QAMPARI: An Open-domain Question Answering Benchmark for Questions with Many Answers from Multiple Paragraphs

Samuel Joseph Amouyal **Ohad Rubin** **Ori Yoran** **Tomer Wolfson**
Jonathan Herzig **Jonathan Berant**
Blavatnik School of Computer Science, Tel Aviv University, Israel
samueljoseph@mail.tau.ac.il {ohad.rubin, joberant}@cs.tau.ac.il

Abstract

Existing benchmarks for open-domain question answering (ODQA) typically focus on questions whose answers can be extracted from a single paragraph. By contrast, many natural questions, such as *"What players were drafted by the Brooklyn Nets?"* have a *list of answers*. Answering such questions requires retrieving and reading from many passages, in a large corpus. We introduce QAMPARI, an ODQA benchmark, where question answers are lists of entities, spread across many paragraphs. We created QAMPARI by (a) generating questions with multiple answers from Wikipedia's knowledge graph and tables, (b) automatically pairing answers with supporting evidence in Wikipedia paragraphs, and (c) manually paraphrasing questions and validating each answer. We train ODQA models from the retrieve-and-read family and find that QAMPARI is challenging in terms of both passage retrieval and answer generation, reaching an F_1 score of 26.6 at best. Our results highlight the need for developing ODQA models that handle a broad range of question types, including single and multi-answer questions.

1 Introduction

Open-domain question answering (ODQA) is a core language understanding task concerned with answering factoid questions over large document collections (Voorhees and Tice, 2000; Brill et al., 2002). Due to its wide applicability, ODQA has received substantial attention in recent years (Chen et al., 2017; Lee et al., 2019; Karpukhin et al., 2020). Typically, systems solving ODQA tasks follow the "retrieve-and-read" paradigm, where *a retriever* first retrieves a set of candidate passages, followed by *a reader* which receives the retrieved passages and produces the final answer.

The retrieve-and-read paradigm has been shown to be effective for benchmarks such as Natural Questions (NQ) (Kwiatkowski et al., 2019) and

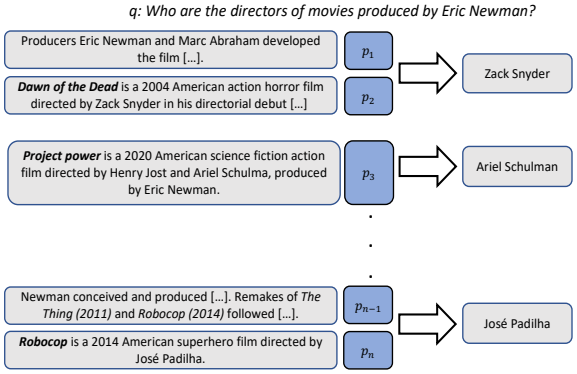


Figure 1: An example from QAMPARI with a generated question q, a subset of its evidence Wikipedia passages (left, p_i) and the answers they lead to.

TriviaQA (Joshi et al., 2017), where the answer is typically a single phrase from a single passage. However, in many cases, a question might have *many* answers that are spread across multiple passages. Consider the example in Fig. 1. Eric Newman produced multiple movies, so finding them along with their directors requires incorporating information from many passages. Such questions pose two main challenges to retrieve-and-read systems. First, as there are multiple answers, that can be far apart, the reader model must reason over a long text sequence to generate all of the correct answers. Second, since the reader is computationally constrained to process at most K passages, the retriever must score all necessary passages at its top-K results, which is challenging and even impossible when the number of such passages is $\geq$K.

While recent works explored questions that involve reading multiple passages, their overall number of passages was quite small. AMBIGQA (Min et al., 2020) studied ambiguous questions from NQ with several plausible answers. However, as 70% of its questions have at most 2 answers, retrieve-and-read models can be easily adapted to the AMBIGQA task. The HOTPOTQA (Yang et al., 2018) dataset focused on multi-hop reasoning, but its

questions require no more than 2 passages to answer. Last, WIKINLDB (Thorne et al., 2021) was proposed as a benchmark for testing reasoning over multiple facts. However, WIKINLDB restricted its text corpus to databases of 1,000 facts at most, making it significantly smaller than standard ODQA corpora. Moreover, these facts are model-generated utterances rather than natural language passages.

In this work, we present QAMPARI, a benchmark for **Q**uestions with many **A**nswers over **M**ultiple **Par**agraphs, **I**ndeed. All questions in QAMPARI have at least 5 answers, with an average of 13 answers per question. Examples are semi-automatically generated using two data sources, Wikidata (Vrandečić and Krötzsch, 2014) and Wikipedia tables. We automatically generate multi-answer questions of the form *"What/Who has [relation] with [entity]?"* and convert these into pseudo-language using manually defined templates. We then verify our questions are answerable, given Wikipedia sentences, by automatically extracting evidence passages for all answers. Finally, we use crowdsourcing to validate example correctness, and to paraphrase questions from pseudo-language into natural language (Wang et al., 2015). To further increase the richness of questions, we also generate *composition* questions, that compose two relations (as in Fig. 1), and *intersection* questions, such as *"What movies were produced and directed by Clint Eastwood?"*. Overall, QAMPARI contains 2K test questions and more than 60K training examples – see Tab. 1 for some examples.

We evaluate models from the retrieve-and-read family and find that they struggle on QAMPARI. Specifically, we use a BM25 (Robertson and Zaragoza, 2009) retriever followed by one of two readers: (1) a RAG-style reader (Lewis et al., 2020) that decodes an answer or abstains given each passage independently, and (2) an FiD reader (Izacard and Grave, 2021), that directly decodes the answer list given encoded representations of many passages. To evaluate, we compare the set of answers predicted by a model to the gold set of answers.

When training models in a multi-task setting of NQ and QAMPARI we observe that QAMPARI is challenging in terms of both passage retrieval and answer generation. Models reach an F_1 score of 26.6 at most. In addition, models are able to return over 80% of the correct answers only for 30% of our examples, well below typical performance on single-answer datasets such as NQ.

To summarize, we present QAMPARI, a challenging benchmark for evaluating the ability of ODQA models to handle questions with many answers over multiple passages from Wikipedia. We advocate to evaluate ODQA models not on QAMPARI alone, but alongside benchmarks such as NQ and TriviaQA. This joint evaluation would better test for ODQA models ability to handle both single- and multi-answer questions, tests which are conspicuously absent from current benchmarks.

The QAMPARI benchmark, models and relevant codebase are available at: `https://samsam3232.github.io/qampari/`.

2 Dataset Construction

We present our process of generating examples for QAMPARI. Each example in QAMPARI is a triple $(q, \mathcal{A}, \mathcal{P})$, where q is a question, $\mathcal{A}$ is a set of answers and $\mathcal{P}$ is a set of passages from our target corpus. Each answer $a \in \mathcal{A}$ has 1-2 evidence passages from $\mathcal{P}$ (see Fig. 1). We define passages to be consecutive sentences from our corpus (Wikipedia), that span at most 100 words. As our focus is on questions with many answers, all examples in QAMPARI have $|\mathcal{A}| \geq 5$.

Overview We generate examples using two steps. First, we generate *simple questions* that involve a single entity and a single relation, e.g. *"Who was drafted by the Brooklyn Nets?"* (§2.1). Then, we expand such questions in order to generate *complex questions* that involve *intersection* and *composition* operations (§2.2).

To increase diversity, our questions are generated using two data sources, Wikidata and Wikipedia tables. We first describe the example generation over Wikidata, then briefly present the generation process from Wikipedia tables in §2.3. In both cases, we ensure all answers can be derived from evidence passages in Wikipedia.[1] Tab. 1 presents examples from each data source and question type.

Notation We introduce notation for formal queries over Wikidata, used for explaining our example generation process. Wikidata is a knowledge graph, $\mathcal{K}$, that can be viewed as a set of labeled edges (e_1, r, e_2). Graph nodes $e_1, e_2 \in \mathcal{E}$ are entities which are connected by an edge labeled with the relation $r \in \mathcal{R}$. For example, one possible labeled edge is (`BarackObama`, `ReceivedAward`, `NobelPeacePrize`).

[1] Wikipedia dump: 2021-08-01

Data source	Question type	Question	Answer example
Wikidata	Simple	Who is or was a member of the Australian Army?	George Macarthur-Onslow
	Intersection	What movie produced by Jerry Ward was also directed by Vincent Sherman?	Hard Way
	Composition	From which country did Seattle Storm make draft selections?	Australia
Wikipedia tables	Simple	What magazine is a satirical magazine?	The Clinic
	Composition	What are the museums found in Concord, Massachusetts?	The Wayside

Table 1: Example questions and one representative answer for all data sources and question types.

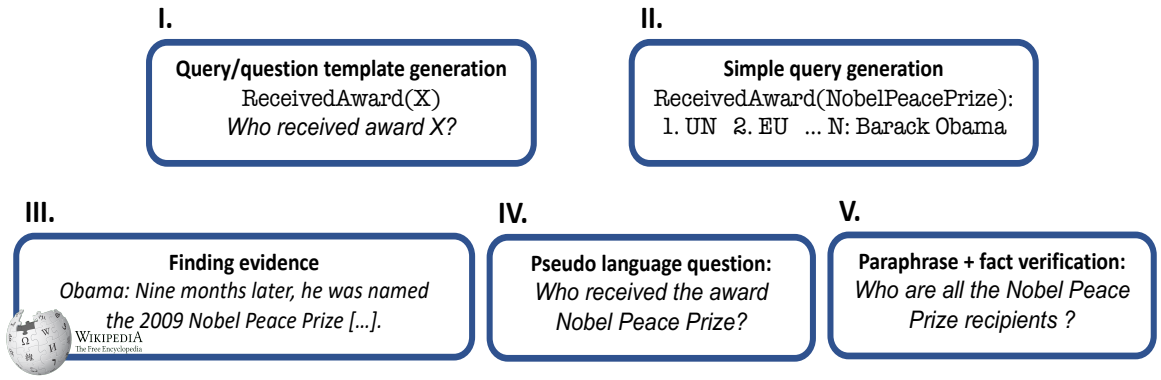


Figure 2: An overview of example generation for simple questions.

One can query $\mathcal{K}$ by applying a relation r over an entity e, resulting in a *simple query* $r(e)$ whose *denotation* (answer set) is $[\![r(e)]\!] = \{e_i \mid (e_i, r, e) \in \mathcal{K}\}$. *Composition queries* can be formed by applying a relation over the result of a simple query. We denote a composition query by $r_2(r_1(e))$, and its denotation is $[\![r_2(r_1(e))]\!] = \{e_i \mid \exists e_j \; s.t \; (e_i, r_2, e_j) \in \mathcal{K} \wedge (e_j, r_1, e) \in \mathcal{K}\}$. Last, an *intersection query* $r_1(e_1) \sqcap r_2(e_2)$ corresponds to the intersection of two simple queries, that is, $[\![r_1(e_1) \sqcap r_2(e_2)]\!] = \{e_i \mid (e_i, r_1, e_1) \in \mathcal{K} \wedge (e_i, r_2, e_2) \in \mathcal{K}\}$.

2.1 Simple Questions

Fig. 2 provides an overview of our semi-automatic procedure for creating *simple question* examples: (i) We manually define query templates, (ii) automatically populate query templates using $\mathcal{K}$ to create queries with a sufficiently large number of answers in $\mathcal{K}$, (iii) automatically identify evidence passages for the answers and filter out noisy examples, (iv) map query templates to question templates to obtain pseudo-language questions, and (v) validate answers and paraphrase pseudo-language questions through crowdsourcing. Next, we describe each of these steps in detail.

Generating query templates We manually select a set of 135 relations $\bar{\mathcal{R}} \subset \mathcal{R}$, which will be used in our query templates. We select relations going through the most frequent relations in Wikidata and choosing ones for which denotations often contain many entities (e.g., `ReceivedAward`). The full list of relations is provided in App. A. For each relation, we manually write a template that will be used to map queries to pseudo-language questions. For example, the template for `ReceivedAward` is *"Who received the award X?"*

Some relations are underspecified – for example, `LocatedIn` can describe the location of buildings, geographical features, and cities. When generating synthetic questions, this leads to vague questions such as *"What is located in Paris?"*. To address this issue we manually split these to *typed relations* that specify the semantic type of their answers/denotations. This split is done using the type hierarchy given in Wikidata and given the type t of answer entities. We denote typed relations by r_t, and the denotation of $r_t(e)$ comprises all entities of type t returned by $r(e)$.[2] For example, the entity `The Louvre` has type `cultural`

[2]This can be written as $r(e) \sqcap \texttt{Type}(t)$, as `Type` is a Wikidata relation.

organization, and we can map the relevant query template to the pseudo-language question *"Which cultural organization is located in Paris?"*.

Simple query generation We instantiate all possible simple queries using all $r \in \bar{\mathcal{R}}$ and entities e in Wikidata. For a relation r (or r_t), we keep the query $r(e)$ iff $|r(e)| \geq 5$. We denote this set of instantiated simple queries by $\mathcal{S}$, which contains 1,431,268 simple queries.

Finding evidence sentences As our goal is to create an ODQA benchmark, we must verify that every answer is indeed found in our target text corpus. We do this by (a) identifying candidate evidence sentences from Wikipedia, and (b) verifying that they entail the answer, using a Natural Language Inference (NLI) model.

Specifically, every simple query-answer pair can be viewed as a triple (e_1, r, e_2). We use a "distant supervision" approach (Mintz et al., 2009), similar to KELM (Agarwal et al., 2021), and define any sentence in the Wikipedia page of entity e_1 that contains the entity e_2, or one of its Wikidata aliases, as a candidate evidence sentence (and vice versa in the Wikipedia page of e_2). For example, in Fig. 2, the evidence for the triple (BarackObama, ReceivedAward, NobelPeacePrize) appears on the Wikipedia page of Barack Obama, where the phrase *Nobel Peace Prize* appears.

Aligning Wikipedia sentences to Wikidata can lead to false positives. For example, for the triple (TheGoonies, HasScreenwriter, StevenSpielberg), most mentions of Spielberg in the page TheGoonies are not as a screenwriter. To account for this, we use an off-the-shelf NLI model.[3] For every answer, we consider each candidate evidence sentence along with its two preceding sentences, and check whether they entail the hypothesis phrase describing triple (e_1, r, e_2). We use templates to phrase triples as short declarative sentences (*"The Goonies has Steven Spielberg as screenwriter"*). An answer is *validated* if there is an evidence sentence that entails the triple. Manual analysis shows this process eliminates 70% of false positives (sentences not entailing the triple), while removing only 7.5% of the correct alignments.

[3] https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli

Query filtering After finding evidence sentences, we only keep queries that at least 80% of their answers were validated and their number of validated answers lies between 5 and 200. The resulting set contains 60,792 simple queries, where each query has a set of validated answers, $\mathcal{A}$, and of passages $\mathcal{P}$ that contain the identified evidence sentences.[4]

We now describe how simple queries are expanded to complex queries.

2.2 Complex Questions

To increase the diversity of QAMPARI, we automatically expand simple queries to composition and intersection queries, for which answers require reading two passages.

Intersection Intersection queries are generated by finding two simple queries such that the size of the intersection of their denotations is at least 5. To avoid improbable questions such as *"Which competition was won by Manchester City and had Manchester City as a participant?"*, we add a constraint that the denotation of one of the simple queries cannot be a subset of the other. Formally, the set of intersection queries are all queries $r_1(e_1) \sqcap r_2(e_2)$ such that $|[\![r_2(e_2) \sqcap r_1(e_1)]\!]| \geq 5$, $[\![r_1(e_1)]\!] \not\subseteq [\![r_2(e_2)]\!]$ and $[\![r_2(e_2)]\!] \not\subseteq [\![r_1(e_1)]\!]$.

Pseudo-language questions are generated by heuristically combining the two simple questions, for example *"Which television program had Chris Carter as screenwriter and had Frank Spotnitz as screenwriter?"*. There is no need to perform answer validation since all of the underlying intersecting answers were already validated.

Composition To create composition queries, we manually handpick a set of 423 relations $\mathcal{R}_{\text{comp}} \subset \mathcal{R}$ (list in our codebase), in a process similar to simple queries. Then, we generate all the possible composition queries $r_2(r_1(e))$ such that $r_1(e) \in \mathcal{S}$, $r_2 \in \mathcal{R}_{\text{comp}}$, and $|[\![r_2(r_1(e))]\!]| \geq 5$. An example composition query is *"What is the height of buildings located in Dubai?"*.

Unlike intersection queries, in composition queries we need to validate that our new triples (e_i, r_2, e_j), where $e_j \in [\![r_1(e)]\!]$, are indeed supported by Wikipedia sentences. We use the same procedure to find evidence sentences for triples (e_i, r_2, e_j), and consider an answer e_i as *validated* if both (e_i, r_2, e_j) and (e_j, r_1, e) can be aligned to Wikipedia. We keep all complex queries where

[4] We keep a single evidence passage for every triple.

80% of the answers are validated. Finally, we manually define templates for relations in $\mathcal{R}_{\text{comp}}$ to generate pseudo-language questions.

2.3 Questions from Wikipedia Tables

To further diversify QAMPARI, we create an analogous pipeline for generating simple and composition questions from Wikipedia tables, with more open-ended relations compared to Wikidata. We briefly describe this pipeline.

We look at all Wikipedia tables with title *"List of X"* that have at least 5 rows, in total, 1,897 tables. We find the "key" column, c_{key} in each table using the table classifier from Talmor et al. (2021), which outputs the column of entities that the table describes. For example, in the table *List of nuclear whistle blowers*, c_{key} is *'name'* and specifies the whistle-blower names. This naturally creates simple questions of the form *"Who or what is X?"*.

Simple questions are expanded to composition questions by looking at non-key columns, $c_{\text{non-key}}$ and asking what rows in the table have the value v in column $c_{\text{non-key}}$. For example, what is the value in the column *'Year'* for nuclear whistle-blowers.

Questions from Wikipedia are validated using a procedure similar to Wikidata. For each answer entity e, we validate that the Wikipedia page for e contains the relevant words that are part of the name of the table as well as the value (for composition questions), and only keep questions where 80% of the table rows are validated and the number of validated answers is at least 5. Overall, we generate 170 simple questions and 6,036 composition questions using this process.

2.4 Data Split

We provide a training set with QAMPARI, whose goal is to teach the model to handle multi-answer questions. However, we do not want the model to use the training set to memorize how particular Wikidata relations map to text patterns, as our goal is to test language understanding regardless of Wikidata relations.

Consequently, we perform a *relation split*, randomly splitting the set $\bar{\mathcal{R}}$ into two equally-sized sets $\bar{\mathcal{R}}_{\text{train}}$ and $\bar{\mathcal{R}}_{\text{test}}$. Simple queries are assigned to the train/test set based on their relation, composition queries $r_2(r_1(e))$ are assigned to the test set iff either r_1 or r_2 are in $\bar{\mathcal{R}}_{\text{test}}$, and intersection queries $r_1(e_1) \sqcap r_2(e_2)$ are placed in the test set iff both r_1 and r_2 are in $\bar{\mathcal{R}}_{\text{test}}$.

At this point, we can create the final train/development/test split (see also Tab. 2). The main bottleneck in our example generation pipeline is validation of the test set through crowdsourcing (§2.5), since each question requires validating all of the answers in the list. Thus, we pre-determine the test set to contain 1,000 simple questions (830 from Wikidata and 170 from Wikipedia tables) and 1,000 complex questions (400 Wikidata composition questions, 400 Wikidata intersection questions, and 200 Wikipedia tables composition questions). For simple Wikidata questions, we sample 830 questions such that the distribution over relations from $\bar{\mathcal{R}}_{\text{test}}$ is roughly uniform. All Wikipedia tables simple questions are placed in the test set, and for complex questions we randomly sample the pre-determined number from the set of generated questions. Last, the test set is randomly split in half to a development set and test set. We also sub-sample training set examples, such that each relation appears in at most 1,000 examples.

2.5 Crowdsourcing

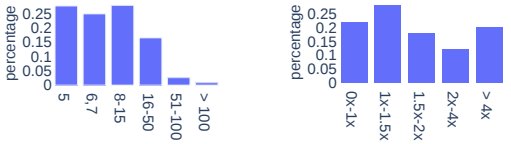
We validate the correctness of development and test examples and paraphrase them into natural language through crowdsourcing.

Correctness validation For every question and answer, we present a crowdsourcing worker with the question, the answer, and links to the Wikipedia page (or pages for complex questions) with the evidence passage. We then ask the worker to check if the question can be answered from the given pages, using the text only (no infoboxes or tables).

As the grand majority of examples are correct, we control for quality by injecting wrong answers in 10% of the cases and reject workers that fail to identify those wrong answers. Moreover, we manually verify 5% of examples marked as *correct* and all examples marked as *incorrect*, and again reject low-performing workers.

Overall, 24 annotators validated 30,259 answers for an average pay of 12.5$ per hour. We find that our process for generating examples is accurate, with 96.6% of the answers validated. Non-validated questions were replaced until 2,000 questions were validated. A question is defined non-validated if its number of distinct answers goes bellow 5. Snapshots from the presented tasks are in App. B.

Paraphrasing Since our questions are in pseudo-language, we follow past work (Wang et al., 2015)



(a) # answers (b) Factor

Figure 3: Left: binned distribution of the number of answers per example. Right: Results of manual analysis over 50 examples for whether Wikipedia contains more correct answers that are absent from Wikidata. We show in each bin the growth factor in the number of correct answers that were found w.r.t the size of the gold set.

and ask workers to re-phrase the questions in the development/test sets. We restrict this task to workers from the US or the UK who pass a qualification test. We randomly verified half of the paraphrases for each worker for quality assurance.

3 Dataset Analysis

Statistics QAMPARI contains 63,911 examples with 1,000 examples in the development set, 1,000 in the test set, and the rest in the training set. Tab. 1 shows examples for all sources and questions types.

Tab. 2 provides key statistics on QAMPARI (development and test statistics are aggregated). Test examples in QAMPARI have 13.23 answers on average and a median of 7 answers. This is substantially higher than, e.g., AmbigQA, where the median is 2. Simple questions tend to have more answers than complex questions and are typically shorter than complex questions. Test examples are shorter than training examples since they were rephrased by annotators who used more concise and natural phrasings.

Figure 3a shows a binned distribution over the number of answers on the development and test sets. We can see that roughly half of the questions have 8 answers or more, a non-negligible fraction (20%) have more than 15 answers, and 3.5% have more than 50.

Tab. 3 shows the frequency of the top-10 relations in the training and test sets of QAMPARI, as well as the top-10 semantic types of answers, which can be inferred from relation. Mirroring Wikipedia, we observe that the relations and types are focused on people, locations, and the entertainment world.

Manual analysis As mentioned, we use crowdsourcing to validate that gold answers have evidence in Wikipedia. However, Wikipedia can contain additional correct answers that are not in Wikidata/Wikipedia tables. Since manually annotating all correct answers on Wikipedia is virtually impossible, we estimate the frequency of this phenomenon. We sample 50 examples from QAMPARI, and ask an expert annotator to find more correct answers on Wikipedia within 5-10 min.

Fig. 3b shows the results of the manual analysis. In 20% of the questions, no additional answers were found, and for more than 60% of the questions, the gold set of answers contains at least half of the answers found by the annotator. Overall, precision estimates of models against the gold set of answers should be taken with a grain of salt.

4 Experimental Evaluation

We now turn to our experimental evaluation.

4.1 Models

ODQA models typically fall into either the retrieve-and-read framework or the 'closed-book' framework (Roberts et al., 2020), where the model provides answers using knowledge encoded in its parameters. Here, we use baseline models from the retrieve-and-read family as they reach state-of-the-art performance and are more parameter-efficient.

Retriever We use BM25 (Robertson and Zaragoza, 2009) to index Wikipedia passages and query the index. As mentioned (§2), we chunk Wikipedia into passages, each containing consecutive sentences with at most 100 tokens, similar to DPR (Karpukhin et al., 2020).

BM25 in a strong sparse retrieval baseline that scores question-passage pairs based on lexical similarity. BM25 is notoriously hard to beat using unsupervised methods (Izacard et al., 2021; Ram et al., 2022), and obtains respectable performance even compared to supervised methods. We leave training a dense passage retriever (Karpukhin et al., 2020) for the near future. Specifically, we return from BM25 the top-200 passages for each question.

Reader We evaluate two encoder-decoder readers, both initialized from T5 (Raffel et al., 2019). a "RAG-like" model (Lewis et al., 2020) that generates answers from each passage independently and Fusion-in-Decoder (FiD) (Izacard and Grave, 2021), which encodes multiple passages and decodes a list of answers.

For RAG, the model encodes each of the 200 retrieved passages independently and outputs either

		Total	Simple (WD)	Simple (WP)	Intersection (WD)	Comp (WD)	Comp (WP)
# Questions	train	63,911	28,574	-	2,301	25,200	5,836
	test	2,000	830	170	400	400	200
Mean # Answers	train	13.25	16.65	-	9.19	9.74	13.35
	test	13.23	15.69	23.84	8.94	8.77	11.51
Median # Answers	train	8.0	9.0	-	7.0	7.0	8.0
	test	7.0	7.5	17.0	7.0	6.0	7.0
Mean Question len.	train	12.69	8.78	-	16.69	15.18	19.47
	test	9.51	7.91	8.61	11.65	10.35	10.99

Table 2: Key statistics of QAMPARI by question type and data source (**WD** for Wikidata, **WP** for Wikipedia tables, **Comp** for composition). Test statistics are an aggregation over development and test sets.

Train relations		Test relations		Semantic type	
Name	**%**	**Name**	**%**	**Name**	**%**
Cast member	11.1	Directors	13.1	Human	27.6
Performers	9.8	Screenwriters	10.7	Creative work	24.2
Location	9.4	Producers	6.9	Film	11.8
Part of	8.3	Education place	6.4	Spatial entity	7.1
Publication date	5.3	Winners	5.2	Competition	5.8
Place of birth	4.6	Owners	4.7	T.V series	2.9
Dates of birth	4.6	Is a	4.6	Album	2.9
Teams	3.7	Composers	4.5	Person	2.6
Directors	3.71	Country of origin	4.1	Building	1.4
Sport played	2.8	Main group	3.6	Human settlement	1.4

Table 3: The 10 most frequent relations and semantic types in QAMPARI.

"Not Relevant" for no answer, or *"Answer: X"* for some *X*. The predicted list of answers is the union of all answers predicted across passages.

We train RAG in the following manner. For simple questions with answers $\mathcal{A} = \{a_i\}_{i=1}^{|\mathcal{A}|}$, we take the evidence passage p_i for each a_i and train the model to decode a_i given the passage p_i and the question q. For complex questions, where each answer a_i requires 1-2 passages of evidence, we create a positive examples from each of the two evidence passages. Last, to train RAG to emit *"Not Relevant"*, we sample for each positive passage a negative passage p_{neg} by selecting the top-scoring BM25 passage that is not an evidence passage and does not contain the answer. We then train the model to emit *"Not Relevant"* given q and p_{neg}.

FiD (Izacard et al., 2021) uses an encoder to encode each of the retrieved passages (along with the input question) and the decoder attends to the encoded representations to decode a list of answers. Since each example is now an input-output pair, we train with standard maximum likelihood.

FiD is computationally expensive as the decoder attends to a large number of encoded tokens and the generated output is long. Thus, we can only fit the top-50 passages from BM25 using T5-large on a single A100 GPU. We apply teacher forcing (Williams and Zipser, 1989) at training time, i.e., the model is given the gold passages and then the top scoring passages according to BM25. If $|\mathcal{P}| > 50$, the model sees a random sample of size 50 from the set of gold passages.

4.2 Experimental Setup

As explained in §1, we create QAMPARI as a benchmark to be evaluated alongside other ODQA benchmarks such as NQ. Since QAMPARI is semi-automatically generated, one can develop models tailored for QAMPARI, but our goal should be to have a single model that can perform well across a wide variety of question types. Thus, we train and test models on QAMPARI only, but also in a multi-task setup, where models are trained on both NQ and QAMPARI.

Our main metrics are recall, precision, and F_1. Specifically, for a test example $(q, \mathcal{P}, \mathcal{A})$, and a predicted set of answers $\mathcal{A}_{\text{pred}}$, recall, precision, and F_1 are computed in the typical manner, comparing $\mathcal{A}$ and $\mathcal{A}_{\text{pred}}$, while allowing for aliases, that is, a gold answer is considered covered if it or one of its aliases appears as a predicted answer. A model score is given by averaging recall, precision, and F_1 across all examples. To get a sense of the average accuracy across examples we measure the fraction of examples where F_1 is at least 0.5 (%$F_1 \geq$0.5) and the fraction where recall is at least 0.8 (%Recall$\geq$0.8). We focus on recall and F_1 since, as shown in §3, precision is a only approximate due to additional answers not covered by Wikidata. For NQ, we use the standard exact match (EM) metric.

We evaluate the retriever with RECALL@K, that is, the fraction of answers that appear in the top-K retrieved passages, averaged across examples. This metric comes in two variants: (a) Answer RECALL@K (ARECALL@K): for every gold answer whether it or one of its aliases appear in the

		QAMPARI					NQ
		Recall	**Precision**	**F_1**	**%F_1 ≥0.5**	**%Recall≥0.8**	**EM**
RAG-Base	QO	46.5	18.1	22.1	11.0	28.5	-
	MT	49.7	22.0	24.9	13.0	29.7	-
RAG-Large	QO	50.5	19.8	24.8	12.6	29.9	-
	MT	**52.4**	21.6	**26.6**	15.3	**30.9**	-
FiD-Base	QO	15.5	31.2	19.2	11.2	0.9	-
	MT	14.2	32.6	18.2	11.0	0.4	35.2
FiD-Large	QO	21.6	**35.8**	25.1	**20.1**	3.9	-
	MT	19.0	34.5	22.7	16.9	2.5	41.2

Table 4: Models performance on the QAMPARI test set. (QO): Trained on QAMPARI only. (MT) Multi-task training with NQ. We also provide FiD results on the NQ test set.

		ARECALL@K	ERECALL@K
BM25	K=10	25.9	17.4
	K=25	38.1	28.8
	K=50	46.8	38.9
	K=100	54.7	47.7
	K=200	62.7	55.2

Table 5: BM25 Retriever test results.

top-K retrieved passages. This is a loose metric since an answer can appear even if the evidence does not support the answer; (b) Evidence RECALL@K (ERECALL@K): since we have evidence paragraphs for every answer, we consider for every gold answer the fraction of evidence passages in the top-K retrieved passages. This is a strict metric since an answer can sometimes be answered by passages other than the ones we identified.

4.3 Results

Tab. 5 shows test retrieval results of BM25 on QAMPARI. We observe that even given 200 retrieved passages average recall is only 55.2-62.7. Moreover, for low values of K, average recall is quite low (17.4-25.9 for K=10, 28.8-38.1 for K=25), and there is still substantial improvement from K=100 to K=200. These results illustrate the challenge in retrieving a large number of relevant passages into the "beam" of retrieved results.

Tab. 4 shows test results on QAMPARI. Overall, performance on QAMPARI is low, where %F_1 ≥0.5 is at most 20.1 (FiD-Large, trained on QAMPARI only (QO)) and %Recall≥0.8 is at most 30.9 (RAG-large, multi-task (MT) training). When training on both NQ and QAMPARI (MT), results on QAMPARI are lower than NQ (despite the more permissive evaluation metric), illustrating the challenge in answering multi-answer questions.

RAG is recall-oriented, while FiD is precision-oriented. This can be attributed to the number of negative passages at test time, which is larger than training time for RAG, and also to RAG taking 200 passages as input, while FiD only takes 50. F_1 for both models is around 25-27, and %Recall≥0.8 is extremely low for FiD (3.9% at most).

The MT setup benefits RAG, but not FiD. We hypothesize this is because training on NQ acts as a regularizer that reduces the number of predicted answers.

4.4 Analysis

Question type analysis We break down by question type the test performance of both FiD-Large and RAG-Large trained on QAMPARI only (Tab. 6). The mean number of answers predicted by RAG is dramatically higher than FiD (30.5 vs 5.1), which agrees with our observation that RAG is recall-oriented, while FiD is precision oriented.

Surprisingly, for FiD, performance on simple questions is lower than performance on complex questions, and specifically, intersection questions seem easiest. Possible explanations are: (a) simple questions have on average more answers (see Tab. 2), which makes them harder, and (b) Models can predict the right answer, even with only one of the evidence passages, due to either "shortcuts" (Chen and Durrett, 2019), or knowledge encoded in the parameters (Longpre et al., 2021).

Unlike FiD, the performance of RAG on Wikidata composition questions is lower than simple questions, potentially since it cannot reason over multiple passages and can only emit answers from each passage independently. Specifically, the recall of RAG on composition questions is less than half the recall on intersection questions. An analogous analysis for the MT setup is in Fig. 8, App. C.

		QAMPARI					
		Recall	**Precision**	**F_1**	**%F_1 ≥0.5**	**%Recall≥0.8**	**Mean # answers**
FiD	WD Simple	17.6	29.4	20.0	12.5	2.2	5.55
	WD Intersection	34.9	48.5	39.0	40.0	11.5	5.75
	WD Composition	22.1	40.6	27.2	22.2	5.1	3.83
	WP Simple	5.5	22.7	8.2	2.4	0.0	4.45
	WP Compostion	24.0	38.5	27.6	25.2	6.1	4.96
RAG	WD Simple	48.4	18.4	23.3	12.7	29.4	36.32
	WD Intersection	81.0	25.3	35.4	23.8	73.1	35.32
	WD Composition	35.3	18.2	19.9	7.7	8.3	27.56
	WP Simple	25.8	17.5	18.9	9.4	7.1	37.09
	WP Composition	48.2	19.0	23.6	10.5	27.3	33.36

Table 6: Question type analysis of RAG-Large and FiD-Large, trained in the QO setup. (WD): Wikidata, (WP) Wikipedia tables.

Model precision As mentioned (§3), precision is a lower bound due to potential additional correct answers on Wikipedia. To estimate the effect of this, we randomly sampled 30 questions from the development set and manually computed "true" precision by checking whether answers are correct on Wikipedia. For FiD-Large (QO), estimated precision on this set is 36.0%, while true precision is 67.3%. For RAG-Large (MT) estimated precision is 21.6%, but true precision is 42.1%. This demonstrates that precision should be used to rank models, but not as an absolute measure of true precision.

5 Related work

Datasets Work on ODQA has mostly focused on datasets where an answer is a single phrase from a single passage, such as NaturalQuestions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (Berant et al., 2013), CuratedTREC (Baudiš and Šedivý, 2015), SQuAD (Rajpurkar et al., 2016), and EntityQuestions (Sciavolino et al., 2021)). Multi-hop datasets, such as HotpotQA (Yang et al., 2018), WikiHop (Welbl et al., 2018), and Multi-evidence FEVER (Thorne et al., 2018) output a single phrase but require reasoning over more than one passage. AmbigQA (Min et al., 2020) deals with questions that have multiple answers due to ambiguity. QAMPARI has questions with many answers and thus requires retrieving a much larger number of passages and generating answers from them.

WikiNLDB (Thorne et al., 2021) tests the ability of models to reason over sets of facts. However, retrieval is restricted to at most 1,000 facts, which are model-generated. Here, our corpus is Wikipedia, and we retrieve much longer passages, which makes the setup more realistic, and retrieval and generation more challenging.

Models The retrieve-and-read paradigm is currently the prevailing approach in ODQA, due to its ability to scale to large corpora (Chen et al., 2017; Yang et al., 2019; Lee et al., 2019; Karpukhin et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021; Sachan et al., 2021). However, when the number of evidence passages is large, retrieve-and-read models need to fetch all relevant passages to generate the correct answer. An alternative approach is to use closed-book models, where information is encoded in the model parameters (Roberts et al., 2020; Tay et al., 2022), however this entails using very high-capacity models. Last, a less explored model family that is potentially suitable for large answer sets are virtual knowledge-bases, which encode a corpus into a differentiable knowledge-base that is amenable for retrieval and logical operations (Sun et al., 2021; Dhingra et al., 2020).

6 Conclusion

We presented QAMPARI, an ODQA benchmark which focuses on the ability of models to handle questions that have many answers and thus require reading multiple text passages. QAMPARI is semi-automatically generated, where examples are generated from WikiData and Wikipedia tables, and manual work is done only to prepare pseudo-language templates, to validate examples, and to re-phrase questions. We evaluate strong baselines on QAMPARI and show that the need to retrieve a large number of passages and generate long lists is challenging for state-of-the-art models from the retrieve-and-read family.

We view multi-answer questions as an integral part of the ODQA problem that has thus far been

neglected, and invite the research community to develop models that can simultaneously answer a wide range of question types, include single- and multi-answer questions.

Acknowledgements

We want to thank Omer Bigi Amouyal, Levana Amouyal and Joseph McCrum for their help with the annotation verification process. We also want to thank Ori Ram for his helpful comments. This research was supported in part by The Yandex Initiative for Machine Learning, and The European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800).

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the YodaQA system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction - Volume 9283*, CLEF'15, page 222–228, Berlin, Heidelberg. Springer-Verlag.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 257–264. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4026–4032, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. *CoRR*, abs/2002.10640.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. 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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

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.

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *North American Association for Computational Linguistics (NAACL)*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems*.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haitian Sun, Pat Verga, Bhuwan Dhingra, Ruslan Salakhutdinov, and William W. Cohen. 2021. Reasoning over virtual knowledge bases with open predicate relations. *CoRR*, abs/2102.07043.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodal{qa}: complex question answering over text, tables and images. In *International Conference on Learning Representations*.

Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. 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 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. Database reasoning over text. 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 3091–3104, Online. Association for Computational Linguistics.

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 question answering track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Simple Relations

In Tab. 7. we gathered all the 135 relations we used to create our simple questions. The 423 relations used to create our composition questions can be found in our code base.

B Crowdsourcing Validation

Fig. 4 shows two screenshots of the task crowdsourcing workers performed.

C Analysis of RAG-Large

Table 8 breaks down the performance of RAG-Large and FiD-Large in the multi-task setup.

D Development Set Results

In Tab. 9 we present results analogous to those in Tab. 4 for the development set.

is a	has author	located in	language	occupation
sex or gender	country of citizenship	part of	place of birth	located in
educated at	language spoken, written or signed	has part	played the sport	employer
genre	position held	cast member	country of origin	award received
place of death	made from material	creator	has participant	depicts
maintained by	operator	performer	member of political party	owned by
religion	headquarter location	participant	member of	position played
original language	competition class	publisher	role	record label
work location	director	doctoral advisor	residence	native language
place of publication	medical condition	winner	field of work	form or work
conflict	place of burial	instrument	composer	league
screenwriter	distribution format	producer	sponsor	ethnicity
voice actor	distributed by	participating team	academic degree	manufacturer
architectural style	fabrication method	present in work	production company	cause of death
military branch	manner of death	industry	director of photography	narrative location
original broadcaster	organizer	student of	location of creation	located in or next to body of water
architect	archives at	nominated for	country of registry	allegiance
movement	voice actor	noble title	based on	dedicated to
legislated by	location of formation	developer	contributor to creative work or subject	lyrics written by
located in protected area	tracklist	editor	presenter	religious order
from narrative universe	location of discovery	media franchise	commissioned by	political ideology
commemorates	port of registry	influenced by	indigenous to	operating area
translator	brand	interested in	designed by	illustrator
vessel class	costume designer	drafted by	coach of sports team	convicted of
scenographer	culture	significant place	executive producer	represented by
broadcast by	investor	cover art by	home port	collection creator
armament	inspired by	first appearance	choreographer	animator
source of energy	musical conductor	adapted by	sound designer	has written for
academic major	ratified by	business model	worshipped by	narrator
partnership with	colorist	art director	has work in the collection	military rank

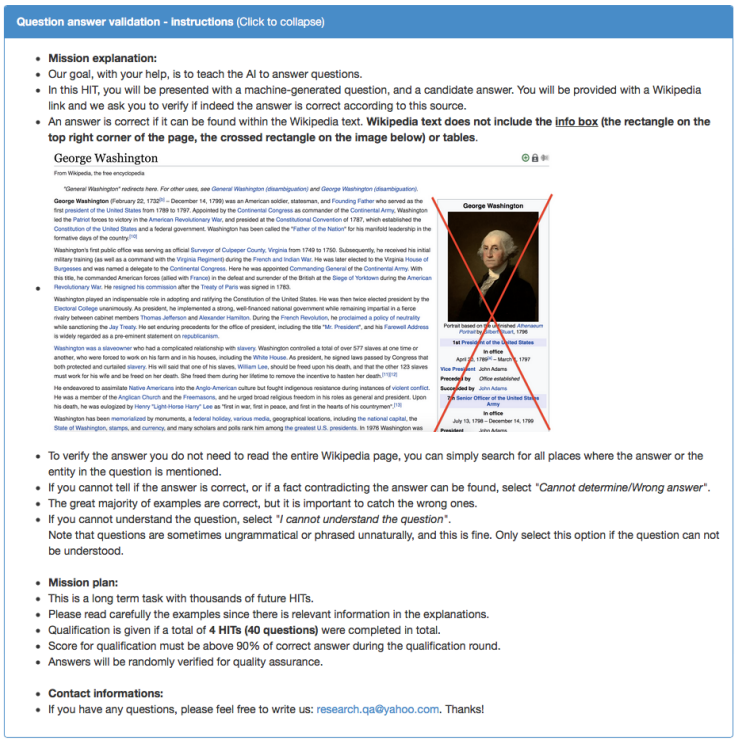
Table 7: Simple relations

		QAMPARI					
		Recall	**Precision**	**F_1**	**%$F_1 \geq$0.5**	**%Recall$\geq$0.8**	**Mean # answers**
FiD	WD - Simple	15.1	28.1	18.0	9.5	1.0	4.86
	WD - Intersection	30.8	46.0	35.3	33.0	7.0	5.57
	WD - Composition	20.6	41.5	25.8	21.2	1.6	3.64
	WP - Simple	5.1	25.1	7.7	2.3	0.0	3.30
	WP - Composition	19.5	31.7	22.8	21.2	4.0	4.57
RAG	WD - Simple	50.0	18.0	23.1	11.7	29.7	37.96
	WD - Intersection	80.7	27.6	37.6	29.5	73.1	31.21
	WD - Composition	39.2	26.5	26.8	18.2	11.6	22.14
	WP - Simple	30.5	14.1	17.0	2.4	8.2	52.96
	WP - Composition	49.4	20.8	25.9	17.9	29.5	30.58

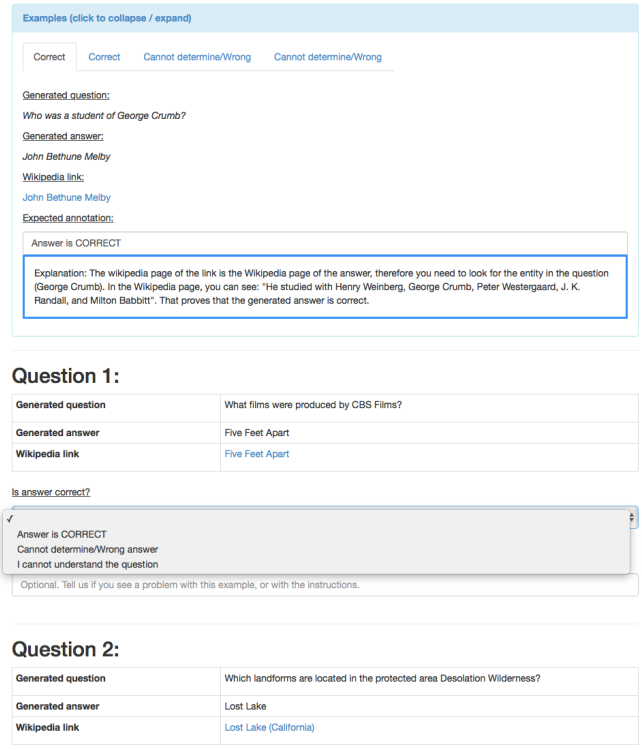
Table 8: Question type analysis of RAG-Large and FiD-Large, trained in the MT setup. (WD): Wikidata, (WP) Wikipedia tables.

		QAMPARI				
		Recall	**Precision**	**F_1**	**%$F_1 \geq$0.5**	**%Recall$\geq$0.8**
RAG-Base	QO	43.6	17.7	21.3	9.6	25.9
	MT	45.0	20.6	23.8	11.5	26
RAG-Large	QO	45.6	19.5	23.8	11.9	26.9
	MT	47.8	20.3	25.0	13.3	27.7
FiD-Base	QO	14.1	29.6	17.6	12	0.9
	MT	12.6	30.7	16.4	10.0	0.9
FiD-Large	QO	19.6	33.0	22.5	17.7	3.1
	MT	18.2	34.0	21.9	16.6	3.1

Table 9: Development results for all models on QAMPARI. (QO): Trained on QAMPARI only. (MT) Multi-task training with NQ.



(a) Instructions



(b) Task

Figure 4: Screenshots from crowdsourcing task.