Hey AI, Can You Solve Complex Tasks by Talking to Agents?

Tushar Khot **Kyle Richardson** **Daniel Khashabi** **Ashish Sabharwal**

Allen Institute for AI, Seattle, WA, U.S.A.
{tushark,kyler,danielk,ashishs}@allenai.org

Abstract

Training giant models from scratch for each complex task is resource- and data-inefficient. To help develop models that can leverage existing systems, we propose a new challenge: Learning to solve complex tasks by communicating with existing agents (or models) in natural language. We design a synthetic benchmark, COMMAQA, with three complex reasoning tasks (explicit, implicit, numeric) designed to be solved by communicating with existing QA agents. For instance, using text and table QA agents to answer questions such as "Who had the longest javelin throw from USA?". We show that black-box models struggle to learn this task from scratch (accuracy under 50%) even with access to each agent's knowledge and gold facts supervision. In contrast, models that learn to communicate with agents outperform black-box models, reaching scores of 100% when given gold decomposition supervision. However, we show that the challenge of learning to solve complex tasks by communicating with existing agents *without relying on any auxiliary supervision or data* still remains highly elusive. We release COMMAQA, along with a compositional generalization test split, to advance research in this direction.[1]

1 Introduction

A common research avenue pursued these days is to train monolithic language models with billions of parameters (Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020) to solve every language understanding and reasoning challenge (Wang et al., 2018, 2019). In contrast, humans often tackle complex tasks by breaking them down into simpler subtasks, and solving these by interacting with other people or automated agents whose skill-sets we are familiar with. This approach allows us to learn to solve new complex tasks quickly and effectively, by building upon what's already known. Can AI systems learn to do the same?

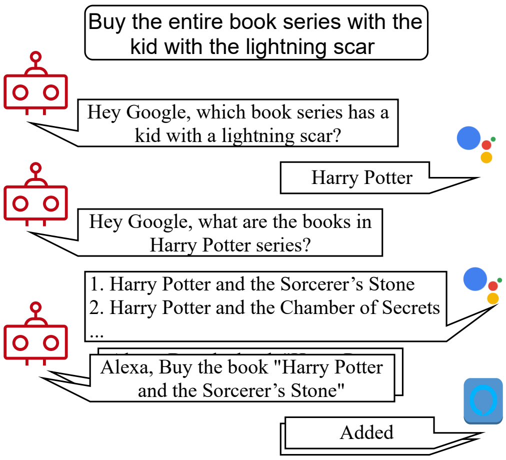

Buy the entire book series with the kid with the lightning scar
Hey Google, which book series has a kid with a lightning scar?
Harry Potter
Hey Google, what are the books in Harry Potter series?
1. Harry Potter and the Sorcerer's Stone
2. Harry Potter and the Chamber of Secrets
...
Alexa, Buy the book "Harry Potter and the Sorcerer's Stone"
Added

Figure 1: Motivating example for a setup where a system is expected to learn to accomplish goals by interacting with agents via a natural language interface.

To facilitate research in this direction, we propose a new reasoning challenge and a benchmark called COMMAQA where, in addition to the usual end-task supervision, one has access to a set of predefined AI agents with examples of their natural language inputs. Importantly, the target end-task is designed to be too difficult for current models to learn based only on end-task supervision. The goal is instead to build models that learn to solve the target task by decomposing it into sub-tasks solvable by these agents, and interacting with these agents in natural language to do so.

As a motivating example, consider the interaction depicted in Figure 1 where a system is asked to buy a book series with a certain property. The system breaks this goal down, using agent-1 (here Google Assistant) to identify the referenced book series as well as the list of books in that series, and then using agent-2 (here Amazon Alexa) to make the purchase. While both of these agents interact with the system in natural language, they have different and complementary skill sets,[2] rely on privately held knowledge sources, and have been

[1] https://github.com/allenai/commaqa

[2] but not necessarily mutually exclusive skills

Findings of the Association for Computational Linguistics: ACL 2022, pages 1808 - 1823
May 22-27, 2022 ©2022 Association for Computational Linguistics

built at an enormous cost. At the same time, neither agent by itself can accomplish the original goal.

An alternative to building such a system that interacts with existing agents is to teach all requisite sub-tasks and skills to a large black-box system, say via multi-task learning (Khashabi et al., 2020; Gupta et al., 2021). This, however, not only wastes time and resources, but is often also infeasible. For example, agents such as Google Assistant and OpenAI GPT-3 use private knowledge resources and are computationally expensive to train even once. It would thus be nearly impossible to build a single system with the capabilities of both of these agents.

We note that agents need not be sophisticated AI assistants. An agent may simply be a previously developed question-answering (QA) model, a math module, a function of textual input, an image captioning system—anything the community already knows how to build. The goal is to *learn to leverage existing agents for more complex tasks.*

To enable the development of general systems for this task, we identify the minimal inputs that must be assumed for the task to be learnable—training data for the complex task, existing agents that together can solve the complex task, and examples of valid questions that can be asked of these agents (capturing the agents' capabilities). We build a new synthetic benchmark dataset called CommaQA (Communicating with agents for QA), containing three complex multihop QA tasks (involving Explicit, Implicit, and Numeric reasoning) and four input QA agents that can solve these tasks.

CommaQA is not yet another multi-hop reading comprehension dataset. It is designed to facilitate the development of a new family of techniques that teach systems to communicate with a wide variety of agents to solve different types of complex tasks.

We demonstrate that black-box models struggle on CommaQA even when provided with auxiliary data, such as domain-relevant agent knowledge. On the other hand, a model that leverages the agents (Khot et al., 2021) can achieve very high accuracy but relies on auxiliary supervision (decomposition annotations). While it is possible to identify valid decompositions using just the end-task labels, the search space is extremely large and naïve approaches, as we show, help only with one of the datasets. CommaQA thus serves as a new challenge for the NLP community.

Contributions: We (1) propose a new challenge of learning to solve complex tasks by communicating with agents; (2) develop a synthetic multi-hop QA dataset CommaQA with three reasoning types; (3) provide auxiliary training data and a compositional generalization test set; (4) demonstrate the challenging nature of CommaQA for black-box models; and (5) show the promise of compositional models that learn to communicate with agents.

2 Related Work

Multi-hop QA (Khashabi et al., 2018; Mihaylov et al., 2018; Khot et al., 2020; Geva et al., 2021) focuses on reasoning with multiple facts. Some multi-hop datasets (Yang et al., 2018; Dua et al., 2019) have been used to develop modular approaches such as TMNs (Khot et al., 2021), which are a step towards our goal—they try to solve complex questions by leveraging agents such as single-hop QA models. However, these approaches have had limited success because current datasets are insufficient for the development of such models, for two reasons. First, prevalent single-hop shortcuts (Min et al., 2019a; Trivedi et al., 2020) incentivize models trained on answer supervision alone to learn to exploit these shortcuts rather than learn to compositionally communicate with agents. E.g., they learn to answer a multi-hop question by just asking one single-hop question (Min et al., 2019b). Second, these datasets often contain sub-problems not solvable by existing models/agents, such as producing structured output (e.g., outputting a *list* of all touchdowns mentioned in the context).[3]

Semantic Parsing typically focuses on mapping language problems to executable symbolic representation based on a pre-defined grammar (Krishnamurthy et al., 2017; Chen et al., 2020). Similar ideas are also found in the area of program synthesis (Gulwani, 2011; Desai et al., 2016). These goals, like ours, seek to simplify complex problems into simpler executable forms, without relying on explicit intermediate annotation (Clarke et al., 2010; Berant et al., 2013). We, however, diverge from this line by seeking agent communication in free-form language, not bound to any pre-specified set of operations or domain specific languages.

Question Decomposition is used to solve multi-

[3] For instance, 65% of the errors of the ModularQA system (Khot et al., 2021) on HotpotQA were due to questions unanswerable by existing agents. Hence these datasets don't satisfy the basic task requirement of being solvable using existing agents. This makes the learning-to-communicate task ill-defined over these datasets and meaningful progress infeasible.

hop QA but the resulting models (Talmor and Berant, 2018; Min et al., 2019b; Perez et al., 2020; Khot et al., 2021) are often dataset-specific, rely on decomposition annotations, and limited to one or two QA agents. To address these limitations, our proposed challenge covers three dataset types and four agents. Additionally, models are expected to learn to decompose the task by interacting with the agents, rather than relying on human annotations.

Synthetic Reasoning Challenges have recently been proposed (Lake and Baroni, 2018; Sinha et al., 2019; Clark et al., 2020; Betz and Richardson, 2021) to help systematically identify the weaknesses of existing models and inspire modeling innovation (Liu et al., 2021). Our new tasks are unique and focus on simulating complex agent interaction to motivate the development of decomposition-based modeling approaches.

Text-Based Games, similar to our work, involve interacting in plain text in order to accomplish a goal (Yuan et al., 2019, 2020; Hausknecht et al., 2020; Ammanabrolu et al., 2021; Jansen, 2021). This is typically done in a physical environment, which acts as an “agent” in our setting. Unlike many works in this area, we focus on different classes of compositional questions (e.g, implicit, numerical) and formulate a challenge that makes minimal assumptions about having access to agents’ internal information or input language.

3 Challenge Task Definition

We formalize the new challenge task of *learning to talk with agents to solve complex tasks*. To ensure generality of solutions, we identify minimal inputs for the task to be well-defined and learnable.

First we must define $\{f_i\}_{i=1}^m$, the agents or models that solve simpler sub-tasks.[4] Minimally, we need to define the space of valid inputs $\mathcal{L}_i$ for each agent f_i, i.e., how can they be invoked. For a system to identify the appropriate agent for each subtask, we also need to define the capabilities of each agent. Since these agents are often defined for natural language tasks, the space of inputs captures the capabilities of these agents too. For instance, "Buy the book ‘Harry Potter and the Sorcerer’s Stone’" captures the Alexa agent’s capability of buying books. Instead of complex formal specifications of the agent’s capabilities, we use natural language inputs as a rich and convenient representation.

Next, we need a target task $\mathcal{T}$ that can be solved via a composition of the capabilities of various f_i.[5] Finally, to pose this as a machine learning problem, we need training data $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^N$ for $\mathcal{T}$. Since collecting annotations for complex tasks can be difficult, $\mathcal{D}$ is expected to be relatively small. Models must therefore use the available agents, instead of learning the complex task from scratch.

Given these pre-requisites, we can define the challenge task as follows:

Challenge: Learn a model to solve a complex task $\mathcal{T}$, given only:
- Training dataset $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^N$ for $\mathcal{T}$;
- Agents $\{f_1, \ldots, f_m\}$ that can help solve $\mathcal{T}$;
- Examples from the space $\mathcal{L}_i$ of valid inputs for each agent f_i that captures its capabilities.

One example of this challenge is answering multi-hop questions given two agents: an open-domain TextQA agent f_1 and an open-domain TableQA agent f_2. Agent f_1 can use large textual corpora to answer questions such as "Who directed Kill Bill?". Agent f_2 can use tables (e.g., Filmography tables) to answer questions such as "List the movies directed by Quentin Tarantino". Finally, the training data $\mathcal{T}$ for the complex task would contain examples such as ("What movies has the director of Kill Bill appeared in?", ["Reservoir Dogs", ...,]).

4 Dataset: CommaQA Benchmark

We next propose a new benchmark dataset CommaQA that enables the development of models that can learn to communicate with existing agents. Specifically, we provide a collection of *three synthetic datasets* where each question is answerable by talking to simple QA agents. Note that we are not proposing a new class of questions but a new dataset for the proposed challenge task. A high-level overview of this dataset is shown in Fig. 2.

We choose QA as the underlying task and use QA agents for this challenge because the question-answer format can capture a broad range of tasks (Gardner et al., 2019) while also naturally surfacing the capability of each agent. For instance, the question "What are the key frames in v?" describes a capability of the invoked agent (namely, identifying key frames), in addition to the specific inputs. We next describe our framework for build-

[4] As mentioned earlier, we use *agents* to refer interchangeably to models, assistants, or functions that take free-text as input and produce free-text as output.

[5] Existing datasets lack this requirement, making it impossible to focus only on the agent communication aspect.

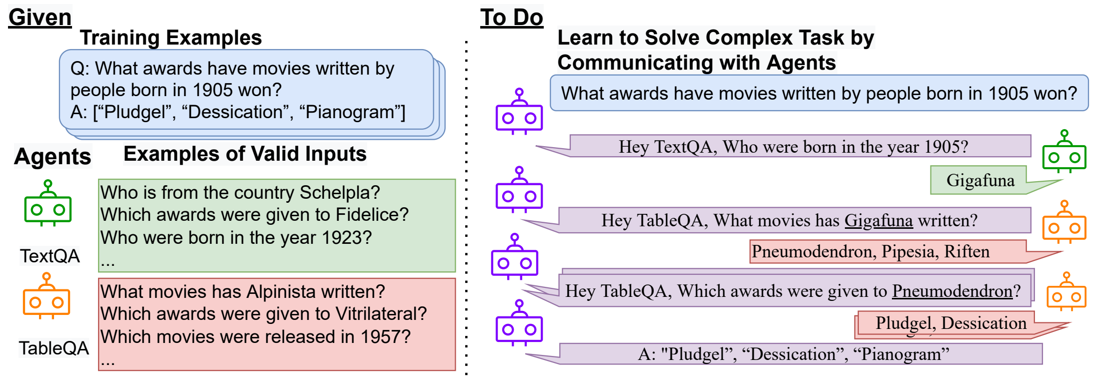

Given
Training Examples
Q: What awards have movies written by people born in 1905 won?
A: ["Pludgel", "Dessication", "Pianogram"]
Agents
Examples of Valid Inputs
Who is from the country Schelpla?
Which awards were given to Fidelice?
Who were born in the year 1923?
...
TextQA
What movies has Alpinista written?
Which awards were given to Vitrilateral?
Which movies were released in 1957?
...
TableQA
To Do
Learn to Solve Complex Task by Communicating with Agents
What awards have movies written by people born in 1905 won?
Hey TextQA, Who were born in the year 1905?
Gigafuna
Hey TableQA, What movies has Gigafuna written?
Pneumodendron, Pipesia, Riften
Hey TableQA, Which awards were given to Pneumodendron?
Pludgel, Dessication
A: "Pludgel", "Dessication", "Pianogram"

Figure 2: High-level overview of the task, with examples from COMMAQA-E. Given the agents, their valid inputs, and training examples for a complex task, the goal is to learn to solve this task by communicating with the agents.

ing COMMAQA, which we believe can be extended to other complex tasks, e.g., video summarization.

4.1 Agent Definition

To define the i-th agent, we build a knowledge base that captures its internal knowledge resource $\mathcal{K}_i$. We use natural language question templates to define the set of questions that this agent can answer over this internal knowledge. For example, given a KB with relations such as "directed(x, y)", the agent would answer questions based on the template: "Who directed the movie __?"

Knowledge Base, $\mathcal{K}_i$. To build the knowledge base, we define a KB schema as a set of binary relations between entity types, e.g., director(movie, person). We build a list of entity names that belong to each entity type. To avoid potential conflicts with the LM's pre-training knowledge, all entity names are generated non-existent words.[6]

Rather than building a static and very large KB, we sample *a possible world* independently for each question, by sub-sampling entities for each entity type and then randomly assigning the KB relations between these entities. This prevents memorization of facts across the train and test splits, which in the past has led to over-estimation of QA model performance (Lewis et al., 2021). This also encourages models to learn proper multi-hop reasoning using the agents, rather than memorizing answers.

Examples of Valid Inputs. To define the space of valid inputs for each agent f_i, we define a set of question templates that can be answered by it over $\mathcal{K}_{ik}$ (e.g., Who directed __?). We construct questions corresponding to a relation in both directions, e.g., "Who all directed __?" and "For which movies was __ a director?". To emulate diversity in natural language, we specify multiple phrasings for the same question. We use these templates to generate examples of valid inputs in $\mathcal{L}_i$ by grounding them with entities of the appropriate entity type (e.g., Who directed Kill Bill?).

To ensure generalization to a broad set of tasks, we do not limit the questions to only single span answers. Depending on the question, the agent can produce answers as a single string (span, boolean or a number), a list of strings (e.g., "Which movies did Spielberg direct?"), or a map (e.g., "What are the states and their capitals in USA?").

Implementation. To answer the question, agents convert questions into queries against their internal knowledge (based on the templates) which we implement as a symbolic function (written in Python), instead of a model. While a language model might be able to generalize to out-of-distribution variations in language, its behavior can be often unpredictable. By implementing the agents as pattern-based functions, we ensure that the resulting systems would stay within the language constraints of each agent and generalize to restricted language models. Additionally, this enables faster development of approaches without spending resources on running a large-scale LM for each agent.

4.2 Complex Task Definition

Given the space of valid input questions for each agent, we construct training examples for the complex task using templated theories. These theories consist of a complex question template and a composition rule expressed as a sequence of questions asked to appropriate agents. For example,

"What movies have the directors from $1 directed?"
#1 = [textqa] "Who is from the country $1?"
#2 = [tableqa] "Which movies has #1 directed?"

[6] https://www.thisworddoesnotexist.com

Operator	Pseudo-code	Example
select	return $f_i(q(a))$	#1=[23, 35] q="Which is largest value in #1?" f_i= mathqa → 35
project	return [(x, $f_i(q(x))$) for x in a]	#1=[Jordan, Johnson] q="What were the lengths of throw by #1?" f_i= textqa → [(Jordan, [23, 34]), (Johnson, [45, 56])]
projectValues	return [(k, $f_i(q(v))$) for (k, v) in a]	#1=[(Jordan, [23, 34]), (Johnson, [45, 56])] q="Which is largest value in #1?" f_i= mathqa → [(Jordan, 34), (Johnson, 56)]
filter	return [x for x in a if $f_i(q(x))$]	#1=[23, 34, 56] q="Is #1 greater than 50?" f_i= mathqa → [56]
filterValues	return [(k, v) for (k, v) in a if $f_i(q(v))$]	#1=[(Jordan, 34), (Johnson, 56)] q="Is #1 greater than 50?" f_i= mathqa → [(Johnson, 56)]

Table 1: Compositional Operators used in this work to transform structured answers into queries answerable by an agent. The operator takes the agent f_i, a structured answer a (we use the answer index, e.g., #1, to refer to any answer), and a query with a placeholder as inputs and executes the pseudo-code shown here.

Composition Operators. While this simple theory would work for single span answers, these agents often return list or map answers. Even within this simple example, there can be multiple directors from a given country and this list cannot be directly fed to the tableqa model, i.e., "Which movies has [...] directed?". This problem gets even more challenging with complex structures. E.g., maintaining a map structure while operating on the values of the map (see 3rd row in Table 1).

To handle the different answer structures, we define a special set of compositional operators in Table 1. These operators take agent f_i, a structured answer a, and a query with a placeholder as inputs, and execute a set of queries (as defined by the pseudo-code in Table 1) against f_i. These operators are inspired by QDMR (Wolfson et al., 2020), but modified to be actually *executable*. E.g., the "project" operator in QDMR: "return directors of #1?" does not specify how to execute this query whereas our operation (project) [textqa] "Who are the directors of #1?" specifies how to use the TextQA model and #1 to generate a map.

We also define a set of agent-independent data structure transformations in Table 2, e.g., convert a map into a list of its keys. Since longer chains of reasoning are prone to more errors (Fried et al., 2015; Khashabi et al., 2019), we don't model these simple transformations as additional reasoning steps. Instead, we concatenate compositional operators with transformations to create about 20 new, combined operators such that transformations can be applied after an operation in a single step, e.g., project_Values operation performs the project operation followed by the Values transformation.

Given these operators, the final theory for the above example would look like:

"What movies have the directors from $1 directed?"
#1 = (select) [textqa] "Who is from the country $1?"
#2 = (project_values_flat_unique) [tableqa] "Which movies has #1 directed?"

Transf.	Procedure
FLAT	Flatten list of lists into a single list
UNIQUE	Return the unique items from a list
KEYS	Return the list of keys from a map
VALUES	Return the list of values from a map

Table 2: Simple transformations that modify the output data structure. These transformations can be chained together with an operation, e.g., PROJECT_VALUES.

Building Examples. Given a KB schema, question templates for each agent, and theories, we can now build examples for the complex task (Fig. 3). We first sample a possible world based on the KB schema. We assign each relation to one of the agents based on which agents are likely to answer such questions, i.e., only this agent would answer questions about this relation. This captures multi-modality of knowledge, e.g., movie awards might be described in text or a table, but a person's birth date is likely described in text. When a relation can be captured by knowledge in multiple modalities, it is assigned to one of them per KB. This emulates the challenging setting where a model must interact with multiple agents to find the answer.[7] We use the templated theories to construct questions by grounding placeholders. We select m valid questions[8] for each KB such that each theory has the same number of examples across the dataset.

4.3 Auxiliary Information

In addition to the basic task definition, we also consider auxiliary information that may be available

[7] With real questions and agents, models may be able to avoid this by just memorizing the agents.

[8] has a non-empty answer and up to five answer spans

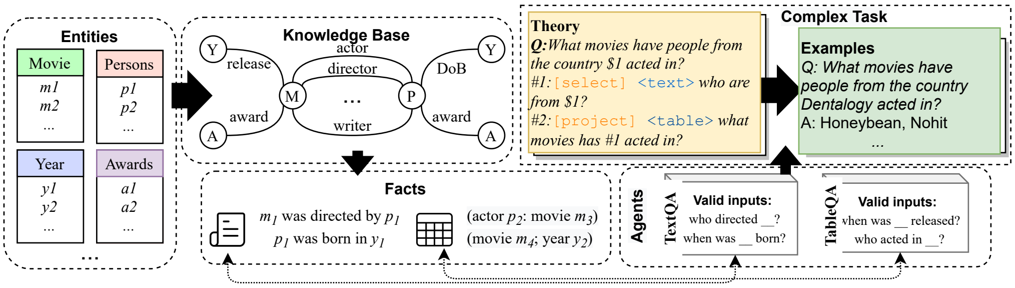


Figure 3: High-level schema of our dataset construction process. We use a list of entities and a KB schema to generate a list of facts. The QA agents operate over these facts to answer a set of pre-determined questions that form the examples of valid inputs from $\mathcal{L}_i$. We define multiple complex question templates and a corresponding *theory* that can be used to answer them. We then ground these question templates (i.e. sample $1) to create complex questions and use the agents to generate the answers.

in some cases. The main goal of this information is to (a) provide stepping stones for development of methods towards the final goal of learning to communicate with agents using answer supervision only, and (b) evaluate the abilities of current state-of-the-art assuming access to this additional information. We emphasize that such auxiliary information may not always be available (e.g., when using a proprietary agents such as Alexa).

We consider two kinds of such information—*auxiliary supervision* for the complex task's training examples $(x_k, y_k) \in \mathcal{D}$, and *auxiliary data* about the agents $\{f_i\}$ themselves (not tied to $\mathcal{D}$). This is summarized in Table 3.

Auxiliary Supervision for $(x_k, y_k) \in \mathcal{D}$:
- Gold Decomposition $\mathcal{D}_k$ for x_k
- Gold Knowledge $\mathcal{F}_k$ for x_k

Auxiliary Data for agents $\{f_i\}$:
- Training data $\mathcal{D}_{f_i} = \{(u_{ij}, v_{ij})\}_{j=1}^{M}$ for agent f_i, where $u_{ij} \in \mathcal{L}_i$ and $v_{ij} = f_i(u_{ij})$
- Complete knowledge resource $\mathcal{K}_i$ used by f_i, or a manageable subset $\mathcal{K}_{ik} \subset \mathcal{K}_i$ containing $\mathcal{F}_k$

Table 3: Auxiliary information as stepping stones towards the full CommaQA task.

For auxiliary supervision, we consider having access to annotated decomposition $\mathcal{D}_k$ of a complex task training input x_k into valid inputs for various agents. We also consider annotated gold facts $\mathcal{F}_k$ that could be used to answer x_k.

For auxiliary data, we consider having access to the training data used to build the agents, or the underlying knowledge base $\mathcal{K}_i$ used by them (and possibly even a question-specific relevant subset $\mathcal{K}_{ik}$). For example, $\mathcal{K}_i$ would be equivalent to the entire text and table corpora used by TextQA and TableQA agents, and $\mathcal{K}_{ik}$ could be the texts and tables relevant to the question domain (e.g., movies). Such information can be used to train a stronger black-box model on the end-task, e.g., fine-tuning on the agent's training data first or using the gold facts to identify relevant context. These approaches that circumvent the agents are not the target of our dataset, but we nevertheless evaluate them to highlight their limits.

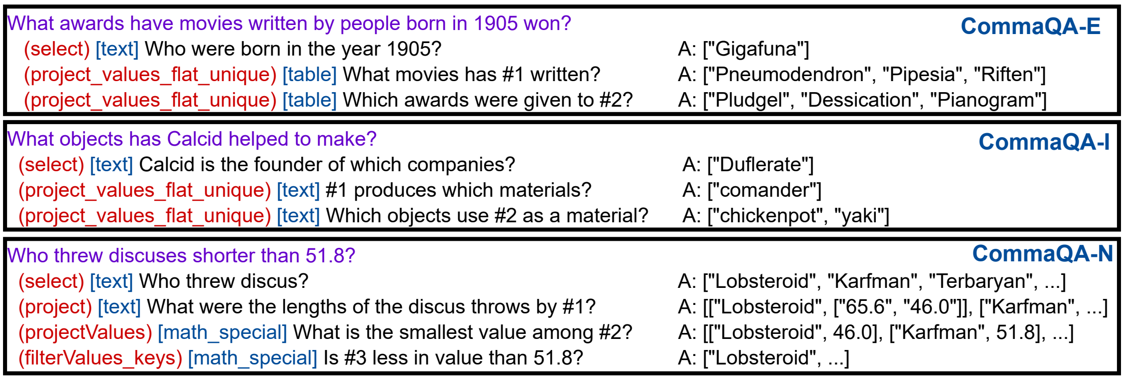
Building Auxiliary Information. We generate the gold decomposition $\mathcal{D}_k$ for each example x_k using the same language as the theories (see Fig. 4). We verbalize each relation to create the underlying knowledge resource $\mathcal{K}_{ik}$ used by the agent f_i (e.g., relation director(M, P) is converted into "M was a movie directed by P" or "movie: M ; director: P" depending on the agent assigned to this relation). While our KB and resulting facts are intentionally simple to show the limitations of black-box models, such verbalization may not always be possible with larger KBs and hence should not be relied upon. For each training example, we collect the facts used by each agent in the decomposition and treat these as gold facts $\mathcal{F}_k$.

4.4 CommaQA Dataset

We use the above framework to build three datasets capturing three challenges in multi-hop reasoning.

CommaQA-E: Explicit Decomposition. This dataset consists of multi-hop questions from the movie domain where the reasoning needed to answer the question is **E**xplicitly described in the question itself (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2021). For example, "What awards have the movies directed by Spielberg won?". We use a TextQA and TableQA agent where certain relations can either be expressed in text or table (more details in App. Fig. 6).

CommaQA-I: Implicit Decomposition. This dataset consists of multi-hop questions where the



What awards have movies written by people born in 1905 won? — CommaQA-E

(select) [text] Who were born in the year 1905? — A: ["Gigafuna"]

(project_values_flat_unique) [table] What movies has #1 written? — A: ["Pneumodendron", "Pipesia", "Riften"]

(project_values_flat_unique) [table] Which awards were given to #2? — A: ["Pludgel", "Dessication", "Pianogram"]

What objects has Calcid helped to make? — CommaQA-I

(select) [text] Calcid is the founder of which companies? — A: ["Duflerate"]

(project_values_flat_unique) [text] #1 produces which materials? — A: ["comander"]

(project_values_flat_unique) [text] Which objects use #2 as a material? — A: ["chickenpot", "yaki"]

Who threw discuses shorter than 51.8? — CommaQA-N

(select) [text] Who threw discus? — A: ["Lobsteroid", "Karfman", "Terbaryan", ...]

(project) [text] What were the lengths of the discus throws by #1? — A: [["Lobsteroid", ["65.6", "46.0"]], ["Karfman", ...]

(projectValues) [math_special] What is the smallest value among #2? — A: [["Lobsteroid", 46.0], ["Karfman", 51.8], ...]

(filterValues_keys) [math_special] Is #3 less in value than 51.8? — A: ["Lobsteroid", ...]

Figure 4: Sample Decomposition Annotations for example questions in CommaQA. We denote the composition operators using the format (operation) [agent] "question".

reasoning needed is Implicit (Khot et al., 2020; Geva et al., 2021), for example, "Did Aristotle use a laptop?". Inspired by such questions in StrategyQA (Geva et al., 2021), we create this dataset using three agents(TextQA, KBQA and MathQA) with just two question styles: (1) "What objects has __ likely used?" and (2) "What objects has __ helped make?". However each question has three possible strategies depending on the context (see App. Fig. 7 for more details). This is a deliberate choice as similar sounding questions can have very different strategies in a real world setting, e.g., "Did Steve Jobs help develop an Iphone?" vs. "Did Edison help develop the television?".

CommaQA-N: Numeric Decomposition. This dataset consists of Numeric (also referred to as discrete) reasoning questions (Dua et al., 2019; Amini et al., 2019) requiring some mathematical operation, in addition to standard reasoning. For example, "Who threw javelins longer than 5 yards?". We create this dataset in the sports domain with TextQA, TableQA and MathQA agents (more details in App. Fig. 8).

Dataset Statistics. The final dataset[9] consists of the three QA sub-datasets described above, key statistics summarized in Table 4.

	CommaQA E	CommaQA I	CommaQA N
#questions	10K	10K	10K
#theories	6	6	6
#steps per theory	2.7	3.2	4.7
#entity types	7	13	5
#relations	11	16	4
#templates in $\mathcal{L}_i$	42	68	30
#entities per answer	3.21	3.29	1.36
#KB facts per KB	169.4	175.7	80
#T5tokens per KB	2252.9	2540.9	1513.4
#Gold facts per qn	7.5	6.9	15.4

Table 4: Statistics of CommaQA. All per-question and per-KB statistics are averages.

There are 10K total examples in each dataset with 80%/10%/10% train/dev/test split. To prevent models from guessing answer spans, we introduce more distractors by sampling a large number of facts for CommaQA-E and CommaQA-I. This results in a larger number of facts in the KB ($\sim$170) and larger length of the KB in these two datasets($\sim$2500 tokens). Since CommaQA-N can have derived answers from numeric reasoning and has longer chains (avg #steps 4.7 vs. 2.7 in CommaQA-E), we do not need a large number of distractor facts (80 facts/KB).

Metrics. The answer y_k to each question x_k in CommaQA is an unordered list of single-word entities.[10] By the design of the dataset, a model that performs the desired reasoning should be able to output y_k correctly, barring entity permutation. Hence, we use *exact match accuracy* as the metric.[11] (see appendix for a softer metric, F1 score)

5 Experiments

We evaluate various models on CommaQA, including a baseline model (with no auxiliary information) for the task and state-of-the-art models that have access to auxiliary information.

5.1 Models

5.1.1 CommaQA Baseline Model

We develop a baseline approach that directly targets the challenge task without relying on any auxiliary information. Specifically, we use the Text Modular Network (TMN) framework (Khot et al., 2021) that trains a `NextGen` model that communicates

[9] released under CC BY license

[10] Although not in the current dataset, entities in the unordered list y_k may be repeated, i.e., we have a multi-set.

[11] Our implementation uses "exact match" in the DROP multi-span evaluator, which accounts for entity reordering.

with the agents. This model is trained to produce the next question (including operation and agent) in a decomposition chain, given the questions and answers so far, which is then executed against the agent to produce the answer for the current step. Additionally this framework samples multiple questions at each step of the chain to search[12] for the most likely chain of reasoning.

We generate the training data for `NextGen` via distant supervision. Specifically, we perform a naïve brute-force search where we sample l questions at each step for up to o steps.[13] The operations are chosen randomly but we only consider the applicable operations (e.g., "select" for the first step). We use lexical overlap between the questions in the examples of valid inputs and the complex question to avoid wasteful random sampling.[14] We assume all chains that lead to the gold answer[15] represent valid decompositions, and use them to build the training dataset for TMNs. We refer to the model as TMN-S$_l$ (see App. B for details).

5.1.2 Auxiliary Supervision Models

We next present models that depend on auxiliary information and hence target a simpler variant of the task: (1) a model trained to communicate with agents using gold decomposition supervision, $\mathcal{D}$; (2) a black-box model trained to answer questions given all the agents' knowledge, $\mathcal{K}_i$; and (3) a two-stage model that first identifies the most-relevant context (using gold knowledge supervision $\mathcal{F}_i$) and uses this shorter context to answer the question.

Models with Decomposition Supervision: Given decomposition supervision, we can directly use this gold data to train the `NextGen` model. We refer to this model as TMN-S when we use this **s**earch and TMN-G when we **g**reedily select the most likely question at each step.

Models with Access to Agent Knowledge: Given access to the facts associated with each (train or test) question x_k, i.e., *each agent's domain-relevant knowledge* $\mathcal{K}_{ik}$, the facts can be concatenated to create a context and frame the challenge as a reading comprehension (RC) task.[16] We train two standard black-box models, T5-L (Raffel et al., 2020) and UnifiedQA-L (Khashabi et al., 2020),[17] to generate answers[18] given a question and context.

Models with Fact Supervision: If, in addition to access to the underlying knowledge $\mathcal{K}_{ik}$, we also have the auxiliary supervision for the gold facts $\mathcal{F}_k$, we can use this annotation to train a model to first retrieve a small subset of relevant facts from $\mathcal{K}_{ik}$ (see App. D.1 for details). Since the context is shorter, we also train a T5-3B model[19] on this task.

Model	Aux. Info	E	I	N	Avg.
TMN-S$_5$		0.0	0.0*	0.0	0.0
TMN-S$_{10}$		17.0	0.0*	0.0	5.7
Auxiliary Supervision Models					
T5-L	$\{\mathcal{K}_{ik}\}$	0.9	10.2	35.4	15.5
UQA-L	$\{\mathcal{K}_{ik}\}$	1.0	10.2	39.0	16.7
T5-L	$\mathcal{F}_k$, $\{\mathcal{K}_{ik}\}$	42.2	49.4	44.7	45.4
UQA-L	$\mathcal{F}_k$,$\{\mathcal{K}_{ik}\}$	40.1	49.7	43.4	44.4
T5-3B	$\mathcal{F}_k$, $\{\mathcal{K}_{ik}\}$	42.3	49.9	43.4	46.2
TMN-G	$\mathcal{D}_k$	75.4	36.0	100.0	70.5
TMN-S	$\mathcal{D}_k$	100.0	100.0	100.0	100.0

Table 5: Accuracy of models trained and tested separately on the 3 datasets. Last column reports average accuracy across the datasets (weighed equally). **TOP** highlighted rows: Target models for CommaQA that try solve the task using no auxiliary supervision by communicating with agents. Naive search is able to generate some training data for CommaQA-E but does not result in any valid decomposition (indicated by $*$) on CommaQA-I. **BOTTOM** rows: Models that rely on auxiliary supervision. Black-box models struggle even when given the domain-relevant KB $\mathcal{K}_{ik}$. Using the additional fact supervision $\mathcal{F}_k$ helps these models, but their accuracy remains below 50%. TMN models with auxiliary decomposition supervision $\mathcal{D}_k$ can solve all tasks with search ("TMN-S") indicating that the task is solvable by communicating with agents.

5.2 Results

Table 5 reports the accuracy of these four classes of models on the CommaQA dataset.

Baseline model has near-zero accuracy: The top two rows represent baseline models that use brute-force search to generate training data for TMNs. For CommaQA-I, we don't find even a single chain leading to the gold answer, and hence no training data. With CommaQA-E and CommaQA-N, we do find valid decompositions

[12] Score is the sum log likelihood of the generated questions.

[13] o is set based on the length of the rules in each dataset, i.e., $o = 3$ for CommaQA-E, $o = 4$ for I, $o = 7$ for N.

[14] We also found random generally performed worse.

[15] We use exact match since the correct decomposition with our error-free agents should lead to exactly the gold answer.

[16] We reiterate that it is often unreasonable to expect access to $\mathcal{K}_i$ and especially $\mathcal{K}_{ik}$. This model tries to solve CommaQA without invoking agents, which deviates from the purpose of our benchmark dataset. Nevertheless, we conduct experiments in this setting for completeness.

[17] We use T5 models as they can handle longer contexts.

[18] We alphabetically sort answers for a deterministic order.

[19] T5-11B performed worse than or same as the 3B model.

for a subset of the questions (see statistics in Table 8 of Appendix), but not enough to train an effective `NextGen` model. Expanding the search to l=20 helps achieve ~100% accuracy on COMMAQA-E (with ~700K agent calls). However, we don't observe any gains on COMMAQA-I and COMMAQA-N with even 2M agent calls (see App. C).

Black-box models struggle even with access to agent knowledge : Due to the large number of distractors, black-box models —even with access to agent knowledge at both train and test time— struggle to learn the task across all three datasets with average accuracy below 20. The extremely low performance on COMMAQA-E is especially notable, given that the reasoning needed for each question is explicitly described. While these models are able to solve similar datasets (Yang et al., 2018), the low scores on our synthetic dataset with more distractors indicates that they are still unable to truly learn this kind of reasoning.

Fact annotations help but are insufficient: Models trained on shorter context (obtained by relying on gold fact training annotation) are able to take advantage of the reduced number of distractors, improving their score to about 45 pts across all datasets. However, even with the larger 3B model, there is no noticeable improvement, indicating 45 pts being roughly a ceiling for these models.

COMMAQA is solvable by talking to the agents: The TMN model, if given gold decomposition annotation for training, can solve this task (bottom two rows). This experiment is an oracle setting that shows that COMMAQA is noise-free, unambiguous, and solvable by a model that learns to talk to the agents (as designed). Note that greedily selecting the next question results in much lower performance on the two datasets (E and I) that have multiple decompositions for the same question.

5.3 Compositional Generalization

We also design compositional generalization test sets COMMAQA-E^{CG} and COMMAQA-N^{CG}. Specifically we create questions using novel composition of queries that have been seen during training but never together in this form. For instance, we create a new question "What awards have the directors of the __ winning movies received?", given that the model was trained on questions such as "What awards have the actors of the __ winning movies received?", "What movies have the directors from __ directed?", and "What movies have people from the country __ acted in?".

Model	Aux. Info	E^{CG}	N^{CG}
TMN-S$_{10}$		16.2	0.0
Auxiliary Supervision Models			
T5-L	$\mathcal{F}_k, \{\mathcal{K}_{ik}\}$	37.0	2.0
T5-3B	$\mathcal{F}_k, \{\mathcal{K}_{ik}\}$	39.2	23.8
TMN-S	$\mathcal{D}_k$	79.4	97.6

Table 6: Lower accuracy on compositional generalization test sets. TMN-S with decomposition supervision still outperforms other models.

As shown in Table 6, all models exhibit a drop in accuracy relative to their score in Table 5, but the compositional model trained on gold decomposition still outperforms black-box models. Our error analysis of TMN-S on COMMAQA-E identified this key issue: While TMN-S learns to generalize, it generates questions outside the space of valid agent inputs (e.g., "Who are the directors in the movie __?" vs. "Which movies has __ directed?").

6 Closing Remarks

We motivated a new challenge of solving complex tasks by communicating with existing AI agents. This challenge, we believe, will help develop more generalizable and efficient models. We introduced a new benchmark dataset COMMAQA with three multi-hop reasoning challenges, all solvable by composing four QA agents. State-of-the-art language models struggle to solve COMMAQA, even when provided with agents' internal knowledge. In contrast, a model that is able to learn to communicate with the agents, albeit using annotated decompositions, is able to solve this task. These results point to the need for and the potential of such approaches, but without reliance on auxiliary annotations, to solve complex tasks.

COMMAQA is only one instantiation of our overall framework. One can extend it in many ways, such as using LMs to enrich lexical diversity, emulating the behavior of imperfect real-world agents that even attempt to answer out-of-scope questions, diversifying to other reasoning types such as Boolean questions where using distant supervision is even harder (Dasigi et al., 2019), and extending the generalization dataset to include new examples of valid inputs as well as new agents.

Acknowledgments

The authors the Aristo team at AI2, in particular Peter Clark, as well as the reviewers for valuable feedback. Experiments were conducted on beaker.org.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *NAACL*.

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *NAACL*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

Gregor Betz and Kyle Richardson. 2021. DeepA2: A modular framework for deep argument analysis with pretrained neural text2text language models. *arXiv:2110.01509*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *ICLR*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *IJCAI*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *NAACL-HLT*.

Aditya Desai, Sumit Gulwani, Vineet Hingorani, Nidhi Jain, Amey Karkare, Mark Marron, and Subhajit Roy. 2016. Program synthesis using natural language. In *ICSE*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Daniel Fried, Peter A. Jansen, Gus Hahn-Powell, Mihai Surdeanu, and Peter E. Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *TACL*, 3:197–210.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question answering is a format; when is it useful? *ArXiv*, abs/1909.11291.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *TACL*.

Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices*, 46(1):317–330.

Tanmay Gupta, Amita Kamath, Aniruddha Kembhavi, and Derek Hoiem. 2021. Towards general purpose vision systems. *ArXiv*, abs/2104.00743.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *AAAI*.

Xanh Ho, A. Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*.

Peter A Jansen. 2021. A systematic survey of text worlds as embodied natural language environments. *arXiv preprint arXiv:2107.04132*.

Daniel Khashabi, Erfan Sadeqi Azer, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2019. On the possibilities and limitations of multi-hop reasoning under linguistic imperfections. *arXiv*, abs/1901.02522.

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 *NAACL*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabhwaral, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single QA system. In *Findings of EMNLP*.

Tushar Khot, Peter Clark, Michal Guerquin, Paul Edward Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *AAAI*.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *NAACL*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, pages 2873–2882.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *EACL*.

Nelson F Liu, Tony Lee, Robin Jia, and Percy Liang. 2021. Can small and synthetic benchmarks drive modeling innovation? A retrospective study of question answering modeling approaches. *arXiv preprint arXiv:2102.01065*.

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:1907.11692*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Sewon Min, Victor Zhong, Luke S. Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *EMNLP*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *EMNLP*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2020. Is multihop QA in DiRe condition? Measuring and reducing disconnected reasoning. In *EMNLP*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2021. MuSiQue: Multi-hop questions via single-hop question composition. *ArXiv*, abs/2108.00573.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, volume 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Blackbox NLP Workshop*.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *TACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Christopher Pal, Yoshua Bengio, and Adam Trischler. 2019. Interactive language learning by question answering. In *EMNLP-IJCNLP*.

Xingdi Yuan, Jie Fu, Marc-Alexandre Côté, Yi Tay, Christopher Pal, and Adam Trischler. 2020. Interactive machine comprehension with information seeking agents. In *ACL*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big Bird: Transformers for longer sequences. In *NeurIPS*.

A Multiple Answers in a Question

If a question refers to multiple answers, e.g. "Is #3 a part of #2?", the operator execution is unclear. To handle such cases, the operator must specify the answer to operate over as a parameter. E.g. (filter(#3)) [mathqa] "Is #3 a part of #2?" would filter the answers in #3 whereas (filter(#2)) [mathqa] "Is #3 a part of #2?" would filter the answers in #2.

B Search Approach Details

We describe in more detail the approach used to build the training data $\hat{\mathcal{D}}$ using the simple search technique. To generate the space of possible decompositions, for each question, we first select f operations from the list of valid operations in Table 7. We only consider these operations as these are the only operators needed for CommaQA. Note that even with this restricted set of operators, models struggle on CommaQA-I and CommaQA-N. Additionally, we only consider the select operation for the first step. For all subsequent steps, we only consider replacements of __ with a previous answer index.

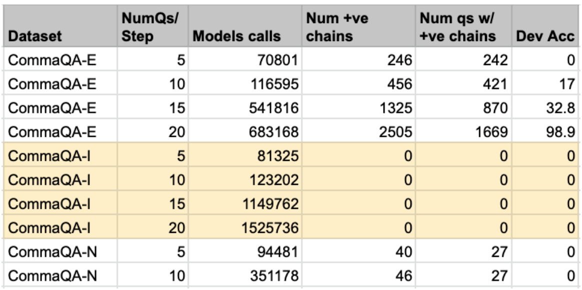
To select the questions, we first simplify the space of inputs by converting the questions into Fill-In-The-Blank (FITB) questions by removing the named entities. E.g "Who was born in 1991?" is changed to "Who was born in __?". This is also a necessary step as the operators need questions with placeholders to handle structured answers. At every step, we expand this pool of questions by replacing the blanks with entities in the complex question and any answer index from the previous steps (e.g. #1, #2 in the third step of a decomposition). To avoid wasteful sampling, we use lexical overlap between questions in this expanded question pool and the input question to identify the top g most relevant questions. The agent associated with each question is tracked throughout this process.

In the end, we consider the cross product between the f operations and g questions to produce $l = f \times g$ total questions at each steps. These l questions are then executed using the appropriate agent and only the successful questions (i.e. answered by the agent) are considered for the next step. This is the key reason why the search space is much smaller than l^o for o reasoning steps.

Table 8 presents the overall statistics of the search approach.

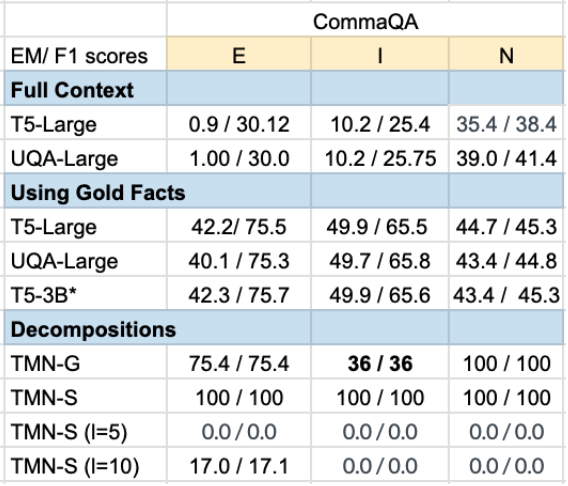
select
filter
filterValues_keys
filter(__)
filterValues(__)_keys
project
projectValues
projectValues_flat
projectValues_flat_unique
project_values_flat
project_values_flat_unique

Table 7: Set of operations considered in the search approach. __ can be replaced by any of the answer indices from the previous steps to create a new operation.



Dataset	NumQs/ Step	Models calls	Num +ve chains	Num qs w/ +ve chains	Dev Acc
CommaQA-E	5	70801	246	242	0
CommaQA-E	10	116595	456	421	17
CommaQA-E	15	541816	1325	870	32.8
CommaQA-E	20	683168	2505	1669	98.9
CommaQA-I	5	81325	0	0	0
CommaQA-I	10	123202	0	0	0
CommaQA-I	15	1149762	0	0	0
CommaQA-I	20	1525736	0	0	0
CommaQA-N	5	94481	40	27	0
CommaQA-N	10	351178	46	27	0

Table 8: Statistic of the search-based approach for different values of l (NumQs/Step). While we get few +ve chains for CommaQA-N, it is not sufficient to train an effective model.



	CommaQA		
EM/ F1 scores	E	I	N
Full Context			
T5-Large	0.9 / 30.12	10.2 / 25.4	35.4 / 38.4
UQA-Large	1.00 / 30.0	10.2 / 25.75	39.0 / 41.4
Using Gold Facts			
T5-Large	42.2/ 75.5	49.9 / 65.5	44.7 / 45.3
UQA-Large	40.1 / 75.3	49.7 / 65.8	43.4 / 44.8
T5-3B*	42.3 / 75.7	49.9 / 65.6	43.4 / 45.3
Decompositions			
TMN-G	75.4 / 75.4	**36 / 36**	100 / 100
TMN-S	100 / 100	100 / 100	100 / 100
TMN-S (l=5)	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0
TMN-S (l=10)	17.0 / 17.1	0.0 / 0.0	0.0 / 0.0

Table 9: EM / F1 scores on the test set using the baseline approaches.

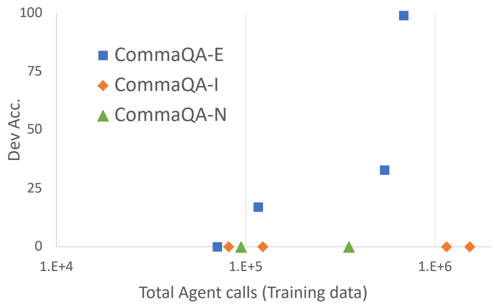


Figure 5: With an order of magnitude increase in search space, we can achieve close to 100% accuracy on COMMAQA-E. However COMMAQA-I and COMMAQA-N need smarter search strategies to generate useful training supervision.

C Search Cost vs Accuracy

One could always exhaustively search for *all* possible decompositions to reproduce the gold decompositions for all the questions. But this would be computationally highly expensive as each call to the agent would often invoke a large-scale LM or a complex AI assistant. To characterize the computational cost of these approaches, we extend the search parameter to include l=15 and l=20 (capped at 5M agent calls) and compute the accuracy of the TMN-S model trained on the resulting dataset (shown in Fig. 5). We can achieve close to 100% accuracy on COMMAQA-E where the search is sufficiently exhaustive(about 700K model calls) mainly due to the shorter rules and the lexical signal. COMMAQA-I and COMMAQA-N, on the other hand, even with an order of magnitude increase in the number of agent calls, we don't observe any increase in the model accuracy.

D Black-Box Models

We train the T5 models on each of the three datasets to generate the answer given the question and facts. We format the input sequence as `<concatenated facts> Q: <question> A:`. Since many of the answers can be multiple spans, we sort[20] and concatenate them into a single string with '+' as the separator. As noted in Table 4, the verbalized facts can result in a context over 2K tokens long. We trained T5-Large models on A100 80G GPUs and RTX8000s to train on such a long context. Transformers designed for longer documents (Beltagy et al., 2020; Zaheer et al., 2020) would be able to handle such contexts more efficiently but generally under-perform due to sparse attention. Hence we don't evaluate them here.

For all T5-based models, model tuning was standardly performed using a random hyper-parameter search in the style of Devlin et al. (2019) using the public huggingface implementation (Wolf et al., 2020); model selection was done based on the highest EM accuracy on the development sets. We specifically experimented with learning rates in the range of ($1e$-$3f$ to $5e$-$5f$) using both Adam and Adagrad optimizers and generally found the settings comparable to the original T5 pre-training parameters (Raffel et al., 2020) to be optimal (Adafactor, lr=0.001, 10 epochs, 0-1000 warmup steps, gradient accumulation was used extensively in place of batching to fit long sequences into GPU memory). The optimal T5-3B models and T5-L for full context on COMMAQA-E were trained with lr=5e-5. All other models were trained with a lr of 1e-3. We will release the complete list of optimal hyper-parameters along with the code.

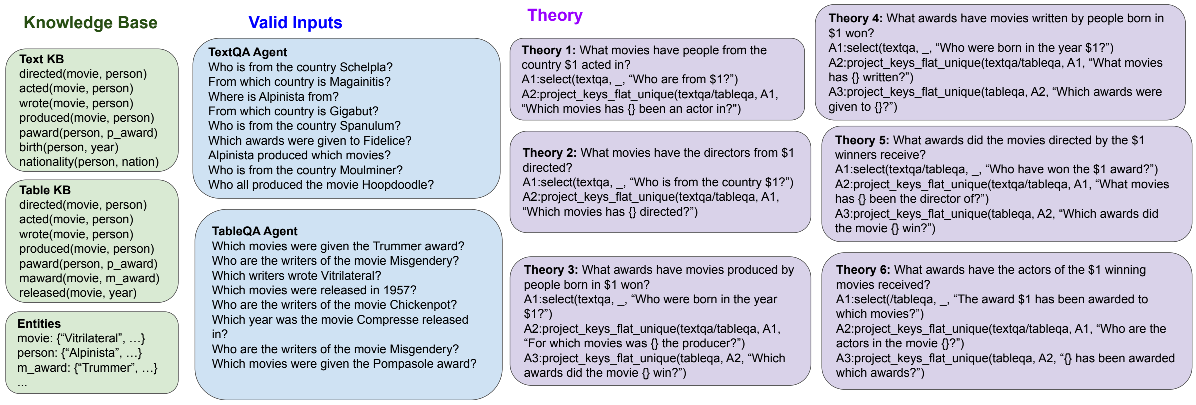
D.1 Models with Fact Supervision

To select the relevant facts, we train a RoBERTa-Large (Liu et al., 2019) model on the gold facts and select the top-scoring facts to produce a shorter context that fits in 512 tokens. The RoBERTa model was training using the AllenNLP library (Gardner et al., 2017) with the standard parameters used for RoBERTa – learning rate of 2e-5, triangular LR scheduler with 10% warmup steps, gradient clipping at 1.0, batch size of 16, 5 epochs of training with patience of 3 epochs. We didn't observe a noticeable difference in score with a random parameter search, so kept these parameters constant. The model was trained to score each fact independently on the train set and the best model was selected based on the accuracy on the dev set. The model was then evaluated on the facts from the train, dev and test set to produce the shorter context for all three sets. The facts were sorted based on the model's scores and the top-scoring facts were added to the context till the number of tokens did not exceed 512 tokens (white-space splitting).

E Text Modular Networks: Training

To train the NextGen model for TMNs, we use the same parameters as the prior work (Khot et al., 2021). We train a T5-Large model as the NextGen

[20] To ensure a deterministic order, we sort the answers in alphabetical order.



Knowledge Base

Text KB
directed(movie, person)
acted(movie, person)
wrote(movie, person)
produced(movie, person)
paward(person, p_award)
birth(person, year)
nationality(person, nation)

Table KB
directed(movie, person)
acted(movie, person)
wrote(movie, person)
produced(movie, person)
paward(person, p_award)
maward(movie, m_award)
released(movie, year)

Entities
movie: {"Vitrilateral", ...}
person: {"Alpinista", ...}
m_award: {"Trummer", ...}
...

Valid Inputs

TextQA Agent
Who is from the country Schelpla?
From which country is Magainitis?
Where is Alpinista from?
From which country is Gigabut?
Who is from the country Spanulum?
Which awards were given to Fidelice?
Alpinista produced which movies?
Who is from the country Moulminer?
Who all produced the movie Hoopdoodle?

TableQA Agent
Which movies were given the Trummer award?
Who are the writers of the movie Misgendery?
Which writers wrote Vitrilateral?
Which movies were released in 1957?
Who are the writers of the movie Chickenpot?
Which year was the movie Compresse released in?
Who are the writers of the movie Misgendery?
Which movies were given the Pompasole award?

Theory

Theory 1: What movies have people from the country $1 acted in?
A1:select(textqa, _, "Who are from $1?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "Which movies has {} been an actor in?")

Theory 2: What movies have the directors from $1 directed?
A1:select(textqa, _, "Who is from the country $1?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "Which movies has {} directed?")

Theory 3: What awards have movies produced by people born in $1 won?
A1:select(textqa, _, "Who were born in the year $1?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "For which movies was {} the producer?")
A3:project_keys_flat_unique(tableqa, A2, "Which awards did the movie {} win?")

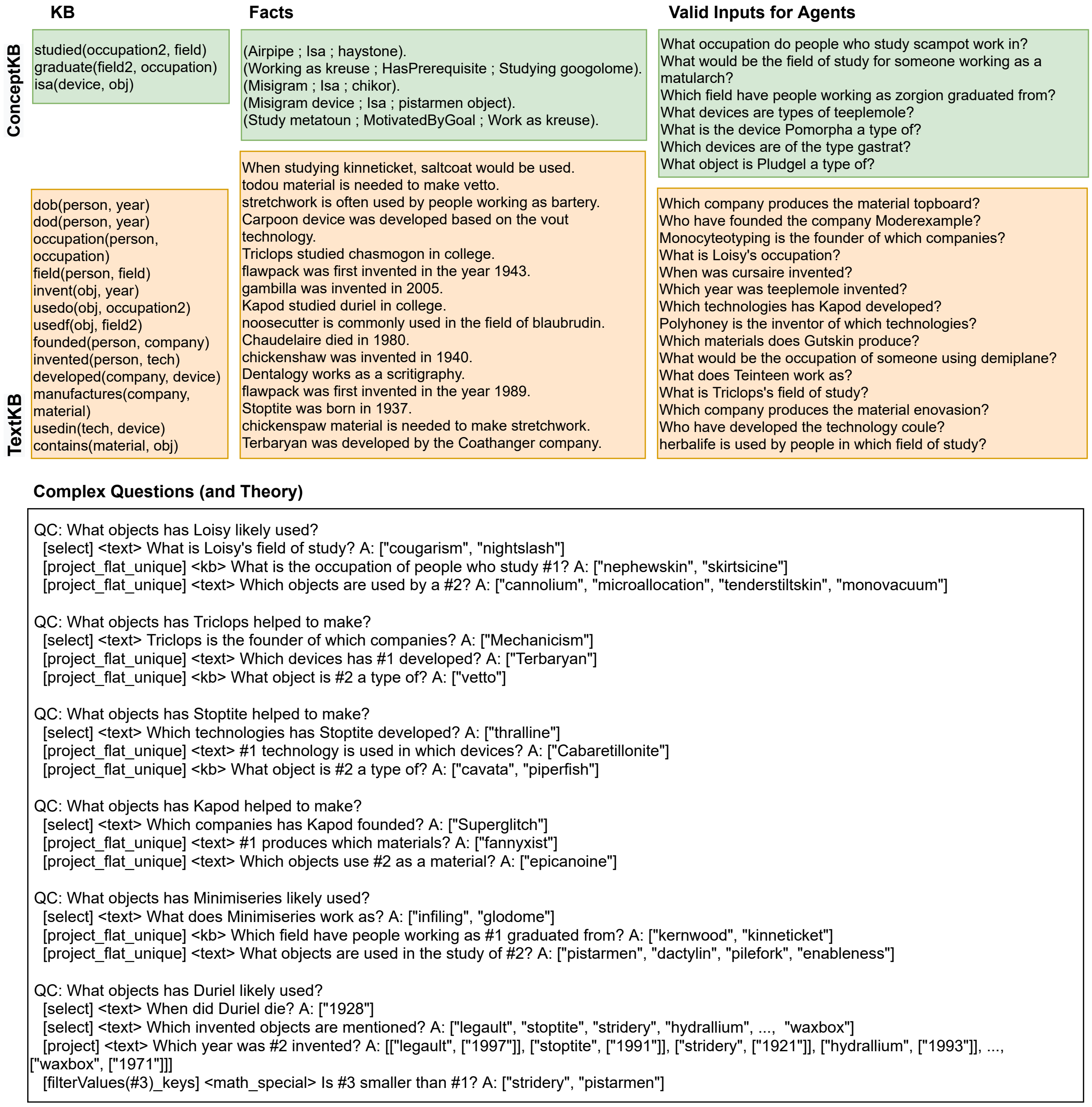
Theory 4: What awards have movies written by people born in $1 won?
A1:select(textqa, _, "Who were born in the year $1?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "What movies has {} written?")
A3:project_keys_flat_unique(tableqa, A2, "Which awards were given to {}?")

Theory 5: What awards did the movies directed by the $1 winners receive?
A1:select(textqa/tableqa, _, "Who have won the $1 award?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "What movies has {} been the director of?")
A3:project_keys_flat_unique(tableqa, A2, "Which awards did the movie {} win?")

Theory 6: What awards have the actors of the $1 winning movies received?
A1:select(/tableqa, _, "The award $1 has been awarded to which movies?")
A2:project_keys_flat_unique(textqa/tableqa, A1, "Who are the actors in the movie {}?")
A3:project_keys_flat_unique(tableqa, A2, "{} has been awarded which awards?")

Figure 6: Example KB, space of valid inputs, and the theory used to construct COMMAQA-E.

model using a batch size of 64, lr of 5e-6, 5 epochs and warmup of 1000 steps in all our experiments. We used the public huggingface implementation (Wolf et al., 2020) to train this model. During inference, we use a beam size of 10 and select 5 questions at each step. We use nucleus sampling with p=0.95 and k=10. For greedy search, we use the same parameters but select one question at each step. We use the sum log likelihood of each generated question as the score of the reasoning chain. (see released code for the exact settings)



KB

Facts

Valid Inputs for Agents

ConceptKB

studied(occupation2, field)
graduate(field2, occupation)
isa(device, obj)

(Airpipe ; Isa ; haystone).
(Working as kreuse ; HasPrerequisite ; Studying googolome).
(Misigram ; Isa ; chikor).
(Misigram device ; Isa ; pistarmen object).
(Study metatoun ; MotivatedByGoal ; Work as kreuse).

What occupation do people who study scampot work in?
What would be the field of study for someone working as a matularch?
Which field have people working as zorgion graduated from?
What devices are types of teeplemole?
What is the device Pomorpha a type of?
Which devices are of the type gastrat?
What object is Pludgel a type of?

TextKB

dob(person, year)
dod(person, year)
occupation(person, occupation)
field(person, field)
invent(obj, year)
usedo(obj, occupation2)
usedf(obj, field2)
founded(person, company)
invented(person, tech)
developed(company, device)
manufactures(company, material)
usedin(tech, device)
contains(material, obj)

When studying kinneticket, saltcoat would be used.
todou material is needed to make vetto.
stretchwork is often used by people working as bartery.
Carpoon device was developed based on the vout technology.
Triclops studied chasmogon in college.
flawpack was first invented in the year 1943.
gambilla was invented in 2005.
Kapod studied duriel in college.
noosecutter is commonly used in the field of blaubrudin.
Chaudelaire died in 1980.
chickenshaw was invented in 1940.
Dentalogy works as a scritigraphy.
flawpack was first invented in the year 1989.
Stoptite was born in 1937.
chickenspaw material is needed to make stretchwork.
Terbaryan was developed by the Coathanger company.

Which company produces the material topboard?
Who have founded the company Moderexample?
Monocyteotyping is the founder of which companies?
What is Loisy's occupation?
When was cursaire invented?
Which year was teeplemole invented?
Which technologies has Kapod developed?
Polyhoney is the inventor of which technologies?
Which materials does Gutskin produce?
What would be the occupation of someone using demiplane?
What does Teinteen work as?
What is Triclops's field of study?
Which company produces the material enovasion?
Who have developed the technology coule?
herbalife is used by people in which field of study?

Complex Questions (and Theory)

QC: What objects has Loisy likely used?
[select] <text> What is Loisy's field of study? A: ["cougarism", "nightslash"]
[project_flat_unique] <kb> What is the occupation of people who study #1? A: ["nephewskin", "skirtsicine"]
[project_flat_unique] <text> Which objects are used by a #2? A: ["cannolium", "microallocation", "tenderstiltskin", "monovacuum"]

QC: What objects has Triclops helped to make?
[select] <text> Triclops is the founder of which companies? A: ["Mechanicism"]
[project_flat_unique] <text> Which devices has #1 developed? A: ["Terbaryan"]
[project_flat_unique] <kb> What object is #2 a type of? A: ["vetto"]

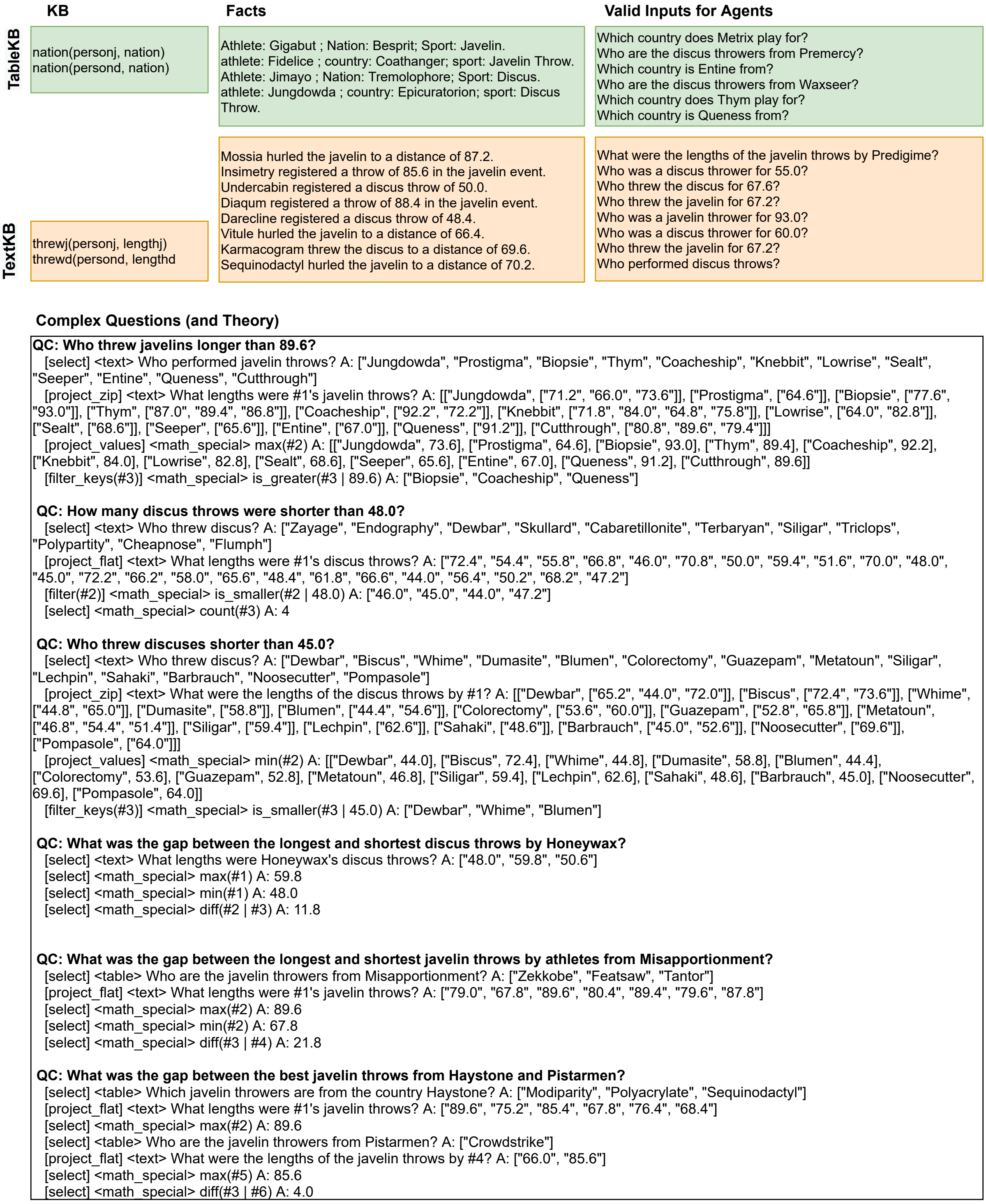
QC: What objects has Stoptite helped to make?
[select] <text> Which technologies has Stoptite developed? A: ["thralline"]
[project_flat_unique] <text> #1 technology is used in which devices? A: ["Cabaretillonite"]
[project_flat_unique] <kb> What object is #2 a type of? A: ["cavata", "piperfish"]

QC: What objects has Kapod helped to make?
[select] <text> Which companies has Kapod founded? A: ["Superglitch"]
[project_flat_unique] <text> #1 produces which materials? A: ["fannyxist"]
[project_flat_unique] <text> Which objects use #2 as a material? A: ["epicanoine"]

QC: What objects has Minimiseries likely used?
[select] <text> What does Minimiseries work as? A: ["infiling", "glodome"]
[project_flat_unique] <kb> Which field have people working as #1 graduated from? A: ["kernwood", "kinneticket"]
[project_flat_unique] <text> What objects are used in the study of #2? A: ["pistarmen", "dactylin", "pilefork", "enableness"]

QC: What objects has Duriel likely used?
[select] <text> When did Duriel die? A: ["1928"]
[select] <text> Which invented objects are mentioned? A: ["legault", "stoptite", "stridery", "hydrallium", ..., "waxbox"]
[project] <text> Which year was #2 invented? A: [["legault", ["1997"]], ["stoptite", ["1991"]], ["stridery", ["1921"]], ["hydrallium", ["1993"]], ..., ["waxbox", ["1971"]]]
[filterValues(#3)_keys] <math_special> Is #3 smaller than #1? A: ["stridery", "pistarmen"]

Figure 7: Example KB, space of valid inputs, and the theory used to construct COMMAQA-I.



	KB	Facts	Valid Inputs for Agents

TableKB	nation(personj, nation)
	nation(persond, nation)
		Athlete: Gigabut ; Nation: Besprit; Sport: Javelin.
		athlete: Fidelice ; country: Coathanger; sport: Javelin Throw.
		Athlete: Jimayo ; Nation: Tremolophore; Sport: Discus.
		athlete: Jungdowda ; country: Epicuratorion; sport: Discus Throw.
			Which country does Metrix play for?
			Who are the discus throwers from Premercy?
			Which country is Entine from?
			Who are the discus throwers from Waxseer?
			Which country does Thym play for?
			Which country is Queness from?

TextKB	threwj(personj, lengthj) threwd(persond, lengthd	Mossia hurled the javelin to a distance of 87.2. Insimetry registered a throw of 85.6 in the javelin event. Undercabin registered a discus throw of 50.0. Diaqum registered a throw of 88.4 in the javelin event. Darecline registered a discus throw of 48.4. Vitule hurled the javelin to a distance of 66.4. Karmacogram threw the discus to a distance of 69.6. Sequinodactyl hurled the javelin to a distance of 70.2.	What were the lengths of the javelin throws by Predigime? Who was a discus thrower for 55.0? Who threw the discus for 67.6? Who threw the javelin for 67.2? Who was a javelin thrower for 93.0? Who was a discus thrower for 60.0? Who threw the javelin for 67.2? Who performed discus throws?

Complex Questions (and Theory)

```
QC: Who threw javelins longer than 89.6?
  [select] <text> Who performed javelin throws? A: ["Jungdowda", "Prostigma", "Biopsie", "Thym", "Coacheship", "Knebbit", "Lowrise", "Sealt", "Seeper", "Entine", "Queness", "Cutthrough"]
  [project_zip] <text> What lengths were #1's javelin throws? A: [["Jungdowda", ["71.2", "66.0", "73.6"]], ["Prostigma", ["64.6"]], ["Biopsie", ["77.6", "93.0"]], ["Thym", ["87.0", "89.4", "86.8"]], ["Coacheship", ["92.2", "72.2"]], ["Knebbit", ["71.8", "84.0", "64.8", "75.8"]], ["Lowrise", ["64.0", "82.8"]], ["Sealt", ["68.6"]], ["Seeper", ["65.6"]], ["Entine", ["67.0"]], ["Queness", ["91.2"]], ["Cutthrough", ["80.8", "89.6", "79.4"]]]
  [project_values] <math_special> max(#2) A: [["Jungdowda", 73.6], ["Prostigma", 64.6], ["Biopsie", 93.0], ["Thym", 89.4], ["Coacheship", 92.2], ["Knebbit", 84.0], ["Lowrise", 82.8], ["Sealt", 68.6], ["Seeper", 65.6], ["Entine", 67.0], ["Queness", 91.2], ["Cutthrough", 89.6]]
  [filter_keys(#3)] <math_special> is_greater(#3 | 89.6) A: ["Biopsie", "Coacheship", "Queness"]

QC: How many discus throws were shorter than 48.0?
  [select] <text> Who threw discus? A: ["Zayage", "Endography", "Dewbar", "Skullard", "Cabaretillonite", "Terbaryan", "Siligar", "Triclops", "Polypartity", "Cheapnose", "Flumph"]
  [project_flat] <text> What lengths were #1's discus throws? A: ["72.4", "54.4", "55.8", "66.8", "46.0", "70.8", "50.0", "59.4", "51.6", "70.0", "48.0", "45.0", "72.2", "66.2", "58.0", "65.6", "48.4", "61.8", "66.6", "44.0", "56.4", "50.2", "68.2", "47.2"]
  [filter(#2)] <math_special> is_smaller(#2 | 48.0) A: ["46.0", "45.0", "44.0", "47.2"]
  [select] <math_special> count(#3) A: 4

QC: Who threw discuses shorter than 45.0?
  [select] <text> Who threw discus? A: ["Dewbar", "Biscus", "Whime", "Dumasite", "Blumen", "Colorectomy", "Guazepam", "Metatoun", "Siligar", "Lechpin", "Sahaki", "Barbrauch", "Noosecutter", "Pompasole"]
  [project_zip] <text> What were the lengths of the discus throws by #1? A: [["Dewbar", ["65.2", "44.0", "72.0"]], ["Biscus", ["72.4", "73.6"]], ["Whime", ["44.8", "65.0"]], ["Dumasite", ["58.8"]], ["Blumen", ["44.4", "54.6"]], ["Colorectomy", ["53.6", "60.0"]], ["Guazepam", ["52.8", "65.8"]], ["Metatoun", ["46.8", "54.4", "51.4"]], ["Siligar", ["59.4"]], ["Lechpin", ["62.6"]], ["Sahaki", ["48.6"]], ["Barbrauch", ["45.0", "52.6"]], ["Noosecutter", ["69.6"]], ["Pompasole", ["64.0"]]]
  [project_values] <math_special> min(#2) A: [["Dewbar", 44.0], ["Biscus", 72.4], ["Whime", 44.8], ["Dumasite", 58.8], ["Blumen", 44.4], ["Colorectomy", 53.6], ["Guazepam", 52.8], ["Metatoun", 46.8], ["Siligar", 59.4], ["Lechpin", 62.6], ["Sahaki", 48.6], ["Barbrauch", 45.0], ["Noosecutter", 69.6], ["Pompasole", 64.0]]
  [filter_keys(#3)] <math_special> is_smaller(#3 | 45.0) A: ["Dewbar", "Whime", "Blumen"]

QC: What was the gap between the longest and shortest discus throws by Honeywax?
  [select] <text> What lengths were Honeywax's discus throws? A: ["48.0", "59.8", "50.6"]
  [select] <math_special> max(#1) A: 59.8
  [select] <math_special> min(#1) A: 48.0
  [select] <math_special> diff(#2 | #3) A: 11.8


QC: What was the gap between the longest and shortest javelin throws by athletes from Misapportionment?
  [select] <table> Who are the javelin throwers from Misapportionment? A: ["Zekkobe", "Featsaw", "Tantor"]
  [project_flat] <text> What lengths were #1's javelin throws? A: ["79.0", "67.8", "89.6", "80.4", "89.4", "79.6", "87.8"]
  [select] <math_special> max(#2) A: 89.6
  [select] <math_special> min(#2) A: 67.8
  [select] <math_special> diff(#3 | #4) A: 21.8

QC: What was the gap between the best javelin throws from Haystone and Pistarmen?
  [select] <table> Which javelin throwers are from the country Haystone? A: ["Modiparity", "Polyacrylate", "Sequinodactyl"]
  [project_flat] <text> What lengths were #1's javelin throws? A: ["89.6", "75.2", "85.4", "67.8", "76.4", "68.4"]
  [select] <math_special> max(#2) A: 89.6
  [select] <table> Who are the javelin throwers from Pistarmen? A: ["Crowdstrike"]
  [project_flat] <text> What were the lengths of the javelin throws by #4? A: ["66.0", "85.6"]
  [select] <math_special> max(#5) A: 85.6
  [select] <math_special> diff(#3 | #6) A: 4.0
```

Figure 8: Example KB, space of valid inputs, and the theory used to construct COMMAQA-N.