and a template is randomly picked and instantiated for a given episode to be created. Since the templates are fixed, the descriptions generated may not offer the variety in descriptions we expect from a general population. We are in the process of incorporating the use of language models to generate episode descriptions as yet another alternative. However, it is interesting to understand what limitations on QA such a benchmark already exposes with templated descriptions.

More variations in the episode activities can be added to the lifelog generator to more closely reflect the categories we find in the Coelition and also what we crowdsourced (Table 8). We leave this for

[2]https://coelition.org/business/resources/visualising-life/

Time scale	Examples
Lifetime	birth, educational milestones, marriage, divorce, jobs & relocation
Annual	travel, medical and dental checkups
Monthly	pet care (e.g., grooming)
Weekly	baking, cooking, dating, hobbies, buying groceries
Daily	eating meals, talking with friends, exercising, consuming content (books, movies)

Table 1: Types of episodes in TimelineQA. Episodes are divided into several time scales. The generator creates episodes in successive time scales, starting from lifetime events.

future work.

Consistency through constraints: To ensure more consistency, we keep track of the attributes of every single day in one's life. For example, the probability of certain episodes can change drastically if a person is on a trip or in the process of getting married. In TimelineQA, constraints can be specified to prevent inconsistencies from occurring. For example, since it is much less likely that one bakes or has an annual dental checkup while traveling, we can explicitly state that these episodes should be mutually exclusive in TimelineQA. If an episode is to be created on a certain day, TimelineQA checks that it is mutually exclusive to any existing episode applicable to that day before creating the new episode.

Generating questions and answers: Every lifelog, $\mathcal{D}$, in TimelineQA is associated with a set of question/answer pairs (Q, A), where Q is a natural language question over $\mathcal{D}$ and A is the correct answer to it. In order to ensure that we can create a variety of questions that are meaningful on a particular lifelog $\mathcal{D}$ and that we know the correct answers to them, the process of creating begins by creating a logical representation of the episodes in the lifelog and of the questions and the answers, and then turning them into natural language. The natural language of the questions and answers are created by instantiating a few templates for every episode type. Because we use templates, TimelineQA clearly lacks the richness of linguistic variation, but as noted previously, dealing with linguistic variation is not the focus of this benchmark.

We generate questions and answers for each lifelog in two steps: atomic questions and complex questions. Since atomic questions are ones whose answer is contained in a single episode in the lifelog, we can create them at the same time the episode is created. For example, if the episode is *I went to a Japanese restaurant with Sarah on October 7th and ate sushi*, then we would generate questions of the form: *when did I have Japanese food?*, *when did I meet Sarah?*, and *where did I eat on October 7th?* For each single episode, we create *what*, *where*, *when* and *who* questions as appropriate along with the corresponding answers.

Complex questions are ones that either rely on a set of facts in the lifelog, such as, *how many times did I go to London?* and *where did I spend the first night in Tokyo?* or require combining multiple facts as in multi-hop questions such as, *which restaurants did I go to during my trip to New York?* To create such question/answer pairs easily, we create a database of the logical representation of all the episodes in the lifelog. We then consider a set of query templates and check whether the template can be instantiated on that database. Examples of templates we consider are:

- How many times did I X?
- When was the first/last time I X?
- Did I go to X before I went to Y?
- How many times did I do X when I was at Y?

Since we have all the episodes, we can compute the answers to these questions correctly.

Size and density: Lifelogs of different sizes can be created with TimelineQA. The user specifies a year and duration parameter, and this will determine the length of the lifelog to generate. For example, if the year is 2023 and the duration is 5, then 5 years of episodes from 2018 to 2023, including lifetime episodes, will be created. Lifetime episodes such as birth and college education may occur outside those 5 years.

The user can also specify the density (sparse, medium, or dense) of episodes to generate in the lifelog. The variations in density are used to mimic that different users log their life events at different frequencies. For example, if the generator is called with the "sparse" parameter, then the probabilities of generating daily/weekly/monthly episodes will be much lower than the case when the generator is called with the "dense" parameter.

4.1 Example lifelog

An example snippet of our generated lifelog and sample question and answer pairs are given below.

2010/01/08, I had lunch. I ate Indian food.
2010/01/09, I had cereals for breakfast with Hazel, Rylee, Piper, Nora, Avery, Eva, Nevaeh, Claire, Lydia, Olivia, Layla, Kinsley.
2010/01/09, I had lunch. I ate sushi.
2010/01/09, I had chinese food for dinner with Kayden, Carter.
2010/01/09, I spent 21 minutes on social media today.
2010/01/10, I did some hiking on 2010/01/10.
2010/01/10, I ate pasta for dinner.
2010/01/11, I talked to Nevaeh, Piper, Olivia, Eva for 37 minutes late in the evening.
2010/01/12, I did some swimming on 2010/01/12.
2010/01/12, I talked to Nora for 47 minutes in the morning.
:

4.1.1 Example question-answer pairs

Atomic QA pairs: These QA pairs are created as each episode in the timeline is generated. Based on the episode that is generated, a question is instantiated from a set of templates and the answer to the question is extracted from the generated episode. Some examples are shown below.

Q: What did I eat with Kayden and Carter on 2010/01/09?
A: I ate chinese food with Kayden and Carter.
Q: How long did I talk to Nora on 2010/01/12?
A: I talked to Nora for 47 minutes.

Complex QA pairs: Using our query templates, we created 42 complex questions in our benchmark for the subset of categories we have implemented in our timeline generator. The answers are computed by applying external algorithms (e.g., SQL queries) over the timeline.

Q: How much time on average did I spend on reading the news each day?
A: On average, you spent 32 minutes reading the news each day.
Q: How many times did I take my kids to an optician in 2010?
A: You took your kids 2 times to an optician.

5 Baselines and experimental results

5.1 Datasets

Table 2 summarizes the lifelogs we generate for TimelineQA. The dataset consists of 128M lifelog entries in total for all 3 types of densities (sparse, medium, and dense). Each entry has an average of 8.4 tokens. TimelineQA covers 25 categories of events ranging from daily chat to lifetime events such as college graduation. Different categories occur at various frequencies and describe events in heterogeneous formats at various lengths. See Table 9 in the appendix for the full breakdown. For our QA experiment, we uniformly sample 40 lifelogs for each density (120 in total) as the hold-out test set.

For each lifelog, we construct test samples for both *atomic QA* and *multi-hop QA*. Atomic QA refers to the *what, where, when, yes/no* types of questions where the answer requires reasoning (or plain extraction) over a valid span of a single input episode. We construct 5,000 such questions for each lifelog (600k in total) as the hold-out test set. Multi-hop QA refers to the complex type of questions that involve selection and aggregation.

Table 2: Statistics of 1,000 sparse, medium, and dense lifelogs. See Table 9 for the breakdown on the 25 event categories.

Datasets	#Logs	#Entries	Avg. #Tokens
sparse	1,000	14,941,703	8.51
medium	1,000	34,522,030	8.12
dense	1,000	78,559,743	8.50
all	3,000	128,023,476	8.40

Table 3: Statistics of multi-hop QA tasks.

	#Logs	#QA's	#Evidence	AVG	%Truncated
Train	240	8,586	10M	1,174.8	20.44%
Valid	120	4,302	5M	1,216.6	20.99%
Test	120	4,284	5M	1,169.8	20.40%

Table 3 shows the statistics of the multi-hop QA datasets. In addition to the test set, we constructed a disjoint training and validation set similarly (240 and 120 logs, respectively) for our fine-tuning experiment. Each lifelog contains $\sim$35 multi-hop queries. Each query also comes with a set of ground-truth evidence records, which are all the episodes for deriving the correct answers. Each question has an average of $>$1k evidence records, which together are beyond the typical max length of 512/1024 tokens of transformer-based LMs. Indeed, even as we set the max input length to 1024, $\sim$20% of the input episodes are truncated.

5.2 Atomic QA

We consider the following QA implementations for atomic QA:

RAG (Lewis et al., 2020b). This is a retrieval-augmented generative QA system, where we first retrieve some documents based on the query, and then condition the answer generator based on these retrieved documents and the query. We replace the Wikipedia based memory in the original RAG with episodes. We use the original *RAG-Token* model released checkpoints.[3]

ExtractiveQA (Karpukhin et al., 2020). The key difference from RAG, is that the answering system is a span-based extractive model, extracting the answer from a given context. Specifically, the

[3]See the implementation in https://haystack.deepset.ai/tutorials/07_rag_generator

Table 4: Atomic QA Results comparing extractive and RAG based QA under 3 conditions for the retriever: Zero-shot (ZS), fine-tuned (FT), and oracle (OR).

Pipeline	Retriever	Exact Match	F1
Extractive	FT	82.6	93.8
Extractive	OR	83.3	94.8
Extractive	ZS	24.1	47.3
RAG	FT	40.3	57.5
RAG	OR	73.7	84.4
RAG	ZS	8.4	32.9

Table 5: Zeroshot (ZS) / Finetuned (FT) model performance on multi-hop QA over 120 TimelineQA lifelogs.

Retriever		Oracle		FT-retriever		ZS-retriever	
Reader	size	ZS	FT	ZS	FT	ZS	FT
Tapex-base	140M	2.8	57.7	2.7	30.8	2.7	30.7
Tapex-large	400M	**6.5**	**59.0**	6.5	32.7	6.5	33.0
Bart-base	140M	0.0	54.4	0.0	28.7	0.0	29.1
Bart-large	400M	0.0	47.0	0.0	21.9	0.0	25.2

reader is a RoBERTa (Liu et al., 2019) model fine-tuned on SQuAD (Rajpurkar et al., 2018).[4]

In both cases, we encode all the episodes using a dense passage retriever, and use FAISS to return the top-5 episodes. The retrieved documents are then fed into the answering component, and we get the top-1 answer. We consider 3 different setups for the retriever: Zero-shot (ZS) using the pre-trained checkpoints, fine-tuned on question-episode pairs from the lifelogs (FT), and oracle retrieval (OR) where we use the ground-truth episode associated with the question.

From the results in Table 4, we observe that extractive QA performs significantly better than generative QA, which is to be expected, given the benchmark construction, where the answers are always a valid span in the input for atomic queries. Furthermore, by fine-tuning the retrievers on the episodic data, we get a significant boost in performance for both extractive and rag setups, indicating that the QA systems do not generalize well to episodic data, and that improving retrieval is crucial to getting good performance from these models, particularly for RAG. After fine-tuning, the generative model performance still lags behind the extractive setup.

5.3 Multi-hop QA

Given the task's nature of aggregating structured data, we consider a baseline based on TableQA (Badaro et al., 2023). In short, a table QA model answers questions by taking as input a relational table (e.g., records of dental visits) and a NL query.

Table 6: Breakdown of Tapex-large (finetuned) with oracle retriever on question types and sizes of evidence sets.

type	accuracy	total	#evidence	accuracy	total
average	11.1	360	[0, 10]	85.1	1,949
count	75.9	1,776	(10, 100]	52.5	1,275
argmax	47.2	1,668	(100, 1000]	19.2	689
list	62.7	480	>1000	4.3	371

We constructed the tables for table QA using an information extraction pipeline over the episodes as they are generated. By exploiting the topics (e.g., medical care, chat, exercise) which are known to the generation pipeline, we define a fixed schema for each topic. For example, we use the schema `(date, place, medical_care_type, person)` for all types of medical care episodes, and run named-entity recognition to extract the tuple from each episode. For example, the record tt (2019/03/23, annual vision checkup, university hospital, Jack) will be created from the input "*I took Jack for his/her for an annual vision checkup on 2019/03/23 at the university hospital.* We then form the "annual_medical_care" table using all quadruples extracted from episodes under the same topic. This simple pipeline works very well (near perfect) for by exploiting the generation pipeline. For real-life lifelog data, additional challenges such as episode construction, topic/attribute discovery, and schema reconciliation, are beyond our current scope.

Due to the large size of the life logs that cannot fit in the max length of LMs, the TableQA baseline also leverages a dense *retriever* for retrieving relevant records and constructing a concise table representation of the entries. We then apply the TableQA model as the *reader* to produce the final answer via selection, aggregates, etc.

More precisely, for multi-hop queries, given a question q over a set of life logs $L = \{l_1, \ldots, l_n\}$, the retriever is a model M_{ret} where $L_{\mathsf{ret}} = M_{\mathsf{ret}}(q, L) \subseteq L$ is the retrieved subset. We then process L_{ret} into table format via NER and pattern matching to convert L_{ret} into its table representation T_{ret}. Finally, the TableQA model M_{read} returns the answer $M_{\mathsf{read}}(q, T_{\mathsf{ret}})$.

We also evaluate variants of the Tapex (Liu et al., 2022) model as baselines. Tapex achieved the state-of-the-art performance of TableQA by pre-training a seq-to-seq LM on table datasets to mimic the behavior of a SQL engine. We also compare the

[4]Details available at https://huggingface.co/deepset/roberta-base-squad2

Table 7: Example correct and incorrect model predictions for multi-hop questions.

Questions	groundtruth	prediction	Notes
How long do I spend on average each day talking to my friends?	84.05	83.94	The question requires aggregating a total of 74k records
In what year did I buy facial wash the most?	2006	2015	This question only needs to deal with 47 records, but requires complex arithmetic reasoning (count+compare)
How many times did I have tacos for dinner in September 2019?	5	5	The model correctly captures simple counting (5 evidence records)
Which places in New York, US did I visit with Sofia?	['Central Park', ...]	['Central Park', ...]	The model correctly selects the 7 relevant locations from the input table

performance of Tapex with BART (Lewis et al., 2020a), which has the same architecture as Tapex but without training on tabular data. For both models, we evaluate using the denotation accuracy as in standard TableQA tasks (Zhong et al., 2017). We evaluate each model under both the zero-shot setting and with fine-tuning on the training sets. We also test InstructGPT as a baseline large LM, but leave the full result in Table 10 in the appendix due to limited space.

Similar to atomic QA, we evaluate each model under 3 settings of retrievers. We first assume an *oracle* retriever which has access to the ground truth set of evidence to construct the input table. A *zero-shot* retriever uses a set of user-defined patterns such as *"I talked to X for Y minutes"* to find matching episodes (the same set of rules for converting episodes to table records). We uniformly sample episodes up to the max length of the LM. A *fine-tuned* retriever trains a dense retriever model (Reimers and Gurevych, 2019) from the training set and returns episodes closest to the question's dense embedding.

Table 5 summarizes the results. Overall, the 400M-parameter Tapex model achieves the best result with fine-tuning and the oracle retriever. The 59% accuracy is also close to the Tapex's performance on the WikiTableQuestions benchmark (Liu et al., 2022). However, its performance greatly reduces (1) under the zero-shot setting (6.5%) or (2) with a non-oracle retriever (33%). Tapex generally outperforms its counterpart BART, which indicates the importance of understanding structured data and aggregation for the multi-hop tasks. We also notice that fine-tuning the retriever generally does not improve the QA performance. This can be due to the hard requirement of retrieving the exact evidence set to correctly answer certain questions like count and average.

5.4 Error analysis

Table 6 shows the breakdown of Tapex-large's fine-tuning performance with a perfect retriever. Among the 4 types of questions, argmax and average have the worst performance, likely because they require arithmetic reasoning. We also observe that the model accuracy decreases significantly (from 85.1% to 4.3%) as the number of evidence records grows, which indicates the hardness of dealing with large input tables. Table 7 shows examples of (in)correct predictions of the model.

6 Conclusions

We presented TimelineQA, a benchmark for generating lifelogs of imaginary people. Our experiments, with state-of-the-art QA models, showed that there is still significant room for improving QA over lifelog data. Specifically, while extractive systems can achieve impressive performance on TimelineQA for atomic queries, the best performing QA system for multi-hop queries scores only 59.0% in the perfect setting where the ground truth set of episodes are available.

We view the current state of TimelineQA as a first version that will be enhanced in several ways as the QA technology improves. In future enhancements the episodes can be made more realistic and varied to also include events such as driving one's children to practices, or car breakdowns, to more unexpected events such as experiencing an earthquake etc. In addition, episodes can be enhanced to include different modalities, such as photos or videos of the episodes and more complicated queries can be included such as "*How many times did I swim in the month before I traveled to Machu Picchu?*". Ideally, with appropriate obfuscations to preserve privacy, a future version can mirror precisely the lifelogs of real people.

7 Limitations and Ethical Considerations

There are several perspectives from which we need to consider the ethical considerations of this work.

Privacy: Lifelogs are personal data and should only be used and shared given user authorization. The lifelogs presented here are *fictitious* and do not reveal the personal information of any individual. No personal data is used to create this benchmark. This work is intended to unlock development in the creation, maintenance, querying and usage of lifelogs, and additional work will certainly be needed to ensure that they are secure and being meaningfully and responsibly used.

Comprehensiveness and diversity: We recognize that the lifelogs generated in this work are far from representing the full range of human experiences. While we strived to make the lifelogs complex enough to benchmark and compare current state-of-the-art, these lifelogs would not be considered diverse in the sense that a social scientist would note, and are likely biased by the life experiences of its creators. We encourage future work in creating lifelogs that are more inclusive and faithful to all walks of life. This includes further work in making lifelogs that are more diverse in terms of life experiences, personas, time scales, and queries as well as more granular and complex in detail. The strength of the benchmark is in identifying *patterns* of questions on lifelogs rather than the specific events described in them.

Inferring episodes: TimelineQA is a collection of time-and-space boxed episodes, and not the raw data itself from which the episodes are inferred (e.g., a wedding photo, or video snippet from smart glasses). Naturally, more research would need to be devoted to understanding how to extract important information in natural language and infer episodic events from this raw data before performing question answering. As mentioned previously, this also involves sometimes grappling with the linguistic variation amongst the language used in the episode description and the query itself.

Intended use: We clarify that the benchmark should not be used to train models for making key decisions that will impact people's lives (e.g., job matching, insurance approvals or building personal assistants). The intended use of TimelineQA is as a benchmark to reveal potential limitations of QA systems over lifelog data. Even if the benchmark is determined to be sufficiently comprehensive, a detailed study should be conducted to understand the potential representational harms of using TimelineQA before using it for training models. Conceivably, TimelineQA can also facilitate research in evaluating the biases of QA systems by creating counterfactual pairs in the dataset: two timelines which are exactly the same, but differ by the demographic group or a specific life event (e.g., having dropped out of college or committed a crime). The QA system can then be systematically probed for differences in performance between the two timelines.

References

Naushad Alam, Yvette Graham, and Cathal Gurrin. 2022. Memento 2.0: An improved lifelog search engine for lsc'22. In *LSC@ICMR 2022: Proceedings of the 5th Annual on Lifelog Search Challenge, Newark, NJ, USA, June 27 - 30, 2022*, pages 2–7. ACM.

Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*.

Vannevar Bush. 1945. As we may think. *Atl. Mon.*, 176(1):101–108.

Jim Gemmell, Gordon Bell, and Roger Lueder. 2006. Mylifebits: a personal database for everything. *Commun. ACM*, 49(1):88–95.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, pages 1988–1997. IEEE Computer Society.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy,

Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880. Association for Computational Linguistics.

Patrick 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. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *ICLR*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. A demonstration of the solid platform for social web applications. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 223–226. ACM.

Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2022. A survey on multi-hop question answering and generation.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP (1)*, pages 3980–3990. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Halevy. 2021. Database reasoning over text. In *ACL/IJCNLP (1)*, pages 3091–3104. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Benchmark Statistics

A.1 Categories of questions

The crowdsourced questions from 7 people led to the categories of questions shown in Table 8. We gave 7 people the task of writing down questions over their potential lifelogs, and also categories of their questions. We than merge the categories which resulted in the categories shown in Table 8 below.

A.2 Events

Table 9 summarizes the 25 main lifelog events in TimelineQA. Chat is the most frequent events with 40M occurrences in all the 3k lifelogs. The grocery event tends to be longest event type since each entry not only describes the items purchases but also people met at shopping. There are also rare events such as college / grad school moves and graduations occurring with low probabilities.

B Fine-Tuning Setup

B.1 Atomic QA

For fine-tuning the QA systems on the timeline episodes, we use haystack[5] implementation for RAG and Extractive QA. For the retriever, we use ground truth training episodes in the training split, and then fine-tune[6]using in-batch examples as hard negatives, with a batch size of 64, learning rate of $1.5e{-}5$, weight decay 0.75, and number of warmup steps 200, for 1 epoch. For the reader, we start with a fine-tuned ROBERTA model[7], with a batch size of 128, warmup proportion of 0.2, learning rate of $1e-5$, for 2 epochs.

B.2 Multi-hop QA

Our implementation of multi-hop QA is based on the Tapex implementation in HuggingFace's Transformers library.[8] We experimented with both the BART-base and Bart-large architecture with or without table pre-training. For fine-tuning, we use a learning rate of 3e-5 with weight decay 1e-2, a batch size of 8, and a beam size of 5 for beam search decoding. We set the max length of the input sequence (the serialized table) to 1,024 sub-word tokens and the max length of the decoded response to 128 sub-word tokens.

Our multi-hop QA dense retriever implementation is based on the SentenceTransformers library (https://www.sbert.net/). We used the all-MiniLM-L6-v2 model checkpoint for the zero-shot setting. For fine-tuning, we randomly sample 20 true positive examples from the grounth truth evidence list for every question in the training set as the positive question-evidence pairs. We create the set of negative pairs by randomly sampling question-evidence pairs where the question and evidence are from different episode category (e.g., chat vs. dining), so that they are guaranteed hard negatives. We fine-tune the model with a batch size of 16 and a learning rate of 3e-5.

We ran all experiments on an AWS p4d server with A100 GPU's (1 GPU is used for each run). The experiments took a total of 25.4 GPU hours.

C Multi-hop QA with InstructGPT

Since large pre-trained LMs (LLMs) have shown promising zero-shot performance across QA tasks, we also test the 175B-parameter InstructGPT (Ouyang et al., 2022) on 100 sampled multi-hop TimelineQA questions. Similar to to the experiments for TableQA, we leverage 3 settings of the retrievers: oracle, fine-tuned (FT), or zero-shot (ZS). Because the model may generate free-form answers, we compute the accuracy by manually checking whether the answers are compatible with the ground truth. As such, the numbers are not directly comparable to those for TableQA.

As shown in Table 10, InstructGPT signicantly outperforms TableQA readers in the zeroshot settings (e.g., 33% vs. 6.5% accuracy). However, the performance still does not outperform that of fine-tuned TableQA models (59% accuracy). The result suggests a potential direction of leveraging fine-tuned LLMs for the TimelineQA tasks.

[5] https://github.com/deepset-ai/haystack

[6] For detailed steps, follow the tutorial at https://haystack.deepset.ai/tutorials/09_dpr_training

[7] https://huggingface.co/deepset/roberta-base-squad2

[8] See https://github.com/huggingface/transformers/tree/main/examples/research_projects/tapex.

Table 8: Categories of questions and some examples

Episode Category	Explanation	Example queries
Care for oneself	Preventive medical appointments, self-care (e.g., massages, pedicures), medications, health metrics (e.g., heart rate, blood pressure)	When was the last time I visited my dentist? What was my average heart rate last week?
Taking care of parents	Visiting parents or family gatherings, taking them for health checkups and self-care, administering medications	When was the last time I took my dad for his annual checkup?When was the last time I had dinner with my parents?
Raising children	Celebrating milestones, taking them for checkups/vaccinations, special moments	When was the last time my child had her yearly checkup? What type of cake did we buy for her last birthday?
Pets	First time pet arrived, pet's birthday, pet care/grooming, loss of pet	When was the last time my pet was groomed? How much did I spend on pet care last year? When did my pet pass away?
Accidents and recovery	Details of accidents, experiences, and recovery	How old was I when I fell from my bike? How many stitches did I receive from my bike accident?
Socializing	Spending time with friend, party, memorable conversations, dating, celebrations of events/holidays	How often did I chat with Avery last year? When was the first time I met Avery?
Daily life	Eating, cooking, drinking, shopping, religious practice, exercising, walking, meditating	When was the last time I visited restaurant X? How often did I cook pasta last month? How long did I meditate last week?
Entertainment	Hobbies, watching sports, participating in sports, watching media, reading media	How long did I exercise last week? when did I first learn to play the piano? where is the meditation group to meet this week? who went to watch the fashion show with me last Friday?
Life Milestones	Starting and graduating from schools, interviewing for jobs, starting and quitting jobs, promotions, engagement, marriage and divorce, anniversaries, work milestones, enrichment activities	When was my first job interview? Where did we go for the anniversary last year?
Managing Finances	Investment decisions, credit score tracking	How much did my daughter obtain from the trust last year? how much did I pay for my first investment property?
Travel	Travel preparation, getting there (by air, water, car), events during travel	Did I take any photo in front of Big Ben? Are we going to London from the hotel by car? How much did the airbnb total for our last London trip?
Housing	Finding a place to live, housework, house maintenance	When did I move the last time? did I make an appointment to clean the drains? when did I last purchase the laundry pods?
Diary Entries / Journaling	Anything I may want to remember about my day, the conversations I had or other experiences I've gone through	I went to a friend's graduation ceremony. Interesting conversation with a stranger at a grocery store.

Table 9: Breakdown of TimelineQA by events.

Event	#entries (M)	#tokens	Category	#entries (M)	#tokens	Category	#entries	#tokens
chat	40.76	11.19	hobbies	2.39	6.05	birth_info	3,000	8.23
watch tv	17.77	7.25	dining	1.25	15.69	college move	726	10.62
read	11.87	5.00	pet care	0.72	6.00	college graduation	726	11.60
breakfast	9.56	6.89	places visited	0.70	13.64	grad school move	3	11.00
dinner	9.56	6.17	bake	0.41	16.94	grad school graduation	3	8.00
lunch	9.55	6.17	cook	0.41	15.72	**Summary**		
exercise	8.99	3.17	child med. care	0.22	15.91	sparse	14,941,703	8.51
social media	5.93	6.00	travel	0.17	10.79	medium	34,522,030	8.12
grocery	4.78	18.94	personal med. care	0.16	11.40	dense	78,559,743	8.50
dating	2.64	8.00	parent med. care	0.16	15.90	all	128,023,476	8.40

Table 10: InstructGPT performance Results on multi-hop QA
We report the results on a sample of 100 questions.

Retriever	Oracle	FT-retriever	ZS-retriever
InstructGPT	33.0	25.0	18.0

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☑ A2. Did you discuss any potential risks of your work?
Section 7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4 and Appendix A

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
This will be put in the open-sourced repository.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. The data is synthetic.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5 and Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5 and Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.