Toward a Formalization of QA Problem Classes[⋆]

Naveen Sundar Govindarajulu[1], John Licato[2], and Selmer Bringsjord[1,2]

[1] Department of Cognitive Science
[2] Department of Computer Science
Rensselaer Poltyechnic Institute (RPI)
Troy NY 12180 USA

Abstract. How tough is a given question-answering problem? Answers to this question differ greatly among different researchers and groups. To begin rectifying this, we start by giving a quick, simple, propaedeutic formalization of a *question-answering problem class*. This formalization is just a starting point and should let us answer, at least roughly, this question: What is the relative toughness of two unsolved QA problem classes?

1 Formalization of Question-Answering Problem Classes

It is not an exaggeration to say that the AI community had a watershed moment when IBM's Watson beat *Jeopardy!* champion Ken Jennings in a nail-biting match in 2011. QA is important to AGI: Levesque et al. [7] give an argument in defense of QA being a test for AGI/AI. Despite the importance and quite impressive real-world success of QA research, there is very sparse formalization on what makes a QA problem difficult.[3] (See [4] for a formalization of a test for AI which has a QA format.)

Concisely, our position is that: 1) QA is crucial for AGI; 2) a rigorous formalization of QA is important to understand the relative toughness of unsolved problems in QA; and finally 3) a formal understanding of QA is important for AGI.[4] Toward this end, we start with a simple formalization that could point the way.

We now present a simple formalization of a QA problem class. A QA problem class consists of a set of questions, a set of answers, a corpus, and some other computational artifacts. The formalization lets us judge, at least coarsely, whether one QA problem (e.g. *Jeopardy!*) is tougher than another (e.g. answering queries about financial data).

QA Problem Class

A *question-answering problem class* consists of these following components:

- A set of possible questions $\mathbf{Q}$: $\mathbf{Q} = \{q_1, q_2, \ldots\}$.
- A set of possible answers $\mathbf{A}$: $\mathbf{A} = \{a_1, a_2, \ldots\}$.[a]

[⋆] We are grateful to IBM for grants to Bringsjord that in part enable systematic thinking about the nature of QA, in light of Watson's victory. We also thank them for providing us with an incarnation of Watson that has helped us with evaluating questions similar to the ones given here.

[3] A related criticism by Cassimatis (2012) is that the toughness of existing AI tests (trying to scale mountain peaks on Earth), are nowhere near what is needed for producing human-level intelligence (trying to reach the moon).

[4] QA also happens to be the most effective way of testing other possible forms of intelligence [2], in line with research that treats the pursuit of AI scientifically [1].

- A set of indices **I**. Indices here are entities that deictic words (words that specifiy context such as *"now", "here", "I", etc.*) can refer to. For the sake of simplicity, we will consider only time-points as our indices in the present paper.
- A set of all facts about the world F. The powerset of this set gives us all possible *contexts*: $\mathbf{W} = 2^F$.
- A corpus $\mathcal{C}$ of question-answer pairs

$$\mathcal{C} = \{\langle q_i, a_i\rangle_i | q_i \in \mathbf{Q}, a_i \in \mathbf{A}, i = 1, 2, 3, \ldots\}$$

- Finally, a mapping function μ which gives us the *gold-standard* answers: $\mu : \mathbf{Q} \times \mathbf{I} \times \mathbf{W} \rightarrow \mathbf{A}$. For any question q, possible answers a are given by:

$$\mu(q, t, w) = a$$

[a] Note: Both **Q** and **A** can be infinite.

From this formalization, we immediately get four dimensions of difficulty for QA problems. For now the dimensions are mostly informal, but even at this early stage they illustrate the benefits of seeking to formalize QA. The first dimension emerges from the varying amount of dependence of the answering function μ on the time index t. The second dimension emerges from the varying amount of world knowledge w that the answering function μ depends upon. Such questions have been noted elsewhere by Leveseque et al. (2012) in what they term "Winograd Schemas." The third dimension is generated by the range of novelty. The corpus $\mathcal{C}$ and its usefulness for computing μ plays a role in this dimension. For the sake of simplicity, we assume that $\mathcal{C}$ is a pre-processed corpus of facts that the machine has learned or acquired. The fourth dimension arises from variation of the amount of computational power needed to compute μ. We quickly discuss the four dimensions with help from sample questions.

1.1 Dimension 1: Dynamicity

Most of the questions asked in *Jeopardy!* are static in nature: The answer does not depend on any time component.[5]

President under whom the U.S. gave full recognition to Communist China. (Answer: Jimmy Carter)

The answer **a** to a question **q** can depend on the time t it is asked, the physical location p it is being asked at, and the person a asking it, and other such context-based information. For now, we focus only on the time the question is asked. Even this trivial feature can render some problems very hard. Some example questions of this nature are given below:

Sample Dimension-1 Questions

q_1 What was IBM stock's Sharpe ratio in the last 60 days of trading?

[5] Philosophers and linguists might disagree with us here. Of course, we cheerfully concede that given enough time, the meaning of words used in *Jeopardy!* could change; and for that matter collective human knowledge could change as well.

q_2 Give me the maximum wind speed in New York in the last year, looking at only the days on which it rained.

1.2 Dimension 2: World Knowledge

Some questions can be static in nature but require human levels of intelligence to answer. These usually are questions that any lay-person with appropriate native-language capacity can answer. These questions typically require processing enormous amounts of background knowledge about the world and contextual information. One such problem class which scores high on this dimension is the class of Winograd Schemas.

The Winograd Schema (WS) challenge, introduced by Levesque et al. (2012), is a reading-comprehension test intended to rectify issues with the Turing Test. Two sample questions are given below.

Sample Dimension-2 Questions

q_1 Paul tried to call George on the phone, but he wasn't successful. Who wasn't successful? **Answer 0**: Paul **Answer 1**: George

q_2 Paul tried to call George on the phone, but he wasn't available. Who wasn't available? **Answer 0**: Paul **Answer 1**: George

Although the syntactic forms of both questions are identical, correctly identifying the referent of the pronoun 'he' seems to require deep sentence comprehension, a process exploiting enough background knowledge to recognize that a caller being successful at a phone call requires that the recipient of the call be available.

One of the problems infecting the Winograd Schema Challenge is that it requires the vocabulary of words be specified in advance. A bigger drawback of this dimension is that it tests rather vague, open-ended world knowledge rather than (what might be called) linguistic knowledge.[6] The next dimension rectifies this issue.

1.3 Dimension 3: Novelty

One of the astounding feats of human language understanding is the capacity to assimilate new words on the fly and discard them later. Observing this can be used to create problem classes with completely made-up words that do not need any background world knowledge (e.g. the knowledge about about phone callers and phone call recipients required for the question above). Such questions directly get to the core of language understanding. In the two questions given below, we have made-up nouns in the first case and made-up nouns and verbs in the second case. We can achieve this novelty by mandating that there be little overlap between the words used in $\mathbf{Q} \cup \mathbf{A}$ and $\mathcal{C}$. A more challenging restriction would be to have no questions in $\mathbf{Q}$ overlap with those in the corpus $\mathcal{C}$.[7]

[6] Having a huge bank of knowledge about the world is not sufficient as CYC [6] still does not empower computers to answer all possibe questions. This fact bolsters the claim that leveraging world knowledge is alone exeedingly challenging.

[7] Note: We include knowledge of basic arithmetic in linguistic knowledge. What we term 'world knowledge' might also be called "Earthling" knowledge. Any AGI should be independent of this kind of knowledge but presumably should possess basic arithmetic skills.

Sample Dimension-3 Questions

q_1 If I have 4 foos and 5 bars, and if foos are not the same as bars, how many foos will I have if I get 3 bazes which just happen to be foos?

q_2 Every foobar weozes a foobar if the latter weozes some other foobar. Foobar 27 weozes foobar 28. Is it true that foobar 28 weozes foobar 28?

1.4 Dimension 4: Computational Hardness

The question *"What time is it now?"* is dynamic but not really that hard to compute. Some questions are inherently hard even if they are posed in an unambiguous machine language. This dimension addresses how hard it is to compute μ given all its inputs. There exist hierarchies of hard computational problems from standard computability and complexity theory that could be used as a starting point for this dimension.[8] The hardness of a QA problem class along this dimension can be cast in terms of the most general *oracle* that would need to be used in computing μ to answer questions in the given problem class. The sample questions below are both Turing-unsolvable but fit the general pattern of a QA problem. Both the questions are static, require very little common sense, and the linguistic knowledge required to comprehend the questions is pretty straightforward.

Sample Dimension-4 Questions

q_1 Does machine M with input i ever halt?

q_2 Is this computer program p the same as the program q in my library of programs?[a]

[a] Note: We could easily describe machines, programs, and inputs using words from normal everyday vocabulary.

2 Extensions

The formalization given above works well only when we consider QA in test settings and experiments. If we are modeling QA working outside the lab in the real world, we would need some more components in the formalization.

2.1 Justification of Answers

In the formalization above, we have focused only on the answers being right and not on how the answers are computed. In order to trust an answer, we would need a justification that supports the answer and is easy to check.[9] So a full formalization of QA would include the ability to justify answers. We can modify the formalization to include justifications.

[8] Is intelligence correlated with computational complexity? Some of us think so [3]. Note we are using standard measures of complexity mainly for convenience. They could be superseded by other measures more naturally correlated with intelligence.

[9] The justification would rest on some premises and information that could be considered to have been accepted.

QA Problem Class: Extension 1

- A set of justifications $\mathbf{J}$: $\mathbf{J} = \{j_1, j_2, \ldots\}$.
- An evaluation function $\epsilon : \mathbf{J} \times \mathbf{A} \rightarrow \{\mathsf{true}, \mathsf{false}\}$ which evaluates the justification j given for an answer a. $\epsilon(j, a) = \mathsf{true}$ iff j supports a.

2.2 Ranking of Answers

A QA system in, for example, a personal assistant app would have to learn about and model its users. In this setting, the gold standard answers would vary from person to person. QA systems in such a setting would also have to be evaluated against how well they understand their users.

3 Conclusion

As noted above, our formalization is but a starting point that could help us eventually compare different QA problems. A formal measure of difficulty is needed for QA due to its central role in testing for AGI. While there are tests and competitions for QA (e.g. [9]), a formal measure stemming from the preliminary formalization presented above might help focus our efforts in the right direction and compare different tests and competitions. Such a formalization might also help us decide which methods might be appropriate for different QA problems well before expensive system building and experimentation. (See [8] for a sampling of the widely different approaches to QA.)

References

1. Besold, T.R.: Human-Level Artificial Intelligence Must Be a Science. In: Proceedings of the 6th International Conference on Artificial General Intelligence (2013)
2. Bringsjord, S.: Could, How Could We Tell If, and Why Should–Androids Have Inner Lives? In: Ford, K., Glymour, C., Hayes, P. (eds.) Android Epistemology, pp. 93–122. MIT Press, Cambridge, MA (1995)
3. Bringsjord, S., Zenzen, M.: Superminds: People Harness Hypercomputation, and More. Kluwer Academic Publishers, Dordrecht, The Netherlands (2003)
4. Bringsjord, S.: Meeting Floridi's Challenge to Artificial Intelligence from the Knowledge-Game Test for Self-Consciousness. Metaphilosophy 41(3), 292–312 (2010), `http://kryten.mm.rpi.edu/sb_on_floridi_offprint.pdf`
5. Cassimatis, N.L.: Human-level Artificial Intelligence Must be an Extraordinary Science. Advances in Cognitive Systems 1, 37–45 (2012)
6. Lenat, D.: CYC: A Large-scale Investment in Knowledge Infrastructure. Communications of the ACM 38(11), 33–38 (1995)
7. Levesque, H., Davis, E., Morgenstern, L.: The Winograd Schema Challenge. In: Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (2012)
8. Strzalkowski, T., Harabagiu, S.M. (eds.): Advances in Open Domain Question Answering, vol. 32. Springer (2006)
9. Voorhees, E.M. (ed.): The Twenty-Second Text REtrieval Conference. NIST Special Publication: SP 500-302, NIST (2014)