JORDAN BOYD-GRABER

QUESTIONING ARTIFICIAL INTELLIGENCE

UNIVERSITY OF MARYLAND

tufte-latex.github.io/tufte-latex/

2022

1

Introduction: The Unanswered Question

Since this is a book about question answering, I should try to answer some questions to start things out. I'm going to try to answer: "what's this book about", "who am I", and "who can use this book".

AI has been defined by asking questions even before it was "a thing": Alan Turing designed a question answering proceduring to determine if machine is intelligent. In the years since, AI has continued to be defined by questions: answering questions on *Jeopardy!* and on our phones.

This book looks at not just the *how* of computers answers questions but also *why* answering questions is so important for AI. Answering that "why" questions requires us to go back to mythological, civil service exams, and game shows and connect them to AI question answering.

We call someone smart if they can answer questions: the pass a test, win on *Jeopardy!*, or offer a witty retort in a debate. This is not just a recent phenomenon. Question answering is deeply embedded in our culture: in myth Œdipus answered the riddle of the sphynx; getting into a school depends on passing a standardized test; good governance depends on civil service exams; a trivia subculture has grown around answering all manner of questions.

AI has been defined by asking questions even before it was "a thing": Alan Turing designed a question answering proceduring to determine if machine is intelligent. In the years since, AI has continued to be defined by questions: answering questions on *Jeopardy!* and on our phones. From IBM Watson on *Jeopardy!* to Siri and Alexa in every home, the ability of computers to answer is a common yardstick of how smart computers are.

As we'll see this is important in both contemporary civilization and is fundamental to artificial intelligence. The goal of this book is to help explain how computers answer questions on your phone, on game shows, and to fight fake news. Importantly, from my perspective, this won't just be about computers answering questions: humans and computers have a lot to learn from each other, and I hope we'll be able to discover new things about those interactions.

This book looks at not just the *how* of computers answers questions but also *why* answering questions is so important for AI. Answering that "why" questions requires us to go back to mythological, civil service exams, and game shows and connect them to AI question answering.

To be transparent, this book has an agenda: we can better understand artificial intelligence by looking at its ability to answer questions through this historical lens. If we want to call a computer smart—indeed, smarter than a human—we should make sure the competition is fair. Moreover, writing and asking questions is an art that has been refined over decades.

Part of the story I want to tell is how we got here. And why little things like the decisions made at a tiny British library in Bedfordshire fifty years ago have locked the artificial intelligence community into a particular way of evaluating whether a computer is smart. These historical decisions have shaped the systems that we have today; as a consequence, we are stuck with the crappy answers from our smartphone.

But it didn't have to be this way! The very definition of computing and artificial intelligence was based a thought experiment that is closer to how computers and humans should interact with each other. However, that definition—from a boffin in Manchester called Alan Turing—was too fanciful to really be implemented in the lab in the twentieth century. I'll argue that we could do something closer to Alan Turing's vision, and you, the reader, can judge whether it's (still) too fanciful.

This is not just a narrow question for those building question answering systems. I think that the way we interact with our smartphones and computers will define the future of human–computer interaction and thus the shape of the modern, AI-infused economy.

Who am I?

I'm excited to be writing this because it's something that I'm super passionate about: as you'll see very soon as we talk about the history of question answering in artificial intelligence, answering questions is central not just to my research program but also to my day job: teaching and guiding students. And it's also important to as a hobby: even before this became part of my research program, I was spending my weekends writing and answering silly trivia questions. I'm hoping that this course will help you appreciate the joy and beauty of posing and answering questions.

In my day job, I have built question answering systems that used the structure of Wikipedia to better answer questions, was one of the first to apply deep learning to the QA task, and played a lot of exhibition games against trivia whizzes like Yogesh Raut, Ken Jennings, and Roger Craig. I'm also a professor of computer science, and one of the courses that I teach is a course on question answering (and this book is the textbook for it).

As a hobby, I'm also a trivia enthusiast. I played QB in high school and college, ran Princeton's pub quiz, and appeared on *Jeopardy!*. To be clear, I'm not very good at trivia: our QB teams at best cracked top ten nationally, and I came in second place in the one game of *Jeopardy!* I taped (my students are particularly fond of mocking my poor performance on the video game category). But this has brought me a familiarity with the trivia community and how it works, one that I think gives me a unique (or deranged) look at how computers answer questions.

In contrast, I've won more awards as a researcher. Probably the one I'm most proud of is the Karen Spärk Jones award (we'll learn more about her in Chapter ?reference???) from the British Computing Society, but I've also gotten an NSF CAREER award and "best of" awards from NAACL, CONLL, and NeurIPS. I've published over a hundred peer-reviewed publications on interactive machine learning, question answering, and exploring document collections; and this isn't my first book, with David Mimno and Yuening Hu, I wrote *Applications of Topic Models*. Although I should say that because I'm a professor, this is mostly the work of the students that I've worked with over the years (I thank them by name at the end of the book).

But the biggest prize that I've gotten is to have a career (and permissive bosses) that allow me to combine my hobby with these intellectual pursuits. And I'm even more lucky that that career allows me to now share it with you.

How the Book is Structured

We set the stage in two British Universities in the mid-twentieth century and show how two researchers set the stage for rival paradigms of artificial intelligence in the twentith century. Having set the stage of this rivalry, we then trace the roots of these two paradigms: civil service exams vs. library information desks, game shows vs. standards bodies vs. standardized tests.

Having covered the history, the book turns to the present and the hotly debated question of whether today's AI is closer to Clever Hans or to HAL. We approach this central question through the influence of the Cranfield and Manchester paradigms in three vignettes. These stories detail three modern AI triumphs where companies have used question answering to turbocharge their AI research: IBM's Watson, Google's Natural Questions (and the connection to Muppets), and Facebook's Dynabench leaderboard. But rather than simply recount what these companies did, the perspective of the first section allows us to critically examine the important questions of whether the comparison between human and machines is fair and how much the technical advantages reshaped the field.

Finally, we turn our eyes to the future of AI and QA. How can we take lessons from trivia games and standardized tests to know when we've made progress in comparing human and machine intelligence and to prod research in the right direction.

Contents

1 *Introduction: The Unanswered Question* 3

I Question Answering Past 9

2 *The Mythic Roots of Question Answering* 11

3 *How Question Answering Saved Civilization* 13

4 *The Turing Test: A Game Show Pitch that Defined Artificial Intelligence* 17

5 *The Cranfield Paradigm: How a University with an Airstrip made Google Possible* 21

II Question Answering Present 25

6 *The Computers' Ways of Asking* 27

7 *The Manchester Paradigm: The Art of Asking the Perfect Question* 43

8 *Watson on Jeopardy!: Unquestioned Answers from IBM's tour de force* 57

9 Found QA Data: Are they rough or diamonds? 69

10 Build Your Own Dataset 71

11 Answering Questions through Knowledge Bases 73

12 Machine Reading 75

13 The Advent of Deep Learning 77

14 Machine Reading 79

15 Siri takes the SAT 81

III Question Answering Future 105

16 Human–Machine Collaboration 107

17 Question Answering in Science Fiction 109

18 What AI Dystopias You Should be Afraid of 111

19 Computer Game Shows of the Future 113

Part I

Question Answering Past

2
The Mythic Roots of Question Answering

Despite its central importance to AI, asking and answering questions is not a recent development—it stretches back into our collective unconcious. Riddles and trivia have been with us for millenia not just because they're entertaining; it's more than that. The ability to ask and answer questions is close to godliness. Through the lens of the Riddle of the Sphynx and Gestumblindi, this chapter examines what makes a good question in the eyes of the Gods. Myths from several cultures connect the process of forming and answering critical questions to the process of forming identity, unlocking the secrets of the universe, and gaining intelligence.

Whith many tasks in computer science, asking the question "where did this task come from" has a simple answer usually in the form of an acronym . Character recognition was sparked by the MNIST dataset; entailment was sparked by SNLI; sentiment classification was defined by .

And while there are particular *flavors* of question answering that habe been defined to make life easier or more discriminative for computers, the basic task of question in, answer out is natural to any inquisitive three year old. Of course, question answering is not the only task like this: translation likewise is recognizable to any stranger in a strange land trying to cope with an unfamiliar language: dozens of books have been written on that subject, my favorite is *Le Ton Beau de Marot*.

Question answering's roots likewise run deep. In this chapter, I want to argue that question answering is not just a useful task but one that is woven deep into our Jungian subconsious. Answering questions can define your identity, reveal fundamental truths of the universe, and create a shared reality. This chapter highlights these pillars of question answering and how these questions of identity and knowledge shape not just our modern world but a world where our reality is also partially defined by artificial intelligence.

The Sphynx

In Greek myth, the Sphynx asked everyone who entered the city a riddle. Here's PDQ Bach's interpretation of the riddle:

What starts out on four legs then goes round on two Then finishes on three before it's through

And the answer is:

Well a baby then a man, that's plain, then finally an old geezer on a cane

https://www.youtube.com/watch?v=4sIp80L1dxY 8:30

Now I won't spoil the whole story, but by answering the question correctly, Oedipus revealed that he was smarter than the average bear and also unlocked secrets from his past. In a commencement address to Whitman college, Dana Burgess explained the difference between this and Google's Natural questions and how that relates to the mission of professors (like me):

https://www.whitman.edu/academics/academic-calendar/convocation/convocation-history/convocation-2013/dana-burgess-the-gift-of-the-sphinx

Nobody told Oedipus who he was; he figured that out for himself. ... You can do a Google search to find out the capital of Arkansas (Little Rock?) but you can't do a Google search to find out who you are, what you're good at, what makes you happy, what matters for your life. Information transferal isn't any help with that stuff.

When folks say that a college professor is like a sphinx, they don't usually mean that in a nice way. It suggests that we know the answers students need, but we keep them secret for the sake of our own power. . . . That's part of the riddle of [The Sphynx's] hybrid nature: the curse of being forced to solve a riddle is also the gift of the ability to solve riddles.

https://www.youtube.com/watch?v=20Zk6DWlw8I

3
How Question Answering Saved Civilization

Corruption and nepotism have been the downfall of multiple civilizations, but the administration of carefully formed questions saved the United States and China from sucumbing to these destructive forces. Civil service exams broke the corruption and randomness of patronage and familial bonds. By examining the reforms of Wu Zetian and the Pendelton Act, this chapter examines how information and competence—measured by questions asked in an exam—created meritocracy and social mobility. The legacy of these exams built an understanding of what it means to write a "fair" and "useful" question, and brought about massive social change, allowing egalitarian and professional governments to bring order to continent-spanning civilizations.

I mostly talk about computers answering questions. And I also talk about silly trivia games. As a result, you may think that question answering is unimportant, but it's often deadly serious. Today, I'm talking about how question answering is the bedrock of civilization: humans answering questions is more common and more important than computer question answering. My goal is to remind you how important these question answering applications are and to see connections to how we can learn from milenia of human question answering.

Because I'm a professor, I am somewhat biased in my appreciation of questions: both in the classrooms and on exams. But I hope one thing that I can convince you of is that these are important as foundations of civilization: the world would be a much worse place without these question answering activities.

The Search for Truth

Let's begin with the socratic method. If you haven't heard of this before, you might have experienced it (especially if you're in my classes). Let's hear how John Houseman explain what it is in the 1971 film The Paper Chase.

I want to particularly hone in on the assertion that it "trains your brain". Q: Whose brain is it training? A: I guess not the person asking the question, since they know the answer. So I guess it must be the person asking the questions. Q: Who first started asking questions this way? A: It's named after So-Crates, an ancient Greek philosopher. Q: It's actually pronounced Socrates. But how did it get that name? A: Plato featured Socrates in his dialogues Euthyphro and Ion. Q: And why does a series of questions and answers work in these books? A: It shows the student, the person answering the questions, struggling with multiple interpretations and "showing their work" as they emerge on the other side, enlightened by the exchange.

Okay, enough of that schtick. One of the best moments for me as a professor is to see the aha moment in students eyes after they answer a question correctly. And this is still how we train lawyers, who have to use argument and discussion to keep our society ordered!

Another pillar of civilization is making sure that those who work for society (in other words, civil servants) are competent. As governments became larger and had to rule an entire continent, countries had to develop civil service examinations. Many countries did this, including the US, whose Pendelton act ended an era of political patronage.

ELITE RECRUITMENT AND POLITICAL STABILITY: THE IMPACT OF THE ABOLITION OF CHINA'S CIVIL SERVICE EXAM

But I'd rather talk about the granddaddy of civil service exams, ke1ju3, which for 1300 years determined who became part of the intelligentsia of Imperial China. It encouraged social mobility by getting smart people good jobs and it encouraged good governance by making sure that important jobs were done by smart people. And this is important when you have a diverse, massive, empire that is run by a centralized bureaucracy. Again, this shows that civilization is not possible without effective systems of question answering.

This system worked well for so long because it was built on a foundation of fairness. The organizers thought about the potential for bias: the exams were transcribed so that bad handwriting wouldn't be judged against an applicant, children of current members of the imperial court had to submit their exams in their home province, etc.

Now, it wasn't perfect, Most notably its quotas led to imbalances between regions. But it was important enough that recent scholarship has suggested that its abolition after the Russo-Japanese war and a turn to "Western-style" education on the Prussian model that turns out good soldiers, helped hasten the end of the Dà Qīng dynasty.

As Shiuon Chu argues, it's not just that you have a test to make sure that the smartest people are doing important jobs in society. Part of the process is also to explain why people got the questions wrong! This ensures that people trust how the questions are being graded. If it's just a black box, it is not an improvement on the capricious systems of patronage that it's supposed to replace. As evidence of this, during the Dà Míng dynasty, there was furor over the punctuation of graded exams.

So why am I, a computer science professor, talking all of this nonsense about history? We use questions to test the intelligence of both humans and machines. I talk about the Turing Test, the classic test of computer intelligence, and modern leaderboards in other videos, but in testing computer intelligence we shouldn't forget about the lessons we've learned about human intelligence.

One of those lessons that I think that we've forgotten is that questions are not just an evaluation but also for instruction: in the lingo of artificial intelligence, they're training data. And because these training data build intelligence, they should be as high quality as possible so that the AI that results is as high quality as possible.

But even if you just take question answering as an evaluation for how smart an AI is, society should be able to trust those evaluations. Just as the civil service exams made people trust their interactions with the government, as AI becomes more tightly integrated into our economy and our society, our vetting of AI will become more important.

Question answering is just one of many ways to secure trust, but it's analogous to how to ensure lawyers, doctors, and pilots are qualified for their jobs. So these tests should be unbiased, reliable, and the feedback from the tests should be transparent and understandable.

And we should do these things not just because we're trying to make society work like a well-oiled machine or because we want to be confident in our estimates of statistics about artificial intelligence. Just like Socrates, the questions we ask are trying to get to the truth, and scientific inquiry requires openness and a willingness to question the outcomes of a test.

4
The Turing Test: A Game Show Pitch that Defined Artificial Intelligence

In the 1950s, Alan Turing proposed a parlor game that would come to define the goal of creating human-like artificial intelligence: could a wiley interrogator discern whom they were talking to just through posing clever questions. Alan Turing's eponymous Turing Test is the most durable (but contentious) definition of what it means for a computer to be intelligent. This chapter reviews how Alan Turing's postwar thought experiment of humans asking computers questions shaped the development of artificial intelligence.

The history of artificial intelligence begins in many ways with a question answering game. This idea came out of the research of a researcher at the University of Manchester named Alan Turing.

He's probably better known for being one of the scientists who decoded the Nazi Enigma device (which is why his memorial bench has ciphertext underneath his name) and helped bring World War II in Europe to an earlier conclusion.

But Turing is also a game designer. Quoting from Bishop's description:

Turing called for a human interrogator (C) to hold a conversation with a male and female respondent (A and B) with whom the interrogator could communicate only indirectly by typewritten text. The object of this game was for the interrogator to correctly identify the gender of the players (A and B) purely as a result of such textual interactions

The players can lie, so the key to correctly deciding the genders of the players is more about determining which player lacks key knowledge about the experience of being a man or a woman.

But what does this have to do with AI? Turing then thought, let's replace the man and the woman with a computer and a human. If the interrogator cannot determine which is the computer and which is the machine, then the machine has displayed something that looks like intelligence.

But I want to emphasize not just the broad strokes of the Turing Test, but why we need to think critically about things that call themselves the Turing test. Also, it's more fun than talking abstractly about the Turing test.

Let's start with PARRY, a system designed by Kenneth Mark Colby to simulate a paranoid schizophrenic. And when you connected psychiatrists to either real patients or PARRY, they couldn't tell the difference. So here the problem is the judges. Not because they aren't experts–they are–but because their backgrounds prevent them from being effective judges.

Psychiatrists are doctors bound by the hippocratic oath: they cannot ask probing, in-depth questions that might harm a patient. So, thus, this really isn't a Turing test.

Stu Schieber has a great take on this problem; I'd encourage you to read the whole thing. His take is on a competition called the Loebner prize that purports to implement the Turing Test. But here, the interrogators are limited in the topics they can ask about. So again, competitions that don't have skilled interrogators allowed to ask any question they which.

Another example that some people claim is an example of AI passing the Turing Test is Google Duplex. Here, the judge is fooled into thinking that they're talking to a human. But this doesn't count either because the judge doesn't know they're a judge! The poor employee on the other end of the phone call is expecting a human until proven wrong.

All of this doesn't mean that the Turing Test is flawed. It has remained a part of AI for three quarters of a century because it's a simple, intuitive test of whether we have achieved artificial intelligence. So although we haven't had a real Turing Test yet, a judge asking questions of either an AI or a human remains many researchers' goal.

So let's be true to the spirit of Turing's idea of a parlor game. Let's make it visible to the public, let's refine the rules and the judges to make it more realistic and more fun. By putting these games in the public view and letting judges learn the best strategies for discerning humans from computers, both sides can become worthier adversaries.

And I think putting this slow advance of ever more capable computers answering trickier questions should be out in front of the public. Not just to keep the interrogators honest but to also keep the companies and interests that sell AI honest. The public has a vested interest in knowing the limits of AI, and this is a fine way to make that public. But on the other side of the coin, it is also worthwhile for the public to know when AI has really advanced… the public has a right to know how computers react to challenging scenarios. Better to see them first played out for fun in a game than in high-stakes transactions, a doctor's office, or a courtroom.

But above all, this only works if we have good questions, so if we believe that the Turing test really is the holy grail of AI, we as humans need to know how to ask the right questions and computers need to be able to answer any question that's thrown at them.

5
The Cranfield Paradigm: How a University with an Airstrip made Google Possible

Information retrieval is the foundation of many multi-billion dollar web companies from Baidu in China to Yandex in Russia to Google and Yahoo! in the US. But none of these companies would exist without the ideas of reusable test collections and term-based queries, which came about because of a few crazy experiments that happened in a small UK University in Cranfield. This chapter lays out the history of this methodology (which has become known as the Cranfield paradigm) and how it was used to answer questions.

Often, when I talk about search engines and how to build them, these ubiquitous portals to the Internet—whether you're using Bing, Yahoo!, Baidu, or Ask Jeeves—seem like a fundamental law of science. But it wasn't always like that: there was a time before search engines, and although there's a certain charm to card catalogs, that's not a way to find information.

My goal in this video is to tell the story of the Cranfield paradigm and how it made the twentieth century Internet possible. Most of this is drawn from the article "The Evolution of Cranfield" by Ellen Voorhees, which I encourage you to read something with more detail, more references, and fewer errors.

The story begins in 1967 with Cyril Cleverdon at Cranfield University in England: which has an airport right on campus. Pretty cool! Here at the University of Maryland we have to walk a whole fifteen minutes to get to the oldest continuously operated airport in the world.

Cyril Cleverdon was in charge of their library there. And he was big into computers. He wanted to see if he could measure how good an index was.

Wait a second. When I say "index", what do you think of?

If you're over 35 and went to a public school, you might think of this sort of thing in the back of a book. But if you're younger and a computer scientist, you probably think of an index as a lookup table of every darn word possible.

The reason you think that is because of the Cranfield experiments. His claim, which was quite controversial at the time, was that you could look up documents based on the words in the document. You didn't need laboriously curated indices you could use a machine, an engine if you will, to search through them.

The academic community at the time was skeptical. There's no way these newfangled search engines could be better than a trained librarian armed with a card catalog!

Before we analyze this, just a reminder of how a search engine query is evaluated. Again, this is old hat now but was revolutionary at the time. To evaluate a system's ability to search a dataset, you take some queries from users and then find the truly relevant documents to that query. Then, given a system's results, you compute the precision—out of all of the documents they returned how many were right—and recall—out of all possible right documents how many did they find.

Let's pause for a moment to recognize how revolutionary this was. Once you found the relevant documents, you can evaluate *any* system. Before, you had to run an expensive user experiment every time you tweaked your system: did that output look good? How about this one. With a reusable test collection, you could turn the crank on your retrieval system without pesky humans getting in the way. Just check whether the red box finds the relevant documents!

But not everybody was convinced. The Cranfield paradigm is making a lot of assumptions here: first, if a document is relevant, it's equally relevant: everything is equally relevant: showing you my explanation of topic modeling is as good as a video from Siraj Raval's The users' information needs are static; the answer to the question "who is the president of the united states" never changes, after all The next assumption is that all users are the same: when I ask the question "why is the sky blue", I should get the same answer as my six year old daughter Finally, the Cranfield paradigm that somebody, somehow can find the right answer from thousands or millions of documents.

Despite all those flaws, the assumptions—on average—mostly hold. That's why we're still talking about the Cranfield paradigm fifty years later. And there were some big additions to the Cranfield paradigm. Most prominent was the use of better document representations and creating larger test collections, which was spearheaded by Karen Spärk Jones a little bit west of Cranfield at Cambridge. These took the theoretical insights of the original Cranfield experiments and made them into something that could actually work. Again, this was revolutionary: as a result the British Computing Society named their annual award for researchers in this area after Karen Spärk Jones.

But for the Cranfield paradigm to really shape how we get answers from the internet, we need to leave central England and meet new player: NIST. The American National Institute for Standards and Technology.

While the story of the birth of the Cranfield paradigm is an innovative insight with Cyril Cleverdon combined with the vision of Karen Spärk Jones, the story of NIST's development of these test collections is one of slow, methodological iteration. Building up decades of expertise and slowly building up bigger and better test collections, updating them each year so that by the time Google's PageRank came around, there's no question about how to tell whether a search engine is good enough.

Just fire up the latest test collection from the TREC conference, compute your precision and recall, and call it a day.

And this approach toward evaluation is with us in everything we do. Test collections are how we evaluate virtual assistants, question answering systems, and just about every aspect of the modern Internet. And it's not always for the best. The ubiquity of these reusable test collections leads to overfitting. TREC puts out new datasets every year or so, but that's not fast enough for machine learning in the 21st century! And while the focus on real user data is good, the data you collect is biased by who is using technology (mostly rich people from Western countries) and how they've been trained to ask questions through decades of living with Cranfield-paradigm systems. And unlike people who work on, say, human-computer interaction, those building QA systems often don't actually put their system in front of real users. And I don't think that's what Cyril Cleverdon would have wanted. His goal, as a librarian, was to help users. And yes, reusable test collections help you do that—and they've spawned multibillion dollar companies around the world—but at the end of the day you still need to check whether real users are actually finding the information they need.

JBG: Forward point to leaderboard chapter

Part II

Question Answering Present

6
The Computers' Ways of Asking

How do you ask a question: is it text, a picture, or a conversation? This chapter reviews the different forms question answering can take and what complexities that can introduce.

Conversational QA The goal of having intelligent computers is a part of both the foundation of artificial intelligence in the form of the Turing Test and science fiction.

But you don't interact with these computers through a command line or a search prompt. You talk to the computer! And the computer remembers what you say, it has situational context to understand, resolve ambiguities, and place your questions in context. This doesn't look a lot like the existing question answering settings that we've discussed before. To answer questions in a conversation, we'll need new datasets, new evaluations, and new models.

Let's start with the datasets. I mentioned some of these very briefly in the datasets video, but now we can linger a little bit longer on these conversational QA datasets. There are others, but I wanted to talk about: QBLink, CoCa, Quack, and CANARD.

Now you might think that conversational QA might mean that you might get a respite from my Quizbowl fixation, but not so fast! The trivia community is also a source of found data for conversation-like questions. With my former students, we created a dataset of questions these questions start with a sentence to set the context and then you have to answer three (usually) questions within that context.

Other than the Manchester-style questions and answers, there's nothing special like pyramidality that makes them "optimal". I just hink they're neat.

For example, you have a leadin that gives the context. "Only twenty one million units in this system will ever be created. Name this digital payment system whose transactions are recorded on a "block chain". Now, notice that you cannot answer this question *just* from the literal question ... it depends on the context. There are plenty of cryptocurrencies (more every day), but the 21 million number is unique to

Bitchoin.

Then we move on to the next question: it was invented by this person, who, according to a dubious Newsweek cover story, is a 64 year old JapaneseAmerican man who lives in California. Now, you could probably answer this just from the clue here, but it helps to know that "it" here is Bitcoin. Otherwise you have to do a multihop to figure out that this is Bitcoin and it was invented by someone who purported to be Satoshi Nakamoto.

Finally, the final question is much easier if you know that Bitcoin is involved.

Let's move on to the next dataset, QuAC (Question Answering in Context) which appeared at the same time as our dataset and is noticeable for introducing emojis in paper titles. Two of my former students—He He and Mohit—worked on this Cranfield paradigm dataset, where two crowdworkers play the role of a teacher and a student. The teacher gets to see a Wikipedia page but the student does not. They have a conversation and the student tries to figure out as much as possible about that topic.

And just like the QB link dataset, you need to do things like resolve coreference, know the topic of the questions, etc.

And again, one of the big plusses is that the answers are all anchored in the Wikipedia page (in this case Daffy Duck), so there's gold evidence that useful for machine reading techniques. But the challenge is that you also need to embed the context somehow!

One approach would be to just transform the conversational, context-specific questions into "normal" that look more like NQ or SQuAD, so we had crowdworkers rewrite the questions into forms that don't require the context. We called this dataset Canard, short for Context Abstraction: Necessary Additional Rewritten Discourse, where Canard is of course the French word for "duck".

And by conditioning on the entire context, you can indeed create questions that don't need additional context, but it's not as good as the human rewrites.

The final dataset I'd like to talk about is CoQA, which looks a lot like QuAC—it came out a year after QuAC. But rather than anchoring on a Wikipedia page, it is anchored on a short vignette, here it's an 80th birthday party. Like in QuAC there's a student and a teacher and the student asks questions of the teacher. But unlike QuAC there are also rationales that explain why a question has a particular answer.

One thing that you'll notice about these datasets is that they're not really a true conversation. It's kind of one sided. And part of that is the tricky problem of evaluation. You need datasets to train models . . . but to train a model you need a loss function. To have a loss function you need to know if an answer is correct.

In some cases, the evaluation can look a lot like our evaluations for "normal" question answering. You just need to give an entity. Then you can use the same metrics that we saw for, say, machine reading question answering: entity equivalence, exact match, span F1, etc. That's what you can do for the datasets that we've talked about thus far.

But is a conversation just spouting the entities? No, of course not. You can't just have a conversation just through cultural references.

As we'll talk about in the long-form question answering segment, evaluating longer answers is really hard.

So are there ways that you can make the problem easier within a conversational context? Research in dialog systems typically divide the task into two phases: figure out a dialog act that fits in the context of the conversation, and then given the correct dialog act, you then need to generate the raw text that actually goes to the user.

So what's a dialog act? There are many ways to define dialog acts, here's one example of an ontology of dialog acts created from real-world conversations by a team from Colorado and SRI.

You need to be able to answer a question with yes or no, ask those questions, ask "wh" questions like "When was your last training session". And because conversations aren't just information exchange sessions, sometimes you need to agree, give an opinion, or just let the partner know you're listening.

It's sometimes lonely just talking into the void with nobody responding to you . . . what dialog act does a comment correspond to?

This was adapted by researchers from Virginia Tech to make it more detailed. Sometimes you need to give an answer with an explanation, sometimes you need to answer with a followup clarification question, or say that you don't know.

So now, to do an evaluation, you can make sure that your system can reconstruct the dialog acts given the context. This turns it into a classification task: does your predicted dialog act match the ground truth? This is much easier than actually evaluating long, rambling answers, which as we'll talk about with generation is actually very difficult.

But even so, this is not without complexity. These datasets have the interactions between two people where presumably everything went according to plan. But what happens if during evaluation if you select a dialog act that's different from the reference interaction?

The thing with conversational question answering is that mistakes are not the end of an interaction. Ideally, a conversational system should recover from it! Let's take a look at this reference conversation where the user asks the system what the capital of Georgia is. The system should give an answer, and the user can give a followup question.

But what happens if the system gets the question wrong? For example with a wrong interpretation of which Georgia the user is talking about. Well, if the system gives that answer, the user can react in all sorts of ways! Perhaps they realize they cared about Georgia the country all along (this is probably the most likely outcome if, say, your "users" came from mechanical Turk and they're pretending to ask questions), perhaps they correct the system, or perhaps they want more information: why did you say "Tblisi"?

In any event, we need to know how bad it is that the system said Tblisi. And we can't really do that without testing the system on real users. Maybe it isn't great that we said Tblisi, but it's better than saying that the capital of Georgia is "Peachtree" or "Washington". But again, we don't know that this is a "near miss" because, again, it's not in the reference.

And it can go in the other direction too! Let's say that the way we generated the reference data was quite a bit worse than what an "optimal" conversation partner could do. Let's say that your system does something better than the reference. Perhaps it asks which Georgia you mean. That's even better!

Just because you do the conversation a little differently than the reference, doesn't mean that it's wrong.

So the challenge is to define an evaluation that measures how useful an interaction is, not the answers of individual turns. I suspect that this will need to take the form of human-centered evaluation. You need to have people interact with a real system and get them to evaluate how well it works.

And we'll see something like that in our next video, where we'll talk about another way of moving beyond single, short answers: what happens if the answers themselves are pretty darn long?

Long Form QA

As usual, let's start with how complex questions are answered in the real world. Let's start with Apu Nahasapeemapetilon's answer to "What was the cause of the US civil war?"

Examiner : All right, here's your last question. What was the cause of the Civil War? Apu : Actually, there were numerous causes. Aside from the obvious schism between the abolitionists and the anti-abolitionists, there were economic factors, both domestic and inter... Examiner : Wait, wait... just say slavery. Apu : Slavery it is, sir.

Like the examiner, it's often easier to go for the shorter answer . . . it's easier to score, anway. There are many ways to answer "What was the cause of the American Civil War". Everything that Apu said was correct, and there are many ways to phrase what he said. Detecting all the things that are semantically equivalent to that is itself a NLP-complete problem. To understand how, despite that difficulty, we're trying to build systems that can answer long complicated questions, we'll go through two datasets that can help computers answer hard questions and we'll see what makes them both hard to answer and hard to tell when the answer is right.

Let's first begin with Complex Answer Retrieval, often called "CAR" for short. This is a dataset from TREC and Laura Dietz at the University of New Hampshire. If you haven't seen our previous video about Like many of the other question answering datasets we've talked about, the source information comes from Wikipedia.

They start with questions like "How do you make Candy" or "Is chocolate healthy" and run a search over all of Wikipedia. Like with traditional relevance judgements, annotators then mark whether the passages retrieved are a good answer for these questions. NIST annotators mark whether a passage *must* be included in the answerset, *can* be included, is on topic (but not a good answer) or just irrelevant.

Now, getting annotators to agree on this isn't trivial: if you look at the kappa statistics, it's between 0.5 and 0.7. You can then run the standard precision / recall statistics for an IR engine. So is it the case that long form QA isn't actually that hard?

No, it's still that hard! This can only work if Wikipedia has the perfect answer for every question already written down. But although Wikipedia is pretty big, it's not *that* big.

Another dataset that tries to address the "long answer" problem is the ELI5 dataset. This is built from a subreddit where people pose questions and people try to answer them using simple language. What I like about Reddit is that there are clear rules about what questions are allowed, how they can be answered, and then people vote on which of the answers is best.

Despite these beautiful rules and style guidelines, the questions aren't as beautiful as the best examples of the Manchester paradigm, which are carefully edited and polished until perfection. However, they're much better than the grabbag of the worst of the Cranfield paradigm, since there are clear rules: you can't ask questions that are actually simple, the questions can't be subjective, and the questions must be about the "real world".

So now we have examples of questions and good answers. But we can't just test whether the two strings are equal or not . . . so how do you know if the output of a system is good or not?

This is actually very similar to the problem faced in machine translation: you want translations to be as close to a reference translation as possible. So they developed a metric call BLEU, or Bilingual Evaluation Understudy that counts up how many words overlap between a reference translation and system outputs.

It's called an "understudy" because it's meant to mirror the adequacy and fluency judgements of human annotators.

But we're not talking about machine translation today, but you can use a very similar idea for comparing two pieces of English text. So BLEU in French means "blue"; the other metric we'll be talking about is ROUGE, which means "red" in French, and stands for "Recall-Oriented Understudy for Gisting Evaluation". Unlike BLEU, which tries to get the translation exactly right—in other words getting precision and recall high—ROUGE focuses on recall.

Let's see how it works. Like BLEU, ROUGE looks for a certain number of n-grams. Let's start with bigrams for simplicity. Let's say you want to compare these two sentences. First, you gather up all of the bigrams. One of the four matches, so this means that it would have a ROUGE-2 of ¼.

But ROUGE-N requires choosing what N should be. A more popular variant is to use the "longest common subsequence" or LCS. Finding the longest common subsequence is a popular example problem when you're first learning dynamic programming . . . while writing a program to compute it can be a little tricky, for short sentences you can fairly easy eyeball: what is the longest sequence of words that is in *both* sequences. For these two sentences, that would be "parents annoy the children".

Given that LCS, we can then compute the precision and recall and then we get the ROUGE-L measure. One nice thing about this is that it's upper bounded by the unigram overlap between the two strings.

We spent a lot of time talking about ROUGE-L . . . does that mean that this is the right way to compute whether a long answer is a good one or not? Compare the generated answer to the reference ELI5 answer? A recent paper from my former student Mohit shows that . . . uhh maybe not!

ELI5, like a lot of other Cranfield QA datasets, have a lot of overlap between the training and test set (this is less of a problem in the Manchester paradigm, as the authors explicitly avoid repetition). Thus, the system doesn't always learn to generalize . . . it can memorize instead.

I'm not going to talk about their system much—it's a good one, check out the paper linked in the description—but like you might do for the CAR task, it also has a retrieval component. But it doesn't really matter what you retrieve! Getting random retrievals can add in some specific words that can up your ROUGE-L score.

And it also seems that you can game the system with really stupid baselines that *shouldn't* get a good score and end up . . . doing quite well. For example, just copying the answer over and over again or copy/pasting a random training answer sometimes beat the gold answer.

So this makes it hard to prove that your snazzy new algorithm is actually good. So what's the answer? I think that question answering always will depend on human evaluations, but while automatic factoid question answering evaluations are often "good enough", I don't think—as I record this in early 2022—we can confidently say that we have a "good enough" automatic evaluation with long for question answering.

So that means that you really need to do a human evaluation. There's nothing wrong with that! Again, I wish that this were more common for question answering. But for our modern data-hungry algorithms and our fast-paced research culture, projects don't always have the luxury to do evaluation right . . . with the finicky, expensive human evaluations of these very long answers.

So what's the right way to quickly evaluate QA systems that output more than a couple of words? Well, I have the answer right here . . . unfortunately it's this video is too small to contain it. Answering questions about images

Mostly we've been talking about computers that can talk to you and answer questions about text. But is that the extent of our ambitions? Take a look at this clip from Kubrick's 2001:

Hal 9000: Have you be doing some more work? Dave Bowman: A few sketches? Hal: May I see them? Dave Bowman: Sure. Hal 9000: That's a very nice rendering, Dave.

And even beyond the fanciful dreams of future AI, there are people today who could really use help answering questions about images. For example, the VizWiz project from Jeff Bigham connected blind users with a QA system over images. Here's Tom Dekker, who is blind, showing off VizWiz.

Tom Dekker: So now we have VizWiz, which I'll open. Phone: Use the camera button to take a new photo. Tom Dekker: I'm going to get positioned above these things so they're nicely viewable. Phone: Start and Stop Recording a Question Tom Dekker: What color are these two socks? And what color is the one that's folded in half?

As you'll see compared to the other datasets that people are training QA systems on, we're nowhere near the ability to answer these questions. The VizWiz project is what's called a "Wizard of Oz System": it's a human pretending to be a machine.

Oscar Zoroaster Phadrig Isaac Norman Henkle: Pay no attention to that man behind the curtain. The great and powerful Oz has spoken!

And these workers are hired on Amazon Mechanical Turk, named after another human pretending to be a machine.

Tom Dekker: It took about a minute and a half. Now let's see . . . Phone: Folded sock is black. The other one is half white, half gray, divided down the middle. –Webworker Phone: New answers available.

And there are other cases where it's not a question of ability but of volume: you might be responsibility for watching lots of camera feeds and a visual QA system could quickly narrow down things that are interesting from huge quantities of data. For example, I could look at my doorbell footage and ask . . . when did my newspaper come today or who took my Amazon package?

So let's take a step back from those fanciful ambitions and take a look at the "real" datasets that are out there for multimodal question answering. The most prominent is "VQA" or visual question answering, which has become nearly synonymous with this girl wearing a banana mustache. This dataset set up the paradigm of looking at a picture and answering the question.

The crowdworkers who wrote the questions in the Manchester paradigm (since they knew the answers) were instructed to write questions that a human could answer but that a robot could not. While it has the flavor of an adversarial setting, there wasn't a specific adversary . . . the workers had to have their own mental model.

However, just like many of the datasets we've looked at, there were many shortcuts that the models could take: if the question asked "is there a" the answer was almost certainly yes, and if there was a sport involved, it was usually about tennis.

So to correct these issues, VQA 2.0 paired two pictures together so that the answer to the same question is different in each of the pictures. So for instance here the man or the woman is wearing glasses. This makes it so that the dataset is intrinsically balanced and the model can't just win through a text only baseline.

So can models answer these questions? The accuracy has been climbing by 14 points over the last four years . . . getting near 80% accuracy. So what models are behind these impressive numbers?

Like with text, transformers with Muppet names like Oscar, BERT, and Ernie are to blame. But how do transformers deal with *visual* input? Thus far, we've only been talking about words and word embeddings. Just like you need to split a sentence into word pieces, the first step is to break an image into pieces to correspond to objects.

Now, I don't know how this works and even if I did, it would be out of scope for this course, so I won't talk about how all these fancy boxes come about. But once you have these boxes, you can then embed these boxes through a linear or non-linear transformation of the constituent pixels which—unlike words—are already continuous vectors. Just like words have an embedding based on the word's type and the token's position, the visual word embedding also is a function of the pixels and its location.

So then you can get a combined visual and text representation of an image, caption or image, question pair. And because a transformer isn't as tied to the strict linear order of a sentence thanks to its attention map, it doesn't really matter in what order you plop down the objects.

Okay, so how do you pretrain transformers with visual and text input? First, you can do masked word prediction, just like you did with text-based BERT. But if you can do that for words, why not also do it for objects? Finally, just like we did next sentence prediction for text representations, we can match images with their captions.

So the models (after you get past object recognition) look a lot like our text-only models. I think you'd agree that visual question answering can be quite a bit harder ... so why are the accuracies so much higher than for text-based QA?

That's because all of the questions in VQA are multiple choice! It basically boils down to selecting which of four answers is correct. And it's not just VQA, other visual datasets are all multiple choice.

And while other datasets are implicitly multiple choice—if you ask SQuAD when something happened, it's essentially multiple choice among all the things that look like dates—it's not as clear-cut as here. And datasets like Quizbowl are a multiple choice task over tens of thousands of entities.

So why is QA over images always multiple choice? Because we have a hard time agreeing on what the answer should be! We saw this problem for long-form QA and even for factoid QA you need to be sure to have all of the relevant names for an item. Because we don't have text to ground the answer (again, which makes text-based QA more like multiple choice), even for everyday items you need to figure out what name to give something.

So, for example, take the question what is Mr. Burns wearing in the picture? What should you call these? Mules? Houseshoes? I personally think that some of these are better than others, but arguably all of these should be accepted. And Mr. Burns seems to prefer "Slippers". And this isn't just a problem in English: here's a map for what slippers are called in the German-speaking world. Naming things from a picture is hard, describing what a thing is doing is even harder, and harder still is describing *interactions*. What is Mr. Burns doing to his slippers? Dancing? Bouncing? Flaunting? Tapping? Given all of that flexibility, evaluating whether an answer is correct is just generally hard, as we've seen many times before.

And images aren't static ... often we care about answering questions about moving pictures. I.e., movies or TV. There have also been datasets about these questions as well. Just like VQA, these are multiple choice ... But does that mean you need to watch the movie to answer the questions? Not necessarily ... movies also have closed captions/subtitles,audio descriptions, and scripts. Just in case you're not familiar with these, let's talk about them and discuss what they can and cannot answer. Captions are the text that appear at the bottom of the screen and are available for people who are hard of hearing or who just want to be able to also read what the person is saying. They're not always purfect, but they usually get the point acroos. It's a lot of work to do it manually! [But a good place to hide jokes!]

While captions were first intended to make movies more accessible for the deaf, audio descriptions are created for the blind . . . for example here's a clip from Frozen. They explain what's going on in the movie beyond the dialog. While humans typically get this through audio, they are written down at some point and so these can be provided to the computer as text as well.

Finally, most movies and TV shows come from a text called a script. These contain not just the dialog, which get turned into caption but additional information like where a scene takes place and the instructions to the actors about what actions they need to take. While not as verbose as the descriptions, it nonetheless provides a way of interpreting the raw audiovisual representation of a movie.

Here are two datasets—MovieQA and TVQA—that have questions (multiple choice, of course) based on movies. And just like how you can insert both visual and text information into BERT, you can feed in recognized objects, the captions (called subtitles in this figure) into BERT.

So all of this looks a lot like how you do machine reading questions given a context . . . but the other kind of question answering we saw was "open domain" question answering using systems like DPR?

Yes! And just like for DPR where we had an encoder for the context and an encoder for the evidence, we can do the same thing for images and text: try to learn encoders such that the encoding of the image description and the encoding of the image are as close together as possible. As I record this in 2022, there are two big models that are doing this: CLIP from OpenAI and ALIGN from Google. They both have fairly similar architectures.

Then you can do things like find good captions for an image automatically or given a search string, find appropriate images for it.

As I wrap up, I hope that you've noticed the pattern here: the tools that we created to help people with visual or hearing impairments end up not just being a good idea for everyone, but they're also one of the key tools that we're using to build up computers' ability to understand the interaction between text and images. So the moral of the story is that if you want to support AI, support accessibility! And maybe we'll have AIs that can sort everybody's socks and answer questions about it afterward. The future can't come fast enough!

This is just a single lecture from a course.

YouTube likes to show you these videos out of order, but if you go to the course web page linked below, you can see the lectures in the right order and you can get resources like homeworks or suggested reading.

You can also visit QANTA dot org if you want to learn about our systems for creating computers that can answer questions.

Where QANTA stands for "question answering is not a trivial activity".

If you want to help the channel provide a big gradient to the algorithm by liking and subscribing.

How Punters Cheating with AI has Made Pub Quizzes' Audio Rounds Better

If you went to a pub quiz at the turn of the century, one staple that you'd *always* see was music identification: the pub master plays a song and you're supposed to identify it. But that's far less common these days. A big reason is that—as far as AI is concerned—it's mostly a solved problem . . . apps like Shazam and YouTube content ID can do this with extremely high accuracy even in a noisy bar. And thus it has become really easy to cheat in a pub quiz. It's also why, even though this video is about answer questions about music, I'm not going to put any popular music in this video.

Bleeding Gums Murphy was never popular.

So how do trivia competitions ask questions about music now? In the same way that YouTube videos speed up or distort music to get around content ID, pub quizzes have increasingly adopted using transformations of songs to ask questions. Identify the song given only a single instrument from the track or identify the song from the bluegrass cover.

In the same way that these questions reward deeper knowledge, there has also been a movement to reward real music knowledge:

This symphony's second movement . . . features a rising horn call of a C major arpeggio followed an A-flat sixteenth pickup to a G.

[orchestral music with a French horn solo]

And this is just asked using text . . . you need to know that an arpeggio is a chord played in sequence, you need to know what a French horn sounds like, and you need to know what a C vs. an A-flat sounds like. Mapping this to an underlying musical score is hard . . . there's no beautiful music track that I added in with my L337 editing skills. But of course, as is de rigueur in the Manchester paradigm, afterward the pyramidal sequence of clues ends with the much easier clue: For 10 points, identify this symphony sometimes called the "Song of the Night", which numerically precedes its composer's "Symphony of a Thousand." Remember that the Manchester paradigm that we talked about before structures questions to go from difficult clues to easy clues, which rewards knowledge and better discriminates between teams.

But I want to emphasize the virtuous cycle here: Pub quiz hosts were writing lazy questions that essentially amounted to playing an iTunes playlist in a crowded bar Deep convolutional neural networks allowed lazy pub quiz participants to to cheat on those questions Out of outrange and to restore equity, pub quiz hosts started writing better questions

The next step—which hasn't happened yet—is for AI systems to start using that training data to build systems that have to think a little bit harder to identify the songs.

So that's what has happened for audio questions . . . let's now turn to questions about images in the Manchester paradigm. Similarly, pyramidal questions about visual art describe less well-known details:

A sword stuck in its holder and the head of a corpse are seen near bushes on the lower left.

A rabbit runs to take cover from the central object in this work, which approaches the viewer.

If you know the painting well, then you'll be able to answer the question. If not, since it's a pyramidal question, in the rest of the question you'll get the rest of the clues.

FTP, name this painting depicting the seacliffs of Crete and the pair of legs belonging to a drowning youth, by Pieter Brueghel the Elder. And you can't just run your favorite computer vision system to figure out the title here . . . ships in the harbor? Ploughman on the hill? No, the title of this picture is "Landscape with the Fall of Icarus". And there's nothing in this picture that actually looks like Icarus here . . . ah, there he is. His legs anyway.

Or going back to our alleged rabbit . . .

FTP, name this 1844 landscape depicting a train crossing a bridge, a depiction of "the Great Western Railway" by J.M.W. Turner.

If you know the answer to this question, put it as a comment down below.

And this isn't just about visual salience. The most well-known aspect of Bronzino's Venus, Cupid, Folly, and Time isn't obvious but is better known than the rest of the painting because of Terry Gilliam's use of it in Monty Python.

So this is how textual clues are rendered. And in the Manchester paradigm, you present the clues in pyramidal order, going from difficult to easy. But these are textual clues. And yeah, perhaps you have to look at a painting or listen to a piece of music to answer it (and I think it would be fun to use visual or auditory information to better answer these questions . . . project idea). But the question is still text . . . can you ask pyramidal questions when the question itself is multimedia? The answer, I think, is yes, but before I go into that, I first need to explain why for the skeptics in the back.

The Manchester paradigm for question answering tries to make questions that are difficult to answer so that it rewards knowledge, understanding, and robustness. Just like Shazaam isn't impressive if it only works with clear, undistorted input ... that it works in such an adversarial environment like a crowded bar is pretty awesome! If we want robust question answering and true artificial intelligence, computers need to be able to "make do" with less and employ the same reasoning and recall ability that humans are able to do.

So let's see how the Manchester paradigm does that! One way is to have the normal presentation of material but to severely limit the quantity. For example, on the game show "Name that Tune", contestants bid each other down to see who can name the tune with the fewest notes.

Let's see that in action ...

Wash: I can name that tune in four notes Sandy: Wash, I can name that tune in three notes Wash: Sandy, name that tune Jim Lange: Alright, Sandy you get a chance to name that tune. Here's your clue once again. Introduced as a title tune of a 30s Broadway musical, this tune later appeared in the film Young Frankenstein and was revived in 1984. Listen carefully, here are your three notes ... [Three notes play] Jim Lange: Sandy? Sandy: Farmer in the Dell? [Sad Buzzer]

The answer of course, is ...

Gene Wilder as Frederick FrankenSTEEN: If you're blue and you don't know where to go to, why don't you go to where fashion sits ... Peter Boyle as the Monster: Ich han mis Portmone verloore!

did you get it?

Oh, that wasn't very clear, let's try that again ...

Taco: Putin' on the Ritz

that's right, Puttin' on the Ritz.

But this shows another way that this can be pyramidal ... go through performances from the lesser known, unintelligible warbling of Peter Boyle to the intelligible but niche performance of Taco to the canonical performance of Fred Astaire.

This is similar to how we structured the human vs. computer comparison at the Efficient QA competition. Give systems a shot at hard examples with little evidence, provide more and more evidence until you see where the first system gets it right. Creating pyramidal examples with n views is much easier than creating n examples of varying levels of difficulty.

For example, we could create a visual version of the Landscape with the Fall of Icarus question by only revealing part of the image ... going in reverse order of salience, just like the text of the question did: the hills in the background, the sword and skull, the farmer, and then finally the harbor with the two tiny legs.

Alternatively, you could pixelate the image, decreasing the pixelation until it can be identified. Again, if you know what this painting is, leave a comment down below.

But wait, Jordan, I hear you say. This just corresponds to different layers or different pools in a convolutional net. This won't make it that much harder for a computer to understand an image.

That's true! And until there is a Shazzam equivalent that's as good and as popular for images, we're probably going to have to wait for the kind of innovation we've seen for audio questions. But some of this is happening. For example, I've seen things like "identify the movie or TV show from the Muppet-ized version". So you can't just identify raw pixel patterns, they need to understand what part of the image is a Muppet and which is a part of the underlying media property.

And if we want a computer to be able to do that, we need to repeat this cycle, probably multiple times. But not just for images, but for all of the kinds of questions that we want a computer to answer.

But one part of this process is AI helping humans cheat! Are you condoning cheating? No, of course not. Because what this is really doing is helping the smart, clever people who create these questions be more creative. But we as AI researchers can make it easier for that adversarial authoring process to happen. Easier both for the authors, who end up with a better output with less effort and easier for the AI systems that need to train on data to get the most challenging data that requires actually *understanding* the underlying information.

And that is what makes this sort of research so fun. Just as a pub quiz should reward knowledge and add excitement of competition, we can use that same creative energy to make our algorithms less brittle and more robust.

What makes it hard to answer spoken questions? When we talked about Watson (link in the description), one of my minor criticisms was that it didn't listen to questions. Now we're going to talk about why that's a scientific issue and what makes this problem difficult.

Now, if you've just been watching the error rate for automatic speech recognition (ASR for short), the numbers are really impressive. And Microsoft has claimed that these results are superhuman! But these headline numbers are typically given as a word error rate, or more precisely a token error rate. And it's true that the word error rate is average is lower than human numbers, that's hiding something. The most common word in English is the ... so just getting that right doesn't mean that much.

Indeed, the most frequent words in English appear *a lot*. This is called a Zipfian distribution, where the probability of a word is inversely proportional to its frequency rank. So getting these highly frequent words right is easy and counts for a lot of the word error rate.

But are these the words that matter for question answering? No!

Named entities are the things that matter, and they're much less common. Let's take a look at question answering data. Every dot here is an error, and when the original word is very common . . . there aren't many speech recognition errors.

Let's look at where the error actually are: when the original word is "Gupta" (e.g., the empire of Chandragupta), "Clarendon" (as in Henry II's council that restricted ecclesiastical power in England) or "Kermit" (proud UMD alum), these either are unknown to the ASR system, as in the case with our green friend here, they get recognized as more common words. "Clarendon" becomes the allergy medication "claritin" and "gupta" becomes "group".

Plus, they often come from foreign languages and are easily confused or mispronounced. This is also a problem of the Cranfield vs. Manchester paradigm: while Alex Trebek is unlikely to mispronounce anything (he has watchdogs to rerecord anything that sounds off), a clueless user might indeed mispronounce "Where was Mao Zedong born?"

So what can you do to better answer spoken questions? First, you can ignore words from the recognized text that aren't confident: if there's enough redundancy in the question, you might still be able to get it right. Or you can look at the different options for recognized words; in other words, look at the "lattice" of the different options. If the top guess isn't right, maybe you'll find it under the old one.

Finally, you can also look for words that "sound" similar to words that you're looking for. So perhaps the original question had "Dumas" and the system detected "Dummy", you might be able to pull up the correct author because it shares the syllable "Dum". But this requires different algorithms: you need to be able to search by syllable rather than by just words.

And you need datasets to do this. One cheap way is to use text to speech (i.e., have a computer do the speaking) and then try to recognize this. This is strictly easier than the real world: there's no background noise, the pronunciations are always consistent, and there's only one speaker.

We did this first for the Manchester paradigm, and then a group from CMU did it for the Cranfield paradigm. As far as I know, there's no dataset of *real* speech even though there are tons of podcasts and youtube videos of people reading and answering questions. And because the question are all written down, you could line then up. But to the best of my knowledge, nobody has done that: project idea!

And if you have diverse speakers, you could contend with the disparities in ASR: much of the training data come from white men, so when tested on other groups, accuracy drops.

And then you could also get to some of the other challenges in parsing questions: for instance, this dataset from Google tackles the disfluencies or corrections that often come up in Cranfield paradigm questions.

Anvil yesterday sets and quest ions, reignite improv broth arable Tuesday quest ions and to wreck a nice beach. And maybe get better captions on YouTube, as long as people are good about uploading good data.

7
The Manchester Paradigm: The Art of Asking the Perfect Question

While computer scientists were getting their feet wet answering questions, during the same period trivia enthusiasts were perfecting how to ask the perfect question. This chapter outlines the conventions and processes of the highest form of question answering—in the biased but informed opinion of the author—variously called University Challenge, College Bowl, or Quiz Bowl and argues for why many of these processing could be a part of creating data for computer question answering.

The QA community is obsessed with evaluation. Schools, companies, and newspapers hail new SOTAs and topping leaderboards, giving rise to troubling claims (Lipton and Steinhardt, 2019) that an "AI model tops humans" (Najberg, 2018) because it 'won' some leaderboard, putting "millions of jobs at risk" (Cuthbertson, 2018). But what is a leaderboard? A leaderboard is a statistic about QA accuracy that induces a ranking over participants.

Newsflash: this is the same as a trivia tournament. The trivia community has been doing this for decades (Jennings, 2006); Section ?reference??? details this overlap between the qualities of a first-class QA dataset (and its requisite leaderboard). The experts running these tournaments are imperfect, but they've learned from their past mistakes (see Appendix ?reference??? for a brief historical perspective) and created a community that reliably identifies those best at question answering. Beyond the format of the *competition*, trivia norms ensure individual questions are clear, unambiguous, and reward knowledge (Section 7).

We are not saying that academic QA should surrender to trivia questions or the community—far from it! The trivia community does not understand the real world information seeking needs of users or what questions challenge computers. However, they have well-tested protocols to declare that someone is better at answering questions than another. This collection of tradecraft and principles can nonetheless help the QA community.

Beyond these general concepts that QA can learn from, Section 7 reviews how the "gold standard" of trivia formats, QB can improve traditional QA. We then briefly discuss how research that uses fun, fair, and good trivia questions can benefit from the expertise, pedantry, and passion of the trivia community (Section ?reference???).

Last year, I wrote a position paper about the AI researchers who make question answering systems can learn from trivia nerds. Given the time constraints of the conference and the need to be more "academic", I couldn't say everything I wanted to. So I'll try not to be too repetitive in this video. What I would like to do is talk more about how the trivia community learned hard lessons and how the question answering community is still learning those lessons. I'm hoping by going into a little more detail about how the trivia community made mistakes . . . might prod computer scientists to see how they might be relevant to how we're training and evaluating computer question answering systems.

Even if you don't care about question answering specifically, I think this will help get ideas across about how to do effective evaluation in machine learning. Okay, so what are hard won lessons of the trivia community that the geniuses working on AI have yet to master? 1. Don't cheat. 2. Reward knowledge 3. Write good questions. While that may seem simple, it took a while for humans to figure these things out for trivia competitions, and the artificial intelligence community is still figuring it out.

GI Bill Footage from CBS This Morning: https://www.youtube.com/watch?v=fgtvMceoimU

As veterans returned home from World War II, there was a huge thirst for new forms of entertainment. The Manhattan project and the space race showed the importance of knowledge and expertise, and the GI Bill turned open the taps of knowledge to an information-thirsty populace. These trends were naturally reflected in popular entertainment, as new quiz shows like Twenty One sprang up.

https://youtu.be/JRKxNpwqBac?t=1154

What's the matter with 21? It was rigged, as Herb sandbagged to let his opponent win as documented in the 1994 movie Quiz Show!

https://youtu.be/bj-m3DdmnoE?t=83

The handsome Charles van Doren was able to peek at the answers and did pretty well!

Now, the obvious version of cheating is something that we don't often see in AI research. But there are subtler ways that systems after a manner cheat. Let's think about this very abstractly. Let's say that when you are asked some questions, you get to absorb some information that you can use next time.

Now, in the 21 cheating scandal, the process allegedly went something like this: Let's try practicing these questions! Sure ... Ooh, you got two wrong, here are the correct answers You could then compute the total length of the correct answers and that's how much information "leaked" from the test set. That's a lot!

Now, let's compare that with a more "reasonable" system that resembles what you actually see for a lot of question answering datasets. You can submit a system that answers questions (usually as a docker container to make sure it doesn't phone home), and then you get back a score with how well you did. There's no problem here, right? You don't know what the answer is, right?

Well, don't forget that computers have infinite patience ... think about it like cracking a password. You could try all possible answers for position 1, all possible answers for position 2, etc. But computers can crack more cleverly ... you have a set of answers and have some confidence on each guess. Take your lowest confidence answer, switch the answer. If the score goes up, the guess went from wrong to right. If the score goes down, the guess went from right to wrong. If the score stays the same, the guess stayed wrong ... keep hunting. But in this example, the score went up, so we know that that answer was actually correct. Nobody explicitly told us the answer was Uluru, but the effect is the same. If you do it this way, there are thousands of possibilities to try, not millions. And this is clearly in the realm of possibility for computers: even if you can't probe every answer, you can still crank up your score quite a bit (and don't forget that outcomes are correlated!).

Now, I don't think people would actually do this (or at least admit to it), but there are many black box neural network optimization toolkits that adjust the many parameters of a neural model (often on dev data). And implicitly, even on test data, you get higher numbers for some models than others ... and that's a signal. People try out many different models and just use the highest score as the legitimate representative of a system or a technique.

Jesse Dodge has done some great work on how to more fairly report experimental results without cheating: essentially, you need to have a budget for how many comparisons you make. And then you need to apply the same budget to every system fairly.

And this is why I think the human model for comparing question answering ability (i.e., a trivia tournament) is the model we should follow: you only get to run on the test data.

And, yeah, maybe the reason is that we can't wipe human memories. But I think it's more realistic, even for computers: you only have a single chance to make a first impression on a user. And this is why the model is also what happens for DARPA evaluations.

Now, bad stuff happens in this style of evaluation. We had a particularly embarrassing outing in 2016 when we lost . . . very badly against some smart humans. Despite having done much better the year before.

Us losing (badly) to top quiz bowlers: https://www.youtube.com/watch?v=c2kGD1EdfFw

Ugh, that still hurts. But that's not an argument against these style of evaluations. You should lower your embarrassment threshold and do evaluations more frequently.

And this is why I use surprise questions for my courses to see how well systems react to novel data.

But it's not just the sanctity of the test set. Let's move on to how you ask questions.

Regular viewers of this channel will know that I am an advocate of the Quizbowl model of asking questions. But let's dig into its evolution a little bit more.

The mythology of Quizbowl says that it was a USO diversion created by Canadian Don Reid as a way of entertaining allied troops in Europe. However, in my reading it seems that a lot of the core ideas were introduced by host Allen Ludden once it became a radio program.

But the key idea is that the questions can be interrupted, meaning that part of the skill is correctly balancing your own uncertainty. Every word is an opportunity for either team to show that it knows more than the other team. I have videos arguing for why this is the right and proper way to evaluate who knows more about a topic, so I will keep this brief.

In that 2020 paper I mentioned before, Ben Boerschinger and I argued that you want question answering datasets to be maximally discriminative.

See the paper for the equations! But the moral of the story is that the Quizbowl format popularized by College Bowl created the right format for reliably determining who knows the most about a topic. Every question is discriminative. In other words, rewarding knowledge rather than buzzer speed or luck (you can see my other videos for why, for instance Jeopardy isn't a good test of human vs. computer intelligence).

But the format is a necessary but not sufficient condition: you also need to have good quality questions.

Let's now turn to the final lesson from the trivia community: writing good questions. During the early nineties, three heroes created the Academic Competition Federation. As a point of Terrapin pride, John and Ramesh were Maryland students. Together, with Carol Guthrie, they created an organization dedicated to creating questions that fully take advantage of the beauty and elegance of the Quizbowl format.

So what innovations did they introduce? First, avoiding ambiguity:

So what does a good question look like? First, you don't want to have ambiguous questions. For example, if you ask the question "What's the capital of Georgia", you answer beautiful Tblisi but the official answer is Atlanta, then you're going to be upset!

Sewon Min recently had a great paper highlighting this problem in modern question answering datasets in 2020, but this was a big part of the diatribes Carol was making in the early nineties.

Part of ACF's style was to make sure that what they're asking is obvious as a reaction to some of the excesses of ambiguous or misleading college bowl questions. There are many dimensions of this. For example, making sure that you don't just ask "when was the Battle of Hastings" you say things like "in what year" or "day and month required". This is a problem in datasets like SQuAD and Natural Questions where it's not always clear what's required to answer the question: annotators just highlight whatever is convenient.

So why is this a problem? Machine learning algorithms have an objective function that they're trying to optimize. If they give a correct answer, they should be rewarded so that they can repeat the process on the next question. If you get the question, "What is the capital of Georgia" and you're ruled incorrect when you say "Atlanta" (after learning the opposite lesson from the previous training example), you are going to be upset. And this is true whether you're a human or a machine learning algorithm.

Similarly, you need to accept all of the possible answers to a question. Again, we see the same pattern as before: research papers are only now recapitulating the lessons learned by the trivia community 40 years ago. A great undergrad I'm working with showed that this can make machine learning question answering systems more robust. If you only accept "Timothy Donald Cook" as the correct answer for "Who is the CEO of Apple", then that makes "Tim Cook" as wrong as "Tim Apple". Again, this confuses algorithms that are trying to learn how to answer these questions. And even if you just care about climbing the leaderboards, we recently had a paper explaining how to squeeze out a few more points by paying attention to that.

https://slate.com/culture/2019/04/jeopardy-quiz-bowl-connection-ken-jennings.html

Another organization that should be mentioned in the promulgation of good Quizbowl is National Academic Quiz Tournaments, founded by R. Hentzel. Not only have they created a system of good, discriminative high-quality tournaments for middle school and high school students in North America, they have also served as the employer of people like Ken Jennings and Larissa Kelly (and full disclosure, me too in the early aughts) and have made Quizbowl more professional. Here's Ken Jenning's extolling the virtues of Quizbowl in a NAQT promotional video.

There are many ways that this is relevant: things have different names because of language, history, pen names, nicknames, convenience, mathematical equivalence, or domain-specific rules. The new professional organizations have created systematic evaluations that, because the goal is to reward knowledge, also are the kinds of rules for judging the correctness of machine answers. Even if a user might prefer to get their answers in Fahrenheit, the computer should know that they're the same thing!

But these organizations are not just question writing organizations. Their goal is to create efficient systems for as quickly as possible determining out of 256 teams at a convention center which knows the most. In addition to the Quizbowl format, the selection of appropriately difficult questions asked at the right time to the right teams makes this possible.

Here, I don't even have a "perfect" recent paper to flash on the screen to show that, yeah, computer science knows that it's a problem 40 years to late. We've been working on using item response theory and adversarial interfaces, but there's still quite a ways to go.

Let's take how computer comparisons lag behind those for humans. While there are monthly competitions to see which human can best answer questions, computers are reusing stale data from 2018 to see who's best. And those questions are written by random people on the internet: either crowdworkers or clueless people looking for answers in a search engine. In contrast, domain experts are writing questions lovingly edited by people like Ken Jennings and Larissa Kelly when people go to NAQT or ACF competitions.

And while things like Dynabench are trying to create adversarial datasets (or our own TrickMe interface), these datasets are tiny compared to the tens of thousands of questions written every year for human trivia competitions. I, of course, think that this is a possibility for synergies between the two communities.

Computers can help find patterns and reduce drudgery: make human trivia more exciting and efficient. And the trivia experts, with their depth of knowledge, can help computers understand more of the world. Likewise, the best practices of the hard won victories of the last seventy years of trivia can help machine learning researchers avoid the same mistakes.

And, most importantly, I think it can be fun for everybody, especially us researchers.

The Craft of Question Writing

Trivia enthusiasts agree that questions need to be well written (despite other disagreements). Asking "good questions" requires sophisticated pragmatic reasoning (Hawkins et al., 2015), and pedagogy explicitly acknowledges the complexity of writing effective questions for assessing student performance (Haladyna, 2004, focusing on multiple choice questions).

QA datasets, however, are often collected from the wild or written by untrained crowdworkers. Crowdworkers lack experience in crafting questions and may introduce idiosyncrasies that shortcut machine learning (Geva et al., 2019). Similarly, data collected from the wild such as Natural Questions (Kwiatkowski et al., 2019) or AmazonQA (Gupta et al., 2019) by design have vast variations in quality. In the previous section, we focused on how datasets as a whole should be structured. Now, we focus on how specific *questions* should be structured to make the dataset as valuable as possible.

Avoiding ambiguity and assumptions

Ambiguity in questions not only frustrates answerers who resolve the ambiguity 'incorrectly'. Ambiguity also frustrates the goal of using questions to assess knowledge. Thus, the US Department of Transportation explicitly bans ambiguous questions from exams for flight instructors (Flight Standards Service, 2008); and the trivia community has likewise developed rules and norms that prevent ambiguity. While this is true in many contexts, examples are rife in format called QB (Boyd-Graber et al., 2012), whose very long questions[1] showcase trivia writers' tactics. For example, QB author Zhu Ying (writing for the 2005 PARFAIT tournament) asks participants to identify a fictional character while warning against possible confusion [emphasis added]:

> He's **not Sherlock Holmes**, but his address is 221B. He's **not the Janitor on Scrubs**, but his father is played by R. Lee Ermy. [. . .] For ten points, name this misanthropic, crippled, Vicodin-dependent central character of a FOX medical drama.
> **ANSWER:** Gregory House, MD

In contrast, QA datasets often contain ambiguous and under-specified questions. While this sometimes reflects real world complexities such as actual under-specified or ill-formed search queries (Faruqui and Das, 2018; Kwiatkowski et al., 2019), ignoring this ambiguity is problematic. As a concrete example, Natural Questions (Kwiatkowski et al., 2019) answers "what year did the us hockey team win the Olympics" with 1960 and 1980, ignoring the US women's team, which won in 1998 and 2018, and further assuming the query is about *ice* rather than *field* hockey (also an Olympic event). Natural Questions associates a page about the United States men's national ice hockey team, arbitrarily removing the ambiguity *post hoc*. However, this does not resolve the ambiguity, which persists in the original question: information retrieval arbitrarily provides one of many interpretations. True to their name,

[1] Like *Jeopardy!*, they are not syntactically questions but still are designed to elicit knowledge-based responses; for consistency, we still call them questions.

Natural Questions are often under-specified when users ask a question online.

The problem is neither that such questions exist nor that machine reading QA considers questions given an associated context. The problem is that tasks do not explicitly acknowledge the original ambiguity and gloss over the implicit assumptions in the data. This introduces potential noise and bias (i.e., giving a bonus to systems that make the same assumptions as the dataset) in leaderboard rankings. At best, these will become part of the measurement error of datasets (no dataset is perfect). At worst, they will recapitulate the biases that went into the creation of the datasets. Then, the community will implicitly equate the biases with correctness: you get high scores if you adopt this set of assumptions. These enter into real-world systems, further perpetuating the bias. Playtesting can reveal these issues (Section ?reference???), as implicit assumptions can rob a player of correctly answered questions. If you wanted to answer 2014 to "when did Michigan last win the championship"—when the Michigan State Spartans won the Women's Cross Country championship—and you cannot because you chose the wrong school, the wrong sport, and the wrong gender, you would complain as a player; researchers instead discover latent assumptions that creep into the data.[2]

It is worth emphasizing that this is not a purely hypothetical problem. For example, Open Domain Retrieval Question Answering (Lee et al., 2019) deliberately avoids providing a reference context for the question in its framing but, in re-purposing data such as Natural Questions, opaquely relies on it for the gold answers.

Avoiding superficial evaluations

A related issue is that, in the words of Voorhees and Tice (2000), "there is no such thing as a question with an obvious answer". As a consequence, trivia question authors delineate acceptable and unacceptable answers.

For example, in writing for the trivia tournament Harvard Fall XI, Robert Chu uses a mental model of an answerer to explicitly delineate the range of acceptable correct answers:

> In Newtonian gravity, this quantity satisfies Poisson's equation. [. . .] For a dipole, this quantity is given by negative the dipole moment dotted with the electric field. [. . .] For 10 points, name this form of energy contrasted with kinetic.
> **ANSWER:** potential energy *(prompt on energy; accept specific types like electrical potential energy or gravitational potential energy; do not accept or prompt on just "potential")*

Likewise, the style guides for writing questions stipulate that you must give the answer type clearly and early on. These mentions specify

[2] Where to draw the line is a matter of judgment; computers—which lack common sense—might find questions ambiguous where humans would not.

whether you want a book, a collection, a movement, etc. It also signals the level of specificity requested. For example, a question about a date must state "day and month required" (September 11, "month and year required" (April 1968), or "day, month, and year required" (September 1, 1939). This is true for other answers as well: city and team, party and country, or more generally "two answers required". Despite these conventions, no pre-defined set of answers is perfect, and every worthwhile trivia competition has a process for adjudicating answers.

In high school and college national competitions and game shows, if low-level staff cannot resolve the issue by throwing out a single question or accepting minor variations (America instead of USA), the low-level staff contacts the tournament director. The tournament director, who has a deeper knowledge of rules and questions, often decide the issue. If not, the protest goes through an adjudication process designed to minimize bias:[3] write the summary of the dispute, get all parties to agree to the summary, and then hand the decision off to mutually agreed experts from the tournament's phone tree. The substance of the disagreement is communicated (without identities), and the experts apply the rules and decide.

Consider what happened when a particularly inept *Jeopardy!* contestant[4] did not answer laproscope to "Your surgeon could choose to take a look inside you with this type of fiber-optic instrument". Since the van Doren scandal (Freedman, 1997), every television trivia contestant has an advocate assigned from an auditing company. In this case, the advocate initiated a process that went to a panel of judges who then ruled that endoscope (a more general term) was also correct.

The need for a similar process seems to have been well-recognized in the earliest days of QA system bake-offs such as TREC-QA, and Voorhees (2008) notes that

> [d]ifferent QA runs very seldom return exactly the same [answer], and it is quite difficult to determine automatically whether the difference [...] is significant.

In stark contrast to this, QA datasets typically only provide a single string or, if one is lucky, several strings. A correct answer means *exactly* matching these strings or at least having a high token overlap F_1, and failure to agree with the pre-recorded admissible answers will put you at an uncontestable disadvantage on the leaderboard (Section ?reference???).

To illustrate how current evaluations fall short of meaningful discrimination, we qualitatively analyze two near-SOTA systems on SQuAD V1.1: the original XLNet (Yang et al., 2019) and a subsequent iteration called XLNet-123.[5]

[3] https://www.naqt.com/rules/#protest

[4] http://www.j-archive.com/showgame.php?game_id=6112

[5] We could not find a paper describing XLNet-123, the submission is by http://tia.today.

Despite xlNet-123's margin of almost four absolute F_1 (94 vs 98) on development data, a manual inspection of a sample of 100 of xlNet-123's wins indicate that around two-thirds are 'spurious': 56% are likely to be considered not only equally good but essentially identical; 7% are cases where the answer set omits a correct alternative; and 5% of cases are 'bad' questions.[6]

Our goal is not to dwell on the exact proportions, to minimize the achievements of these strong systems, or to minimize the usefulness of quantitative evaluations. We merely want to raise the limitation of *blind automation* for distinguishing between systems on a leaderboard.

Taking our cue from the trivia community, we present an alternative for MRQA. Blind test sets are created for a specific time; all systems are submitted simultaneously. Then, all questions and answers are revealed. System authors can protest correctness rulings on questions, directly addressing the issues above. After agreement is reached, quantitative metrics are computed for comparison purposes—despite their inherent limitations they at least can be trusted. Adopting this for MRQA would require creating a new, smaller test set every year. However, this would gradually refine the annotations and process.

This suggestion is not novel: Voorhees and Tice (2000) accept automatic evaluations "for experiments internal to an organization where the benefits of a reusable test collection are most significant (*and the limitations are likely to be understood*)" (our emphasis) but that "satisfactory techniques for [automatically] evaluating new runs" have not been found yet. We are not aware of any change on this front—if anything, we seem to have become more insensitive as a community to just how limited our current evaluations are.

Focus on the bubble

While every question should be perfect, time and resources are limited. Thus, authors and editors of tournaments "focus on the bubble", where the "bubble" are the questions most likely to discriminate between top teams at the tournament. These questions are thoroughly playtested, vetted, and edited. Only after these questions have been perfected will the other questions undergo the same level of polish.

For computers, the same logic applies. Authors should ensure that these discriminative questions are correct, free of ambiguity, and unimpeachable. However, as far as we can tell, the authors of QA datasets do not give any special attention to these questions.

Unlike a human trivia tournament, however—with finite patience of the participants—this does not mean that you should necessarily remove all of the easy or hard questions from your dataset. This could inadvertently lead to systems unable to answer simple questions like

[6] Examples in Appendix ?reference???.

"who is buried in Grant's tomb?" (Dwan, 2000, Chapter 7). Instead, focus more resources on the bubble.

Why QB *is the Gold Standard*

We now focus our thus far wide-ranging QA discussion to a specific format: QB, which has many of the desirable properties outlined above. We have no delusion that mainstream QA will universally adopt this format (indeed, a monoculture would be bad). However, given the community's emphasis on fair evaluation, computer QA can borrow *aspects* from the gold standard of human QA.

We have shown example of QB questions, but we have not explained how the format works; see Rodriguez et al. (2019) for more. You might be scared off by how long the questions are. However, in real QB trivia tournaments, they are not finished because the questions are *interruptible*.

Interruptible A moderator reads a question. Once someone knows the answer, they use a signaling device to "*buzz in*". If the player who buzzed is right, they get points. Otherwise, they lose points and the question continues for the other team.

Not all trivia games with buzzers have this property, however. For example, take *Jeopardy!*, the subject of Watson's *tour de force* (Ferrucci et al., 2010). While *Jeopardy!* also uses signaling devices, these only work *once the question has been read in its entirety*; Ken Jennings, one of the top *Jeopardy!* players (and also a QBer) explains it on a *Planet Money* interview (Malone, 2019):

> **Jennings:** The buzzer is not live until Alex finishes reading the question. And if you buzz in before your buzzer goes live, *you actually lock yourself out for a fraction of a second*. So the big mistake on the show is people who are all adrenalized and are buzzing too quickly, too eagerly.
> **Malone:** OK. To some degree, *Jeopardy!* is kind of a video game, and a *crappy video game where it's, like, light goes on, press button*—that's it.
> **Jennings:** (Laughter) Yeah.

Jeopardy!'s buzzers are a gimmick to ensure good television; however, QB buzzers discriminate knowledge (Section ?reference? ??). Similarly, while TriviaQA (Joshi et al., 2017) is written by knowledgeable writers, the questions are not pyramidal.

Pyramidal Recall that effective datasets discriminate the best from the rest—the higher the proportion of effective questions ρ, the better. QB's ρ is nearly 1.0 because discrimination happens *within* a question: after every word, an answerer must decide if they know enough to answer. QB questions are arranged so that questions are maximally *pyramidal*: questions begin with hard clues—ones that require deep understanding—to more accessible clues that are well known.

Well-Edited QB questions are created in phases. First, the *author* selects the answer and assembles (pyramidal) clues. A *subject editor* then removes ambiguity, adjusts acceptable answers, and tweaks clues to optimize discrimination. Finally, a *packetizer* ensures the overall set is diverse, has uniform difficulty, and is without repeats.

Unnatural Trivia questions are fake: the asker already knows the answer. But they're no more fake than a course's final exam, which—like leaderboards—are designed to test knowledge.

Experts know when questions are ambiguous (Section 7); while "what play has a character whose father is dead" could be *Hamlet*, *Antigone*, or *Proof*, a good writer's knowledge avoids the ambiguity. When authors omit these cues, the question is derided as a "hose" (Eltinge, 2013), which robs the tournament of fun (Section ?reference? ??).

One of the benefits of contrived formats is a focus on specific phenomena. Dua et al. (2019) exclude questions an existing MRQA system could answer to focus on challenging quantitative reasoning. One of the trivia experts consulted in Wallace et al. (2019a) crafted a question that tripped up neural QA by embedding the phrase "this author opens

Crime and Punishment" into a question; the top system confidently answers Fyodor Dostoyevski. However, that phrase was in a longer question "The narrator in *Cogwheels* by this author opens *Crime and Punishment* to find it has become *The Brothers Karamazov*". Again, this shows the inventiveness and linguistic dexterity of the trivia community.

A counterargument is that real-life questions—e.g., on Yahoo! Questions (Szpektor and Dror, 2013), Quora (Iyer et al., 2017) or web search (Kwiatkowski et al., 2019)—ignore the craft of question writing. Real humans react to unclear questions with confusion or divergent answers, explicitly answering with how they interpreted the original question ("I assume you meant. . . ").

Given real world applications will have to deal with the inherent noise and ambiguity of unclear questions, our systems must cope with it. However, addressing the real world cannot happen by glossing over its complexity.

Complicated QB is more complex than other datasets. Unlike other datasets where you just need to decide *what* to answer, in QB you also need to choose *when* to answer the question.[7] While this improves the dataset's discrimination, it can hurt popularity because you cannot copy/paste code from other QA tasks. The cumbersome pyramidal structure complicates[8] some questions (e.g., what is log base four of sixty-four).

[7] This complex methodology can be an advantage. The underlying mechanisms of systems that can play QB (e.g., reinforcement learning) share properties with other tasks, such as simultaneous translation (Grissom II et al., 2014; Ma et al., 2019), human incremental processing (Levy et al., 2008; Levy, 2011), and opponent modeling (He et al., 2016).

[8] But does not necessarily preclude, as the Illinois High School Scholastic Bowl Coaches Association shows:

> This is the smallest counting number which is the radius of a sphere whose volume is an integer multiple of π. It is also the number of distinct real solutions to the equation $x^7 - 19x^5 = 0$. This number also gives the ratio between the volumes of a cylinder and a cone with the same heights and radii. Give this number equal to the log base four of sixty-four.

8
Watson on Jeopardy!: Unquestioned Answers from IBM's tour de force

It's been nearly a decade since IBM Watson crushed two puny humans on Jeopardy! Some people took that to mean that computers were definitively better than humans at trivia. But that isn't the complete answer—this chapter, inspired by Jeopardy!'s gimmick of responding to answers to questions, questions some of the "answers" that emerged from IBM's tour de force.

I remember when I first heard rumblings of Watson. Because I had a foot in both the AI and trivia communities, I heard two different stories. I heard rumors of amazing work in parsing and semantic role labeling happening from researchers who ventured north to Nassau county in New York (I was doing my PhD at Princeton). From the trivia community, I heard of some people who were being paid by IBM to play trivia games but that they couldn't say anything more.

I was very sceptical. By the time that Watson came to fruition, I had moved to the University of Maryland. Then, my scepticism turned to jealousy. I watched, along with the rest of the world, one of the greatest achievements of AI unfold in front of me. What was I doing wasting my time working on topic models if this was also legitimate research?

Let me be clear that the technical triumphs are indisputable (and, in my opinion, under-appreciated). From the work on wagering to synthesizing multiple information sources, Watson (Ferrucci et al., 2010) was from top to bottom a top-notch well-oiled machine. And it computed all of this in real time—something that wasn't strictly necessary but still impressive.

After talking a little about how Watson works, I'll get into how Watsons victory left me feeling unconvinced about the primacy of computers' ability to answer questions.

How Watson Works

While "neural" question answering (which we'll discuss in the next chapter) is the big thing these days, Watson came of age in the *statistical* age. To understand some of how Watson works, its helpful to review some of the work that came before Watson that helped inspire it.

Preceeding the statistical age of QA was the *rule-based* age: systems that were impressive on individual questions about baseball or geology but that faltered as soon as it saw a question that was unexpected in terms of the domain, they'd fail miserably.

One of the guiding principles of the statistical age was "the unreasonable effectiveness of big data", and this required scaling approaches to web-scale data. The first things that people tried was scaling up the "good, old-fashioned" approaches that defined the rule-based systems. Probably the most prominent system in this category is Start from Boris Katz at MIT. This system first launched in 1993. This like many of the systems in this era, take an approach that's somewhat similar to to the old-fashioned AI approaches: parse the query with a and look it up in a knowledge base. And since this was in the age of the semantic web, this was in the form of RDF files.

But this is more a change in scale rather than in form from LUNAR and BASEBALL. . . but still an amazing achievement to deal with the messiness of Internet text. And its ability to *search* the Internet to find answers is clearly a huge influence on later techniques that we'll see like Watson today and Dense Passage Retrieval later (Chapter ?reference???).

A more transformative approach was the QASM approach from Radev at al. It used the concepts from machine learning to find answers given a query. In the information theory interpretation of machine translation, you imagine that all of the utterances in a target language is a corruption of English. Now this is pretty complicated for machine translation, but this is a little bit like what happened with how the the game show *Jeopardy!* came to be.

In Chapter ?reference???, we'll talk about the cheating scandal around the American game show *21*, where Charles van Doren got the answers in advance. Part of the mythology of Jeopardy! is that Merv Griffen said, why don't we just give them the answers! So you have clues like "During WWII this computer scientist & code breaker converted his money into silver & buried it; he never found his buried treasure". The correct response, which is phrased as a question, is "Who is Alan Turing?" In other words, the central conceit of Jeopardy! is that it converts questions into answers (and then the contestants need to give a response to that answer in the form of a question).

What QASM is doing is doing the reverse: it has a model that tries to turn questions into the search string that could find the answer to the question. There are a number of transformations that it applies: swapping words, deleting words, ignoring words (e.g., if you have Cleveland, Ohio and ignore Ohio, you'll find more things about US president Grover Cleveland), etc.

In the end, you have a model that creates specialized queries from the original question. Later on, when we talk about dense passage retrieval, those systems are doing something similar, except they're generating the query in a continuous vector space. Doing it with real words like QASM is more interpretable: a human could understand what's going on, while a vector query is inscrutable it seems to work better.

Okay, enough delay. Let's talk about *Jeopardy!* and how Watson answers *Jeopardy!* clues. First, Jeopardy! doesn't have one kind of clue. There are many different types. Let's start with rhyme time: from the name of the category, you know that both words in the response have to rhyme with each other. This is in essence a constrained search: "tramp stamp" you could find with a single search, but other times you need to do two searches and try to find the thing that rhymes.

In other cases, the constraint is that the correct answer starts with a particular letter.

The category that I feared the most was anagrams: where you need to rearrange the letters in the clue to get to the answer. This is actually easier for a computer than a human!

Each of these can have different approaches that can generate guesses that might be the correct answer. And you might think that this is all encoded in the category. Not so!

And for "normal" clues, this looks a lot like the reformulations that we saw in QASM: 400th anniversary of 1898 gets rewritten to 27th May 1498, India gets transformed into Kappad, and then you can find something with the correct response: Vasco de Gama.

So the first phase of the Watson pipeline is to—in parallel—generate all sorts of guesses based on different interpretations of the question. Some try to solve anagrams, others look for rhymes, others run an IR query like the Start system we talked about before, others are doing transformations like the QASM approach we talked about before.

And sometimes the approaches need to take a recursive approach, combining different subsystems to get to the right answer.

From this you have dozens of possible guesses. How do you know which—if any—to select? This all gets fed into a logistic regression problem. This can take into account how consistent the evidence matches the clue, how popular the response is, and it can even take into account things like Jeopardy!'s love of puns.

In my homework code, I call this component the buzzer. And this is the same function that the logistic regression function plays here. If the logistic regression is confident enough, Watson buzzes in to answer.

How Does Watson Know when to Buzz?

If you're watching this video, I'm assuming you know both about logistic regression and a little bit about IBM's Watson that played the US game show Jeopardy! If you want to know more, I have some suggestions for videos linked into the description.

The most interesting part of what Watson did to answer questions was to estimate its confidence: how likely is guess a to clue c going to be correct. It gathered a bunch of individual clues together to figure this out.

But how did Watson actually put this information together? It used a tool called logistic regression. The goal of this approach is to take all of the information about a possible guess and output a probability of how likely it thought the guess would be correct. The higer the score, the more it trusted the guess.

There are many little pieces that could go into thinking that a guess is good or not. These pieces of evidences are called features. While features aren't always active, when they are, they provide evidence of whether a guess is good. For example, you could have a feature (and Watson probably did) for when a category has a quote in it.

In a computer programming language, this could simply be an if statement: if ASCII character 34 is in this string, return 1. But just knowing that there's a quote in the category does't tell you which clues to favor. You would need to make the the feature more specific.

Let's go to a question that I got while I was on Jeopardy! And this shows that these aren't just a category-specific phenomenon: . However, this can also apply to individual clues as well. In a quite normal category about geography, this clue came up:

"G.I." hope you know that 0 degrees latitude & 180 degrees longitude is just east of this group, part of Kiribati

Notice that "G.I." is in quotation marks. This is a signal, that the correct response will start with G.I., in this case the correct response is "Gilbert Islands". Named of course for Johnny Gilbert, "the voice of Jeopardy!". Actually not, it's far more confusing than that: it was named by a German admiral—Adam Johann von Krusenstern—who led the first Russian circumnavigation of the globe. Krusenstern recognized that the British Captain Thomas Gilbert had described each of the islands individually and applied the the name to the group of islands.

So perhaps we need to have multiple features to capture whether our answer is consistent with this clue. To be clear, we don't know exactly what features Watson used, but it could look something like this. You might have one feature to indicate that there is a quotation mark in the clue. You could have another feature to show that the quote matches the candidate guess. Perhaps you have another feature that indicates whether the feature matches a multiword guess.

But these are just answering yes or no questions . . . how does the system know how to balance these different features to come to a final decision? At this point, you need to look at the cases where the system has generated a lot of guesses and seen which of them are correct or not. From this, you can learn that when a letter is in quotes and it matches, that's a good sign. These are called feature weights and for the binary features that we've talked about, they're either on or off.

But not all of the features need to be binary. Another very good feature might be how well the guess matches the clue (e.g., cosine similarity between representations). So a way to generalize both the binary features and continuous features is to take the dot product of a vector representing all of the features and all of the weights to compute a score for the guess. The higher the score, the better the guess and the more likely the system will answer.

This is not a one and done process. If you're building a system like this, you should look at things the system is still getting wrong and add a new feature to correct the issue. This is the same process that Watson used to build their statistical system. Maybe the system got confused when somebody has a quote from literature and you need to make sure that doesn't get counted or add a new feature to handle that case. It's very rare to get the features right the first time around.

The statistical approach has some advantages over the neural approach we'll see in the next chapter: it is easy to understand why a system is doing what it is. And it's easy to fix problems as they pop up.

Learning as You Go

Something that is particularly unique to Jeopardy! and not to the other QA settings we've looked at is that the category is a huge constraint on what the correct responses are. A large part of the Jeopardy! buzzer is figuring out the "lexical answer type". Part of this is is looking at clues in the question, things like "this country" or "this game" tell you what kind of entity the response would be. And this is sometimes at odds with the category, as shown here.

And part of what made Watson a good Jeopardy! Player is that it would learn as it explored the category. After getting a couple of months wrong, Watson can learn that all of the answers should be a month.

One thing that made Watson particularly impressive was that it not just computed raw accuracy but also compared against human performance.

This chart showed how Watson progressed in different iterations of the system, inching up to the cloud of Jeopardy! champions. Ken Jennings is in red there, still clearly dominating even the final version of Watson.

This Game is Rigged, I Tell Ya!

These successes have been well documented (not least by IBM itself, who rightly celebrated the great achievements); however, things were not perfect. . . the game was rigged. It's useful to go over the lifecycle of an entire question: how it was written, how it's communicated, how players answer, and how the game unfolds afterward. At every stage, there's a slight benefit to the computer, which taken together makes this an unfair competition.

This is a problem! First, it's a problem scientifically because we want to have fair comparisons of human vs. computer intelligence. More importantly, I want to have my turn having my question answering robots face off against trivia whizzes (Chapter ?reference???), and I can't do that if everybody thinks that Watson's spin on *Jeopardy!* settled the issue (and it hasn't).

But first, in case you don't know how *Jeopardy!* works, we'll review that. However, if you've calculated a Coryat score before, you can go ahead and skip ahead.

What Jeopardy! is and How it Works

Jeopardy! is a gameshow created by Merv Griffin that debuted in 1967 (Griffin, 2003). Its big gimmick is that the player responses are given in the form of a question in reaction to infamous cheating scandals. For example, the clue

> The CAPTCHA test against spam & robot programs is called the 'reverse' test named for this British code breaker

would have response Who is Alan Turing. The clues are arranged in a grid: columns represent categories and rows represent difficulty, with the more difficult questions being worth more.

There are three players who stand side by side behind podiums. When a clue is read, any of the three players can "buzz in" to say that they want to give a response. If they give the correct response, they then have "control of the board" and can select the next clue.

One advantage of controlling the board is that some questions are called "Daily Doubles" which allow the player to potentially double their score: a player can wager any part of their score, and if they get it right they get that ammount added to their score (but a wrong response will subtract it from their score).

The Pool of Questions

JBG: Citation needed

Part of the agreement between *Jeopardy!* and IBM was that the competition would take place on normal, written questions. In the media coverage of the competition, this focused on avoiding video and picture daily doubles (fairly reasonable, but we'll discuss this more in a bit). However, this causes two problems: the questions are too easy and do not necessarily challenge computers.

So what makes up "normal" questions? Every game of *Jeopardy!* has questions that range in difficulty. Because it's a television show, many questions are easy enough that the average viewer at home can get them. Moreover, the humans on the stage with Watson are not normal contestants. Ken Jennings is certifiably the greatest of all time (Low, 2020, GOAT) *Jeopardy!* player, and Brad Rutter isn't bad himself.

The average "normal" *Jeopardy!* contestant, including not so great players like yours truly, know a large majority of the clues. For top players like Brad and Ken, they know—with a handful of exceptions–*all* the clues. In a one-on-one fight with normal clues, Ken and Brad would be fighting over every clue: it would come down to who could buzz first.

This isn't fun to play. Nor is it fun to watch. This is why *Jeopardy!*'s tournament of champions is played on much more difficult clues (Harris, 2006). Nonetheless, this is the battlefield where Watson won: "normal" questions that didn't challenge the human players. Instead, it all came down to the buzzer.

John Henry vs. the Buzzing Machine

Unlike QB (Chapter ?reference? ??), while *Jeopardy!* also uses signaling devices, these only work *once the question has been read in its entirety*; Ken Jennings (also a former QB player while he was a student at BYU) himself explains it on a *Planet Money* interview (Malone, 2019):

> **Jennings:** The buzzer is not live until Alex finishes reading the question. And if you buzz in before your buzzer goes live, *you actually lock yourself out for a fraction of a second*. So the big mistake

> on the show is people who are all adrenalized and are buzzing too quickly, too eagerly.
>
> **Malone:** OK. To some degree, *Jeopardy!* is kind of a video game, and a *crappy video game where it's, like, light goes on, press button*—that's it.
>
> **Jennings:** (Laughter) Yeah.

Jeopardy!'s buzzers are a gimmick to ensure good television; however, QB buzzers discriminate knowledge.

So how does this interact with Watson? Watson receives all of the clues electronically and likewise gets an electronic signal to know when it is safe to buzz in. In contrast, humans have to either guess when it is safe to buzz or wait for a light to turn on.

JBG: citation needed for gurus

Jeopardy! gurus explicitly advise new players not to wait for the light—your puny human reflexes are too slow. Indeed, one of Ken Jenning's strengths was his uncanny ability to internalize the cadence of Alex's voice and when a technitian would activate the buzzer (Jennings, 2006). In contrast, Watson is literally an electromechanical buzzing machine that could get first crack at every question it wants.[1]

Unfortunately, despite subjecting everyone within earshot to these rants, the computer science community thinks that the question is settled: computers are better than humans at answering questions. This is despite Ken basically saying that it did, indeed, come down to the buzzer.

Moreover, what makes for a "normal" difficulty question for a human does not always apply to a computer. Let's first talk about what makes a clue easier for a computer and then we'll talk about what makes a clue harder for a computer.

Easy for a Computer. When I appeared on *Jeopardy!*, my final *Jeopardy!* was:

> After this woman's death, her daughter wrote, "As far as we in the family are concerned, the alphabet now ends at Y"

All of us got the question right; it just so happened that *Jeopardy!* used a very similar clue that aired as we were recording:

> "G" is for grand master as well as this woman who received the 2009 Grand Master Award.

The correct response is of course Sue Grafton. For poorly read contestant like myself, only studying previous clues allowed us to get the answer right. I've never read a Grafton book, but I know she writes mystery books and has titles of the form *"A" is for Alibi* (and that's the only title I can think without looking at Wikipedia).

[1] In practice, this is not always true. Because Watson computed its responses in real time, it could not come up with a response in time for particularly short questions.

For Watson, this is "memorization" is trivial: a single letter implies that the answer is Grafton. But just like you should not think that I'm smart for getting lucky to have seen a reused clue, you should not praise Watson for finding near-repeat questions. And it's not just trivia games: Google's dataset of questions (which we'll talk about in more detail the next chapter) have many identical questions (Lewis et al., 2021). Moreover, Watson can also store the entirety of Wikipedia, easily looking up capitals, authors, etc.

Indeed, when a computer can find *an exact quote* (as was in my final *Jeopardy!* clue), the question becomes even easier. Then the computer just needs to find the appropriate article that contains the quote and then just find whatever entity is mostly likely to be a *Jeopardy!* answer.

Where systems like Watson struggle are on computation, matching novels and movies to plots, combining multiple clues, lateral thinking, and wordplay (Kaushik and Lipton, 2018). And this is not just a matter of degree: computers struggle with *all* such questions, even if they're in the top row of *Jeopardy!*. While a computer is theoretically good at math, the kinds of programs that answer trivia questions struggle answering match questions with numbers in the double digits (Wallace et al., 2019b).

This is why the goal of AI is *general* artificial intelligence (Chapter ?reference???): while we can build specialized systems for *Jeopardy!* questions or math questions, unlike a reasonably smart human, a single program can't "do it all". Unlike for human contestants, the "difficulty" of *Jeopardy!* questions for a computer has no relationship to the nominal value. We talk about how Facebook/Meta dealt with this problem in Chapter ?reference???: tell the authors what's hard for a computer.

Two Nice Guys, One Computer with no Shame

Given the "too easy" questions and the buzzer, what does this actually mean for gameplay? A question comes in, and Watson has the choice of answering it or not: it can win every race to the buzzer if it wants. Then, of the things it cannot answer, Ken and Brad fight over the scraps. Thus, for a computer to win this competition, it needs only to be able to answer a third of the questions correctly.

Now, the *Jeopardy!* nerds reading the book (I love you all), will point out that this isn't true, because the clues are not weighted equally: some are worth more than others. However, as we discussed above, what's difficult for a computer isn't always difficult for a human (and *vice versa*), so it really is a random third of the questions. While a good human player might be weak on the buzzer and be confident that if they know more they'll win the harder clues, this isn't true for a human facing off against Watson.

Moreover, the computer has no shame: it uses a strategy called the "Forrest Bounce" (Rogak, 2020, more infamously associated with James Holzhauer and Arthur Chu). Rather than going through the categories top to bottom (easy to hard), Watson goes through the clues somewhat randomly, searching for Daily Doubles and trying to optimize its score. Again, there's nothing wrong with this—it's the optimal strategy! But if it's the "right" way to play the game, why doesn't every human do it?

That's because humans want to follow social norms. The producers of the show (hi, Maggie and Corinna!) tell you up and down that you shouldn't play the game like that, and you don't want to make them unhappy with you. . . they can make your life miserable. I remember watching *Jeopardy!* with my grandmother and when someone hunted for the Daily Double, she would always say "who does he think he is" (and it was always a he). Alex Trebek also wasn't a fan (Marchese, 2018):[2]

> When the show's writers construct categories they do it so that there's a flow in terms of difficulty, and if you jump to the bottom of the category you may get a clue that would be easier to understand if you'd begun at the top of the category and saw how the clues worked. I like there to be order on the show, but as the impartial host I accept disorder.

And nobody, nobody, wants to make Alex unhappy.

Ken would sometimes do a little hunding for the Daily Double against a particularly formiddable opponent, but he would normally be well-behavied so as not to upset the powers that be. Watson, however, was a soulless machine; and having a machine on the stage was no exciting that nobody faulted it for its strategy. If anyone is to blame, it's probably Gary Tesauro; adding his strategy for playing the board increased Watson's win percentage considerably (Tesauro et al., 2013). But if you put him in a room with the withering stares of *Jeopardy!* producers (or worse, Alex Trebek) and that code would be deleted in no time.

The Legacy of Watson

Let's review Watson's appearance on *Jeopardy!*:

1. all questions are of "normal" difficulty;
2. thus the two human contestants know nearly all of the clues; but
3. Watson can win the buzzer race whenever it wants.

If Watson wins such a match, does that mean that it is superior to these supurb humans?

[2] Rogak (2020) quotes a saltier take Trebek offered to Howard Stern: "It only works, dickweed, if you know the correct response to everything that's up there."

I hope that you are reluctant to answer "yes" (not just because *Jeopardy!* has trained you to respond to answers with questions). Perhaps I've planted a seed of doubt: *no, we cannot yet conclude computer superiority* from this experiment. While many might want to believe that Watson is the end of human–computer question answering competitions, it's only the beginning. Watson was just the beginning of the story, a story that unfolds further in the next few chapters.

While *Jeopardy!* and quiz bowl are the domain of trivia nerds, what happens when computer scientists set the stage to decide whether humans or computers are smarter? Do humands stand a chance on *that* battlefield?

9
Found QA Data: Are they rough or diamonds?

The cliche is that data are the new oil, powering AI. Fortunately, because humans naturally ask questions, there are many datasets that we can find 'for free'. However, these datasets still come at a cost: many of these datasets have inherent problems (e.g., ambiguities and false presuppositions) or oddities (only talking about American men) that make them difficult to use for question answering.

10
Build Your Own Dataset

If you can't find the data you need, build it yourself. But this is not always a perfect solution. This chapter discusses the datasets that people have built and the problems that this can create.

11
Answering Questions through Knowledge Bases

The first question answering approaches were about extracting specific nuggets from a database using natural language. These first approaches are not just historically important but are still influencing many question answering approaches of today. This chapter reviews how we can turn natural language queries into actionable queries in databases.

12
Machine Reading

However, putting information in a database is difficult and time-consuming. . . not all information is in a database (indeed, some information defies strict database schemas). Thus, we need to teach computers how to read. This chapter reviews the process of "machine reading", where computers find information in a large text corpus and then extracts the answer from it.

13
The Advent of Deep Learning

As deep learning became practical, the field has moved from representing text with discrete words to continuous vectors. This is also the case for question answering. This chapter reviews how these representations can help us find relevant answers to questions.

14
Machine Reading

The deep learning revolution not just helps us find answers but to generate them. This chapter talks about the promise and peril of these approaches: how we can synthesize much richer information, make stuff up, encourage these agents to bow to our wishes, and how it's hard to tell if an answer is any good.

15
Siri takes the SAT

How do we know how smart a machine is? Like a human, we typically give it a test; the difference is that tests for computers are called 'leaderboards'. This chapter talks about the pros and cons of leaderboards and how some of the methods used to analyze human standardized tests can help us understand the strengths and weaknesses of computer question answering, and how human trivia tournaments can help make deciding the smartest computer more efficient.

In a previous video, I talked about how questions created modern civilization: the use of exams to select civil servants. Compared to the alternative that preceded them, I think these tests were an unmitigated good.

Today, we're talking about something where the verdict of history is less clear-cut: standardized tests and the psychometric analysis that is applied on top of them. When standardized tests were introduced, the ideal was that these tests could make admissions more scientific and less random or connected to family ties.

At the end of the 19th century, a couple of universities banded together to create the "College Board" to administer tests to see who could get into universities like Harvard, Yale, and Princeton. Do well on the test, and you get in!

We'll talk about the math in the next video, but for now I want to focus on the promise and the big picture of standarized testing and how it relates to the leaderboards that have taken over machine learning.

Let's talk about what the College Board and later Educational Testing Service promises: anybody can sign up for a test whenever they want, and it gives a number that pops out that accurately describes how smart they are. Let's call that number theta to stand for the skill of a student.

So can you just report accuracy? Would that be a good way of getting theta? No! The test in January might not be the same as the test in July. So the mathematical challenge is how to create that estimate theta so that it's consistent against different tests and to select questions from the pool of possible questions so that you can reliably generate that number theta. And that's where the "standardized" in standardized tests comes from.

The process has changed over the years... at first you had to give everybody the same test at each offering of the test, but now they do adaptive testing to do a better job of estimating theta for a specific test-taker. But the two big problems remain the same: making theta standardized and describing in general what kind of questions they should be asking.

JBG: Need to also connect to leaderboards

So why am I, a machine learning researcher, talking about this? Is it because I want to talk about how Harvard added admissions essays to standardized tests because they were concerned that too many Jews were getting in? Or because the federal government deemed UCLA's admissions policies racist because not enough qualified Asians were getting in? Or how there's now a backlash against standardized tests because African Americans systematically have lower scores than other groups? Nope! As animated as I get about these topics, these are decidedly out of scope of this video.

But I mention them because they're definitely real problems, and causes a lot of people to hate standardized tests. But it's not a problem with the *math* behind standardized tests. It's a problem with the people who write the questions and what happens when the results come out. And QA systems aren't immune from these problems either, as I talk about elsewhere. But the math is sound.

Hopefully it reminds you of a leaderboard: systems come in, you need to give them a score. But the machine learning community has a distinct advantage here. You can wipe the memory of a computer system, so you can give it the same test over and over again. But as I argued before in my plea for an embrace of the Manchester paradigm from trivia shows, this isn't always the case: there's still some information that gets leaked.

So we'll get into the math soon. But it's not just for creating leader boards in machine learning and natural language processing. Very similar models are used for rating players in chess and other games and for predicting how politicians will vote on particular bills.

Leaderboards are Shiny

Leaderboards are widely used in NLP and push the field forward. While leaderboards are a straightforward ranking of NLP models, this simplicity can mask nuances in evaluation items (examples) and subjects (NLP models). Rather than replace leaderboards, we advocate a re-imagining so that they better highlight if and where progress is made. Building on educational testing, we create a Bayesian leaderboard model where

Figure 15.1: Difficulty and Ability Discriminating (DAD) leaderboards infer the difficulty, discriminativeness, and feasibility for examples. Negative discriminability suggests an annotation error; for example, the question with most negative discriminability asks "Why did demand for rentals decrease?" when the answer is "demand for higher quality housing increased."

latent subject skill and latent item difficulty predict correct responses. Using this model, we analyze the ranking reliability of leaderboards. Afterwards, we show the model can guide what to annotate, identify annotation errors, detect overfitting, and identify informative examples. We conclude with recommendations for future benchmark tasks.

Leaderboard evaluations—for better or worse—are the *de facto* standard for measuring progress in question answering (**?**) and in many NLP tasks (**?**). An unfortunate side effect of leaderboard popularity is SOTA-chasing, often at the expense of carefully inspecting data and models (**?**). For example, the same "super-human" models that top question answering leaderboards (Najberg, 2018) often fail spectacularly (**??**) by learning non-generalizable statistical patterns (**??**). Finally, focusing *solely* on metrics conflates progress on a specific *task* with progress on real-world NLP *problems* behind the task (**?**). Plainly, focusing on headline SOTA numbers "provide(s) limited value for scientific progress absent insight into what drives them" and where they fail (**?**).

In this work we take leaderboards "as they are," and imagine how they might better support research. Leaderboards establish differences between models on a fixed task. Hence, leaderboards should enable and encourage the comparison of models and inspection of examples. And leaderboards should also signal when they have outlived their usefulness (**?**).

How to Direct Leaderboards' Light

To help focus attention on examples and models of interest, we propose Difficulty and Ability Discriminating (DAD) leaderboards that explicitly model both task and submissions *jointly*, rather than either in isolation. DAD's underlying model is based on Item Response Theory (**??**, IRT, reviewed in §15), a widely used (**?**) alternative in educational testing to simple summary statistics (**?**).

DAD can explicitly identify the *difficulty* and *discriminability* of items (Figure 15.1),[1] which in turn can lead to a more nuanced ranking of models, identifying poor items, and better understanding of a dataset and task. Throughout the paper, we use the question answering (QA) benchmark SQuAD 2.0 (**?**). For example, DAD can identify questions that are challenging to models and questions that are wrong (incorrectly annotated). In addition to better understanding datasets, it is also helpful for efficiently selecting evaluation items to annotate. We conclude with recommendations for future leaderboards (§15) and discuss where IRT in NLP can go next (§15).

[1] Example and feasibility distribution in Appendix 15. Interactive visualization linked from http://irt.pedro.ai.

How the SAT can Help Leaderboards

If you've ever applied to a university, you've probably had to take a standardized test. As we talked above, the "standardized" in standardized test means that the score you get makes sense no matter when you took the test. So how does the math work that lets that happen?

The big idea is that we want to model the probability of a person j getting question i right. We're going to treat this as a logistic regression. As with every logistic regression, you pass some number through a sigmoid function to generate a probability. The number that you pass in this function—like a feature in a logistic regression—helps explain your data. And in the case the data are a big matrix of who got what questions right and who got what questions wrong.

In the lingo of IRT, the questions are the "items", the agents answering the questions are the "subjects" and then whether they got a question right or not is the "response". So now hopefully you see what it's called item response theory. The goal is to predict each of these binary responses as if it were a logistic regression.

So like all logistic regression, the prediction comes out of a sigmoid function that you hopefully remember. So what goes *in* to that sigmoid function? We'll add more parameters soon, but let's start with the first two: difficulty and skill.

For concreteness, let's assume that we have a question-answering task and two subjects: Ken, who is good at trivia, and Burt, who is not. In the simplest IRT models, each subject j has a random variable θ_j corresponding to their skill: Ken's is big, Burt's is small. The higher the skill number (theta), the more likely they are to answer questions correctly.

But you cannot know that until you start asking them questions of varying *difficulty* β_i. Harder questions have a higher difficulty ("what is the airspeed of an unladen swallow") than easy ones ("who is buried in Grant's tomb"). The bigger the margin between a subject's skill θ_j and an item's difficulty β_i, $\theta_j - \beta_i$, the more likely that subject j responds correctly $p_{i,j}(r_{i,j} = 1)$.

So how do these two numbers feed into a sigmoid function? You take the difference between the difficult and the skill. When the skill minus the difficulty is high, the higher the probability of the subject answering it correctly. If the skill matches the difficulty, then the subject has a 50-50 chance of answering correctly.

Okay, so let's make this a little more complicated and add two additional terms: discriminability and feasibility: discriminability γ_i and feasibility λ_i.

Discrimination Discriminability is how well the question distinguishes better subjects from worse ones. It's multiplied by the difference between the skill and the difficulty. Mathematically, this changes the slope of the response curve. A discriminative question is challenging but can still be answered correctly by a strong subject. If Ken's ability is higher than most items' difficulty ($\theta_j - \beta_i$ is large), item discriminability multiplies this gap by γ_i. Questions with low γ_i are low quality: they have annotation error or do not make sense.

Here's a picture of that for a question with zero difficulty. As the skill (x-axis goes up), the more likely the subject is to answer the question. The bigger the discriminability, the steeper the curve. Here it is for 0.5 and here it is for 2.0.

This would never happen in practice, but let's imagine you could create a test set where every question had perfect discrimination. Then the response curve won't be a curve, it'll be a step function. So let's say you have a question with difficulty 0.2 and perfect discrimination. Anybody with skill greater than 0.2 will get it right 100% of the time.

But if you can do it once, you can do it again. Let's create several perfect discriminators with a nice grid of difficulty. So now, when a person comes in, they'll answer every question correct until their skill is surpassed by the question. First question they get wrong means they can leave the test and go home early.

Again, this fantasy is impossible, but you can imagine why folks want these sorts of questions on a standardized test. There's more information about a subject's skill for each question.

Feasibility Another way of capturing poor quality questions is the feasibility λ_i. For example, if the question "who was the first president" has the answer Rajendra Prasad, the question has an unstated implicit assumption that subjects must guess what country or company the question is about. In the model **TODO, FIX ME!!!!** , if a large fraction of subjects all get an item wrong, everyone's probability of getting the item right is capped at λ_i. In NLP terms, $1 - \lambda_i$ corresponds to the prevalence of annotation errors that lead to unsolvable items.

Let's now see what feasibility does. Feasibility is relevant to NLP since infeasible questions suggest an annotation error. Let's go back to our favorite "bad" question: what is the capital of Georgia? If this sort of question can have the answer "Atlanta" or "Tblisi" with equal probability, then nobody, no matter how smart they are, can get this right with more than 0.5 probability.

Mathematically, see how the feasibility parameter works is more straightforward. It's just a linear scaling, that brings down the maximum value the probability can take on. The normal logistic curve is on top, lambda=1, but it gets squished down as lambda gets smaller and smaller. If lambda=½, then no matter how smart you are, it's basically a coin flip whether you can answer the question.

Figure 15.2: A DAD leaderboard uses IRT to jointly infer item difficulty β_i, discriminability γ_i, feasibility λ_i, and subject skill θ_j. These predict the likelihood $p_{ij}(r_{ij} = 1)$ of a correct response r_{ij}.

A Generative Story for Leaderboards

Leaderboards are a product of the metrics, evaluation data, and subjects (machine or human) who answer items (Figure 15.2).

Generally, given n test items $\mathcal{X} = (X_1, \ldots, X_n)$ and m subjects $\mathcal{S} = (S_1, \ldots, S_m)$, where each subject answers every item, we want to estimate subject skills and item difficulties. To discover the random variables that best explain the data, we turn to probabilistic inference Pearl (1988).

Having introduced all of the constituent elements of the model, we can now present the full generative model:

1. For each subject j:
   (a) Draw skill $\theta_j \sim \mathcal{N}(\mu_\theta, \tau_\theta^{-1})$
2. For each item i:
   (a) Draw difficulty $\beta_i \sim \mathcal{N}(\mu_\beta, \tau_\beta^{-1})$
   (b) Draw discriminability $\gamma_i \sim \mathcal{N}(\mu_\gamma, \tau_\gamma^{-1})$
   (c) Draw feasibility $\lambda_i \sim \mathrm{U}[0, 1]$
3. Draw subject i response on item j,
   $r_{ij} \sim p_{ij}(r_{ij} \mid \theta_j, \beta_i, \lambda_i) =$

$$p_{ij}(r_{ij} = 1|\theta_j) = \frac{\lambda_i}{1 + e^{-\gamma_i(\theta_j - \beta_i)}}. \qquad (15.1)$$

For TODO, FIX ME!!!! , γ_i and λ_i are fixed to 1.0, while for TODO, FIX ME!!!! , only λ_i is fixed.[2]

Means $\mu_\theta, \mu_\beta, \mu_\gamma$ are drawn from $\mathcal{N}(0, 10^6)$ and $\tau_\theta, \tau_\beta, \tau_\gamma$ from a $\Gamma(1, 1)$ prior, as in **?** and recommended by **?**.[3]

Because it is difficult to completely codify skill and difficulty into a single number, we can rewrite the exponent in Equation 15.1 as a sum over dimensions $-\gamma_i(\sum_k \boldsymbol{\theta}_{j,k} - \boldsymbol{\beta}_{i,k})$, where each dimension captures the interaction between an item's difficulty and a subject's skill. For example, perhaps Burt could better exploit artifacts in one dimension (their skill for $\theta_{j,k=5}$ is high but everywhere else is low) while Ken might not know much about a particular topic like potent potables ($\theta_{j,k=2}$ is low but everywhere else is high). We call this model **TODO, FIX ME!!!!** .[4] Multidimensional IRT models (**?**) could—in addition to better modeling difficulty—also cluster items for interpretation; we briefly experiment with this (Appendix 15), but leave more to future work (§15).

[2] In psychometrics, TODO, FIX ME!!!! is called a Rasch (**?**) or 1 parameter logistic (1PL) model, TODO, FIX ME!!!! is a 2PL model, and TODO, FIX ME!!!! is a 4PL model with guessing set to zero.

[3] We differ by allowing $\gamma < 0$ to identify bad items.

[4] We do not incorporate feasibility into the TODO, FIX ME!!!! model since it already improves over 1D models without it.

Examples are Not Equally Useful

IRT's fundamental assumption is that not all items and subjects are equal. This explains why leaderboards can fail while having "normal looking" accuracies. As a thought experiment, consider a dataset that is one third easy ($\beta_i \in [0, 1]$), one third medium difficulty ($\beta_i \in [2, 3]$), and one third hard ($\beta_i \in [6, 7]$). Suppose that Ken has skill $\theta_k = 4$ while Burt has skill $\theta_b = 2$. A standard leaderboard would say that Ken has higher accuracy than Burt. But suppose there's a new subject that wants to challenge Ken; they are not going to reliably dethrone Ken until their skill θ_c is greater than six.

This is a more mathematical formulation of the "easy" and "hard" dataset splits in question answering (**???**). In TODO, FIX ME!!!! , this recapitulates the observation of **?** that annotation error can hinder effective leaderboards. DAD helps systematize these observations and diagnose dataset issues.

Inference

To estimate the latent parameters of our model, we use mean-field variational inference Jordan et al. (1999). In variational inference, we propose a distribution over the latent variables, $q_\phi(\cdot)$, that approximates the true but intractable posterior $p(\cdot)$. We then minimize the KL-divergence between these distributions, equivalent to maximizing the evidence lower-bound (ELBO) with respect to the variational parameters.

In our case, $q_\phi(\cdot)$ is a mean-field distribution, which means it factorizes over each of the latent variables (the product is over the $n \times m$ subject-item pairs)

$$q_\phi(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\mu}, \boldsymbol{\tau}) = q(\boldsymbol{\mu})q(\boldsymbol{\tau}) \prod_{i,j} q(\theta_j)q(\beta_i)q(\gamma_i)$$

Specifically, for our key latent variables $z \in \{\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma}\}$, the associated variational distributions are of the form $q(z) = \mathcal{N}(u_z, t_z^{-1})$. Recall that in the generative distribution, each latent z is drawn from a $\mathcal{N}(\mu_z, \tau_z^{-1})$ whose parameters are also latent variables; for these variables, we use the variational distributions $q(\mu_z) = \mathcal{N}(u_{\mu_z}, t_{\mu_z}^{-1})$ and $q(\tau_z) = \Gamma(a_{\tau_z}, b_{\tau_z})$. We optimize the ELBO with respect to the variational parameters

$$\boldsymbol{\phi} = \{\boldsymbol{u}_z, \boldsymbol{t}_z, \boldsymbol{u}_{\mu_z}, \boldsymbol{t}_{\mu_z}, \boldsymbol{a}_{\tau_z}, \boldsymbol{b}_{\tau_z}, \boldsymbol{\lambda}\}$$

for all z using ADAM (**?**).

With DAD's leaderboard IRT model introduced, we next discuss how leaderboard subjects are statistically compared and alternative methods—such as using IRT parameters—to evaluate whether two models are truly different.

Ranking and Comparing Subjects

Fundamentally, the objective of comparative evaluations like leaderboards is to decide whether model A is better than model B. A thread of NLP has rightfully advocated for adding rigor to these decisions using statistics (**?**, Classical Testing Theory) where the objective is to infer a true score T from the observed test score $X = T + E$ given a measurement error E, uniform across subjects. However, in educational testing—a field measuring skill and knowledge in humans—IRT is a primary measurement instrument (**?**, p. 2). A major motivation for IRT is that subjects of different skill have *different* errors. IRT explicitly accounts for the bandwidth-fidelity dilemma (**?**): items can either accurately measure a narrow ability range (fidelity) *or* inaccurately measure large ability ranges (bandwidth).[5] This section and the next contrast methods for identifying the best model and advocate for IRT.

Implicit in nearly all leaderboard evaluations is ranking models by a statistic such as the average accuracy. As we show in §15, naïve rankings are noisier than IRT rankings.

IRT for Leaderboards

Leaderboards should: (1) reliably and efficiently rank better models ahead of worse models (**??**) and (2) guide inspection of items and subjects (§15). The first ameliorates the unavoidable randomness of finite evaluations while the second enables error analysis (**?**) and model probing (**??**). First we verify that IRT models accurately predict the responses of subjects (§15). Next, a ranking stability analysis shows that IRT has modestly better reliability than classical rankings (§15). Lastly, using IRT to actively sample items for annotation yields rankings with better correlation to complete test data (§15).

Why a Linear Model Baseline

At first blush, the differences between IRT and logistic regression are minimal, but we include the comparison to address natural questions from the NLP community: (1) do the idiosyncrasies of the IRT formulation hurt accuracy? (2) should we add features to better understand phenomena in the questions? (3) why not use deep models?

The next section argues that both IRT and logistic regression are accurate even without laboriously engineered task-specific features. Adding obvious features such as item words (e.g., questions) only minimally improves the accuracy. We explicitly omit less interpretable deep models since our goal is to make leaderboards *more* interpretable.

[5] Estimation error of θ varies by position (Appendix 15).

Response Prediction is Accurate

Just as educational testing researchers validate IRT models by seeing if they predict subject responses correctly (**?**), we validate how well DAD predicts whether SQuAD models get questions right.

We compare against a logistic regression linear model (LM) implemented with Vowpal Wabbit (**?**). Since integrating hand-crafted features is easy, we incorporate features derived from subject IDS; item IDS; functions of the SQuAD question, answer, and title; and IRT parameters (details in Appendix 15). As in IRT, logistic regression predicts whether a subject correctly responds to an item. Later, we discuss ways to integrate more features into IRT (§15).

SQuAD Leaderboard Data

Experiments are on the SQuAD 2.0 leaderboard. Development data are publicly available, and organizers provide test set responses. There are 161 development subjects, 115 test subjects, and 11,873 items (1.9 million total pairs). Experiments that do not need test responses use all development subjects; those that do use the smaller test subset.

Evaluation Scheme

Following prior work (**?**), we evaluate IRT and linear models by holding out 10% of responses and computing classification metrics.[6] In SQuAD, predicting whether a response is correct is an imbalanced classification problem (80.4% of responses in the development set are correct). Thus, we use ROC AUC, macro F1, and accuracy.

IRT Response Prediction is Accurate

IRT models that incorporate more priors into the generative story should be better, but are they? We compare four IRT models: TODO, FIX ME!!!! , TODO, FIX ME!!!! , TODO, FIX ME!!!! , and TODO, FIX ME!!!! (§15). The more sophisticated models are better and all improve over the LM (Figure 15.3) and correlate well with each other (Appendix 15). To be clear, while higher accuracy than LM is good, our goal is to validate that IRT models are accurate; later, we inspect model errors and identify annotation errors (§15).

Figure 15.3: We compare each IRT and linear model (LM) by how well they predict subject responses. We focus on ROC AUC since predicting responses is an imbalanced classification problem (most subjects are correct). Under that metric, all IRT models improve over the best LM, and the strongest LM ablation only uses IRT features. That textual features are predictive in the LM suggests they could improve future models.

[6] Everywhere else in the paper, we train on all responses.

Figure 15.4: Compared to the final ranking over a large test set, how well does a small test set correlate? The left shows correlation between mutually exclusive development set samples and the right between development samples and the full test set. In both experiments (panes), ranking systems by IRT ability is more stable—across all sample sizes—than mean accuracy and thus more reliable (Kendall's rank correlation is higher). Bands show 95% confidence intervals of rank correlations across ten trials per sample size.

What Model Features are Predictive?

Integrating additional features into Bayesian models is not trivial, so we instead use the flexibility of linear models to identify useful features. Our leave-one-in ablation compares features (Figure 15.3): the top ablations both use IRT features, further validating IRT parameters. The subject and item identifier features are also strongly predictive, but item is the stronger of the two. Text-based features are weaker, but this suggests future work to better integrate them into IRT models (§15).

Ranking with IRT

Leaderboards should produce *reliable* subject rankings: can DAD rank systems even with a tiny test set? Thus, we compare the correlation both of traditional average accuracy (§15) and IRT rankings on the whole test set compared to the rankings of the same metric on a smaller test set. Our first experiment (§15) examines the stability of existing items and subjects while the second (§15) investigates stability of "new" evaluation data using sampling strategies.

IRT Rankings Have Better Reliability

Rankings should be reliable within the same dataset (e.g., on dev set) and generalize to similar datasets (e.g., with a test dataset). To test the first, we measure the ranking stability of mutually exclusive samples of the development data **?**. To test the second, we measure the correlation between development set sample rankings to test set rankings (**?**).

Specifically, for a range of sample sizes[7] we (1) sample two partitions of the data, (2) compute the classical ranking[8] and the IRT ranking from a refit TODO, FIX ME!!!! model, then (3) compute Kendall's correlation (**?**) between the samples for each ranking (details in Appendix 15). In both cases IRT rankings have higher correlation than classical rankings (Figure 15.4, left). Since the benefit is strongest at low sample sizes, IRT can improve the reliability of small-scale evaluations.

The second experiment examines ranking generalization: IRT yields more reliable measures of subject skill, implying a greater consistency in subject rankings across evaluation settings. Figure 15.4 compares the development set sample rankings computed above to rankings obtained using subjects' *test* set responses (with the same IRT model).

Across all sample sizes, subjects' IRT ability estimated on the development set correlates well test set ability. Crucially, this is better than the corresponding classical metrics like accuracy (Appendix 15

[7] The sample size must be less than half the size of the development data so that we can obtain two samples.

[8] For SQUAD, ordering by mean exact match score.

Figure 15.5: Suppose we need to cold start and collect annotations for a new subject: what order would most rapidly increase correlation to the full test data? As we expect, the correlations eventually converge, but with little data, IRT has better correlation than other methods. We suspect that the IRT information underperforms early on when the subject ability estimate is unstable.

quantifies the statistical significance of the difference), supporting our original motivation for using IRT.[9]

IRT Improves Cold Start Reliability

IRT can also guide the construction of tests. Just as IRT practitioners prepare tests for humans, we too construct tests for machines. In educational testing, collecting responses from humans is expensive; likewise, although *questions* are cheap in search-based QA tasks (**??**), annotating *answers* is expensive. Likewise, "grading" machine dialog responses is expensive and IRT helps (**?**). To emulate this setting, we use computerized adaptive testing (**?**) to iteratively select SQuAD items to "annotate."

As in human test preparation, we use existing annotations to infer item parameters and iteratively infer the ability of new subjects. This experiment splits *m* subjects into a training group (80%) and a testing group (20%). The training group represents subjects for which we have full item predictions and annotations; the testing group represents a new group of subjects that we need to rank. To efficiently rank, we should iteratively choose items to annotate that yield the most information about the ranking if all the data were annotated.

This experiment compares how well several item selection strategies work. For each selection method, we (1) choose a sample size, (2), sample from the development set, (3) compute the ranking of subjects, and (4) compute Kendall's rank correlation (Figure 15.5).[10]

Which item selection strategies should we compare? As a baseline, we use naïve random sampling. Like prior work, we compare selecting items with the highest difficulty and the highest discriminability (**?**) as well as the sum of the two.[11] We propose that items should be selected according to their Fisher information content (**?**)

$$I_i(\theta_j) = \frac{(p'_{ij})^2}{p_{ij}(1 - p_{ij})} = \gamma_i^2 p_{ij}(1 - p_{ij}) \tag{15.2}$$

as derived by **?**, p. 70.

Intuitively, if we do not yet know the true skill θ_j, we should pick items whose expected response we are most uncertain about. Our uncertainty (entropy) is maximized when the likelihood of a correct response p_{ij} is the same as the likelihood of an incorrect response $1 - p_{ij}$, which corresponds to the maximal value of $I_i(\theta_j)$; it is also sensible this value increases as discriminability γ_i increases.

To infer the maximally informative items, we estimate the ability θ_j of each subject using the currently selected items, use the ability

[9] Since the maximum trial size was limited, we train one final model with the full data, see Table 15.3 in the Appendix 15.

[10] We compute correlations with the complete development set on ten trials to build 95% confidence intervals.

[11] We train an TODO, FIX ME!!!! model to simplify sampling (e.g., avoiding a tradeoff between feasibility and discriminability).

Figure 15.6: We partition evaluation data by IRT difficulty and discriminability with accuracy in each quartile. Most improvements in high-accuracy systems come from getting high-difficulty questions right. Items with low discriminability (and thus prone to annotation errors) are difficult for all subjects except the overfit ARGS-BERT model. We include top-performing SQUAD subjects, several notable subjects (systems), and a pair from the bottom of the leaderboard.

to compute the information of each yet-to-be-annotated item for each subject, and then aggregate the informativeness

$$\text{Info}(i) = \sum_j I_i(\theta_j) \tag{15.3}$$

by item i summed over subjects j. This approach is similar to uncertainty sampling and reduces to it for the TODO, FIX ME!!!! model (**?**). We initially seed with the twenty-five most discriminative items (details in Appendix 15).

Like computerized adaptive testing (**?**), Figure 15.5 shows that at lower sample sizes three of the IRT sampling methods are better than random sampling—difficulty does worse. The other IRT methods have comparable correlation. Thus, by using IRT, DAD can both improve rankings and guide annotation.

Qualitative Insights on Leaderboards

DAD also helps qualitative analysis of items and subjects. First, IRT identifies overfitting and generalizes partitioning datasets by difficulty. Then we show that—like in educational testing—IRT identifies good and bad items.

Guiding Analysis with IRT

Several works curate easy and hard QA subsets based on how many models answer correctly (**?**) or heuristics (**?**). IRT can create similar subsets using TODO, FIX ME!!!! , the best 1D model. Difficulty finds where subjects improve while discriminability and feasibility can surface items that may be invalid. For example, one low feasibility question (Figure 15.10) asks "what are two examples of types of Turing machines?" which has two problems: (1) the answer omits five types and (2) span-based evaluation precludes selecting non-contiguous types.

After excluding items with negative discriminability—they are likely erroneous—we sort items into bins. We break both difficulty and discriminability into four bins—taking the 25th, 50th, and 75th percentiles—creating eight total bins. Then we select representative SQUAD subjects with their exact match scores (Figure 15.6). Let's examine a feasible item with positive difficulty and discriminability like "what reform was attempted following the Nice treaty?"[12] In this case, the annotator's span is too long—resulting in almost no correct answers and a low fuzzy match (token F1). In contrast, one highly discriminative question

[12] A: "there was an attempt to reform the constitutional law of the EU and make it more transparent." (Appendix Figure 15.11)

succeeds because there are multiple plausible guesses to "who did the Normans team up with in Anatolia?"[13] While both the Armenian state and Turkish forces are superficially plausible answers, only Turkish forces is correct; nonetheless, some models are fooled. Using IRT to guide subject analysis is helpful; next, we test how efficient it is in identifying annotation error.

Figure 15.7: We annotate SQuAD items by discriminability, difficulty, and IRT prediction errors. For example, one question with negative discriminability was classified as "Wrong" with the explanation that the annotated answer indicates it is *not* answerable, but the question actually *is* answerable. Items with negative discriminability or where IRT's prediction is wrong have a much higher rate of annotation error ("Flawed" or "Wrong"). Using similar methodology, errors in datasets could be more rapidly identified.

Identifying Annotation Error

To test if IRT can identify annotation error, we inspect sixty SQuAD development set items. We select ten items from each of these groups: the most negative discriminability, discriminability nearest to zero, the highest discriminability, the least difficult, most difficult, and IRT model errors. For each, we annotate whether the item was correct, was "correct" yet flawed in some way, or simply wrong (Figure 15.7).[14] Inter-annotator agreement between three authors on this three-way annotation with Krippendorff's α (**??**) is 0.344. Despite only modest agreement, just as in the development of education tests, negative discriminability is predictive of bad items. When discriminability is negative, then the probability of getting the answer right is higher when ability is *lower*, which is undesirable: Ken consistently loses to Burt on those items. This could identify bad items in evaluation sets for removal.

A Re-Imagined Leaderboard Dashboard

pr: Move to separate demo paper

Basing a new leaderboard on IRT models has another significant benefit: as they are purposefully interpretable, they are very useful for visualizing leaderboard data. The final component of this work is a demonstration—based on SQuAD—of components that a re-imagined leaderboard may have.[15] Our proposed visualization has three main components: (1) an improved model listing, (2) a model-centric view, (3) an example-centric view.

The majority of leaderboards list only one ranking metric or list multiple while only ordering models by a primary metric. However, multiple factors—including multiple metrics—should be integrated such as model size and efficiency (**?**). This presents a challenge: how should all these be visualized? In our proposed model listing, while there would be a default ranking, our visualization would support ranking by any metric or combinations of rankings while showing the

[13] Example with statistics in Appendix Figure 15.12

[14] Annotation guidelines provided in supplementary materials; Figure 15.7 uses the first set of annotations which were later augmented by two additional sets of annotations.

[15] The visualization doubles as my course project for CMSC734: Information Visualization.

influence of each as in the parallel axis plots in LineUp (?).

Figure 15.8: A proof-of-concept visualization that compares two models—a BERT and IR retrieval system—(?) visualizing which examples model agree or disagree on. We propose extending this to incorporate more than two models.

Similarly to Manifold (?), the goal of the model-centric view is to understand the differences—based on predictions—between models. An initial approach (Figure 15.8) compares how pairs of models differ, but could be extended to support multiple models by enhancing the underlying software. Another approach to a model-centric views instead attempts at characterizing the performance along partitions of the data (?). For example, **?** propose faceted histograms as a way to "un-aggregate" metrics to inspect QA accuracy by factors like question type and confidence. In addition to visualizing which instances pairs of models agree as in Manifold, we propose integrating IRT parameters to better identify instances that have a larger influence over the output ranking.

A dominant approach to inspecting models examines their performance on individual examples or partitions of examples sharing a trait (?). In addition to domain-relevant traits, we can guide example sample selection with IRT which makes it more likely to identify important issues over random sampling (?) while still combining careful specification of error types (?). Similarly, visualizations can use the multidimensional IRT parameters for clustering which can also be combined with domain-specific attributes like question type or topic.

A major advantage of our approach—embedding these comparisons in leaderboards—is that model developers can compare the outputs of many models on individual or small sets of examples. While there are numerous tools for comparing models, error analysis, and related tasks, they face the headwinds of convincing practitioners that they are worthwhile time investments. Instead, we aim to provide real benefits of leaderboard-wide analysis without incurring up-front time commitment from practitioners. Although this does require commitment from task organizers, we believe that the trade-off of inherently improving the evaluation while providing practitioners with "shiny tools" is well worth it.

Related Work

DAD draws together two primary threads: we use IRT to understand datasets, which has been applied to other NLP tasks, and apply it to improving leaderboards. Finally, we explore how the insights of IRT can improve not just the analysis of test sets but to improve the *construction* of test sets.

irt *in* nlp IRT is gaining traction in machine learning research (**??**) where automated metrics can be misleading (**?**): machine translation (**?**) and chatbot evaluation (**?**). Concurrent with our work, **?** compare NLP test sets with IRT. Closest to our work in NLP is **?**, who rank machine translation subjects and compute correlations with gold scores. Similarly, **?** use IRT on non-language AI video game benchmarks. Just as we use IRT to identify difficult or easy items, **?** create challenge sets for textual entailment. We test IRT as a way to guide annotation, but it can also train NLP models; for example, deep models learn "easy" examples faster (**?**) and maintain test accuracy when training data are down-sampled (**?**).

Improving Leaderboards The rise NLP leaderboards has encouraged critical thought into improving them (**?**), improving evaluation more broadly (**?**), and thoughtful consideration of their influence on the direction of research (**??**). DAD aims make leaderboard yardsticks (**?**) more reliable, interpretable, and part of curating the benchmark itself. In line with our reliability goal, just as statistical tests should appear in publications (**??**), they should be "freebies" for leaderboard participants (**?**). Alternatively, **?** posit that leaderboards could be automatically extracted from publications. How to aggregate multi-task benchmarks (**???**) and multi-metric benchmarks (**?**) is an open question which—although we do not address—is one use for IRT.

This work implicitly argues that leaderboards should be continually updated. As a (static) leaderboard ages, the task(s) overfit (**?**) which—although mitigable (**??**)—is best solved by continually collecting new data (**?**). Ideally, new data should challenge models through adversarial collection (**??**) and related methods (**?**). However, if making an easy leaderboard more difficult is possible, the leaderboard has outlived its helpfulness and should be retired (**?**).

Part of our work centers on alternate task efficacy rankings, but this naïvely assumes that task efficacy is the sole use case of leaderboards. Indeed, focusing solely these factors can mislead the public (**?**) and may not reflect human language capabilities (**?**). Leaderboards are also well positioned to provide incentive structures for participants to prioritize fairness (**?**) and efficiency (**???**) or incorporate testing of specific capabilities (**??**). To enable these more nuanced analyses, leaderboards should accept runnable models rather than static predictions (**?**).

Active Learning Beyond IRT, the analysis of training dynamics and active learning (**?**) is helpful for actively sampling specific items or identifying low-quality items (**?**). For example, **?** and **?** propose alternative training dynamics-based methods for identifying difficult items as well annotation errors. Even closer to goals, **?** use active learning to build a test collection. Explicitly measuring how effectively examples separate the best subject from the rest allows test set curators to "focus on the bubble" (**?**), prioritizing examples most likely to reveal interesting distinctions between submitted systems.

Alternate Formulations IRT is an example of convergent evolution of models that predict subject action given an item. Ideal point models **?** consider how a legislator (subject) will vote on a bill (item) and use a similar mathematical formulation. The venerable ELO model Glickman and Jones (1999) and modern extensions **?** predict whether a player (subject) will defeat an opponent (item) with, again, a similar mathematical model. Certain IRT models can also be formulated as nonlinear mixed models **?**, where the item parameters are fixed effects and the latent subject parameters are random effects. This allows for comparisons between IRT models and other mixed effects models under a consistent framework. **TODO, FIX ME!!!!** and **TODO, FIX ME!!!!** can be formulated as nonlinear mixed models, and **TODO, FIX ME!!!!** can be formulated as a discrete mixture model over items. As we discuss further in the next section, DAD's application of IRT can further be improved by adopting interpretable extensions of these models.

Conclusion

This paper advocates incorporating decades of research in crafting education tests to improve how we evaluate the capabilities of NLP models. We propose and validate an alternate IRT ranking method for leaderboard evaluations, show it can guide annotation, detect annotation error, and naturally partition evaluation data. Just as educators moved from classical testing to IRT, the NLP community should consider future evaluations with IRT.

Limitations

Although there is much to gain through IRT evaluation, there are limitations which make it hard to implement. First, it requires access to item-level responses for all examples for all subjects which are often only available to organizers. Second, **?** notes that sampling mutually exclusive subsets has drawbacks—samples are not entirely independent. Lastly, our work is a proof of concept using SQuAD 2.0 as a test bed and our results may not generalize.

Future Work

We see a few directions for future work. First, this paper is intended to validate IRT and its usefulness as an active part of the leaderboard lifecycle; the natural next step is to implement it in a leaderboard. Second, our IRT models do not incorporate the item content (e.g., example text) to predict responses, but in principle could; Bayesian models with metadata (**?**) and ideal point models from political science (**?**) that incorporate bills and speeches do exactly this (**???**). Analogously, IRT for leaderboards can and should also incorporate text from passages, questions, and answers to better model what makes questions difficult. Such a model can also predict which characteristics would create discriminating or difficult items. Lastly, multidimensional IRT models to evaluate multiple skills could aid multi-task or multi-metric leaderboards like MRQA (**?**) and Dynaboard (**?**).

Acknowledgements

For their work on early iterations of leaderboard visualizations, we thank Jacob Bremerman and Wei Wei Chi. For insightful discussions and ideas we thank Shi Feng, Doug Oard, João Sedoc, Mike Wu, and Patrick Lewis. We thank Peter Rankel for recommendations on statistical testing methods. For discussion and feedback on visualizations, we thank Leo Zhicheng Liu, Calvin Bao, and classmates in UMD's Fall 2020 "Information Visualization" course. For suggestions on topic modeling, we thank Philip Resnik and Maria Antoniak. For feedback on prior versions of this paper, we thank our anonymous ACL reviewers and members of the UMD CLIP lab. Boyd-Graber and Rodriguez's work at UMD were supported by NSF Grant IIS-1822494. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsor. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

SQuAD Item Examples

Figures 15.9, 15.10, 15.11, and 15.12 show previously discussed SQUAD examples (§15) in full. On the same page, we provide a web interface for inspecting the parameters of the IRT models. Figure 15.13 shows the feasibility distribution corresponding to Figure 15.1.

Discriminability: -9.63 **Difficulty**: -0.479 **Feasibility**: 0.614 **Mean Exact Match**: 0.472
Wikipedia Page: Economic inequality **Question ID:** 572a1c943f37b319004786e3
Question: Why did the demand for rentals decrease?
Official Answer: demand for higher quality housing
Context: A number of researchers (David Rodda, Jacob Vigdor, and Janna Matlack), argue that a shortage of affordable housing – at least in the US – is caused in part by income inequality. David Rodda noted that from 1984 and 1991, the number of quality rental units decreased as the demand for higher quality housing increased (Rhoda 1994:148). Through gentrification of older neighbourhoods, for example, in East New York, rental prices increased rapidly as landlords found new residents willing to pay higher market rate for housing and left lower income families without rental units. The ad valorem property tax policy combined with rising prices made it difficult or impossible for low income residents to keep pace.

Figure 15.9: The example from SQUAD with the lowest discriminability. Surprisingly, it had a *negative* discriminability, implying that the less skilled a subject is, the more likely their response is to be correct.

Discriminability: 3.24 **Difficulty**: 3.86 **Feasibility**: 0 **Mean Exact Match**: 0
Wikipedia Page: Computational Complexity Theory **Question ID:** 56e1b00ce3433e1400422b0a
Question: In the determination of complexity classes, what are two examples of types of Turing machines?
Official Answer: probabilistic Turing machines, non-deterministic Turing machines
Context: Many types of Turing machines are used to define complexity classes, such as deterministic Turing machines, probabilistic Turing machines, non-deterministic Turing machines, quantum Turing machines, symmetric Turing machines and alternating Turing machines. They are all equally powerful in principle, but when resources (such as time or space) are bounded, some of these may be more powerful than others.

Figure 15.10: This question is regarded as infeasible by the IRT model. Upon further inspection, the answer omits five acceptable answers, but more importantly does not permit all combinations of Turing machines.

Logistic Regression Features

The linear model (§15) includes features based on item IDS, subject IDS, textual features of the question, context, and answer, and topic model features. Table 15.1 lists the feature names from Figure 15.3 with descriptions of each. When IRT features or the statistics features are used, they include interaction terms with themselves.

IRT Model Type Correlation

Although each IRT model differs in expressiveness, they should—in general—produce similar results. This is confirmed by computing the Kendall's rank correlation between the subject abilities and item difficulties (Table 15.2).

Ranking Stability Experiments

Here we provide further details for the ranking stability experiments (§15). First, we filter from the 161 subjects that have development set scores to the 115 that also have test set scores.[16] In our simulation, we run 10 trials for every sample size; sample size begins at 100 and with steps of 100. In addition to these, we also run trials for sample sizes 25, 50, and 75. Since each sample can be no larger than half the dataset, we stop at half the dataset.

[16] The SQUAD organizers curate the test set subjects to avoid overfit, garbage, or duplicate submissions.

Discriminability: 2.1 **Difficulty**: 2.38 **Feasibility**: 0.995 **Mean Exact Match**: 0.00621 **Mean F1**: 0.546
Wikipedia Page: European Union Law **Question ID:** 57268f2bf1498d1400e8e3c4
Question: What reform was attempted following the Nice Treaty?
Official Answer: an attempt to reform the constitutional law of the European Union and make it more transparent
Context: Following the Nice Treaty, there was an attempt to reform the constitutional law of the European Union and make it more transparent; this would have also produced a single constitutional document. However, as a result of the referendum in France and the referendum in the Netherlands, the 2004 Treaty establishing a Constitution for Europe never came into force. Instead, the Lisbon Treaty was enacted. Its substance was very similar to the proposed constitutional treaty, but it was formally an amending treaty, and – though it significantly altered the existing treaties – it did not completely replace them.

Figure 15.11: This example shows that the answer span is likely too large, causing models to fail in both SQuAD's exact match and F1 metrics.

Discriminability: 8.01 **Difficulty**: -1.41 **Feasibility**: 0.939 **Mean Exact Match**: 0.64 **Mean F1**: 0.667
Wikipedia Page: Normas **Question ID:** 56de10b44396321400ee2595
Question: Who did the Normans team up with in Anatolia?
Official Answer: Turkish forces
Context: Some Normans joined Turkish forces to aid in the destruction of the Armenians vassal-states of Sassoun and Taron in far eastern Anatolia. Later, many took up service with the Armenian state further south in Cilicia and the Taurus Mountains. A Norman named Oursel led a force of "Franks" into the upper Euphrates valley in northern Syria. . . .

Figure 15.12: This highly discriminative question succeeds because there are many plausible answers. For example, although only "Turkish forces" is correct, some models answer "the Armenian state."

Development and Test Set Correlations

Table 15.3 uses a TODO, FIX ME!!!! model since we noticed that in comparison TODO, FIX ME!!!! overfit the data, yielding worse results. The correlations with the full data are all strong, but not the same. We conclude that—at least on SQuAD—IRT rankings are modestly more reliable than classical rankings.

Statistical Significance of Difference in Kendall Tau Coefficients

While Figure 15.4 shows a consistent difference in correlation between ranking methods, it is unclear whether this difference is statistically significant. We estimate the statistical significance of the difference through bootstrap sampling (**?**).

Since the null case is no difference in correlation coefficients, we seek a symmetric sampling distribution centered at zero that represents a realistic density function. Each ranking stability experiment[17] trial results in two lists of number pairs. The lists correspond to subject scores on two datasets;[18] each number pair is the subject's accuracy and IRT score. To create the bootstrap distribution, we (1) sample with replacement pairs from one list, (2) compute the correlation between the resampled ranking and unused ranking when using accuracy versus IRT score, and (3) compute and store the IRT correlation score minus the accuracy correlation score. We repeat this process 1000 times for each of the 10 trials in the original experiment and aggregate all the differences to build the bootstrap distribution. For each sample size we

[17] One experiment for development sample to development sample and one for development sample to test set.

[18] In the first experiment, development set samples; in the second, a development set sample and the full test set.

Figure 15.13: The feasibility parameter λ of our IRT model represents the probability that an example is unsolvable. For example, annotation error could lead to an example always being scored incorrectly—regardless of how good the model is. In SQuAD 2.0, $\lambda < .434$ in the 5% percentile, $\lambda < .698$ for the 7.5%, and $\lambda < .931$ in the 10% percentile.

Table 15.1: The linear model integrates a variety of features to determine which are most predictive of a subject responding correctly to an item.

Feature	Description
All	All the features
IRT	IRT values for difficulty, discriminability, feasibility, and ability
Item ID	The item's ID
Subject ID	The subject's ID
Question	Question words
Context	Context words
Stats	Question & context lengths; answerability, answer position & length; difficulty from **?**
Subject & Item ID	Item and Subject ID
Topics 1K	Topic weights of question words
Title	Wikipedia page title words
Baseline	No features, majority class baseline

compute the empirical P-Value on each trial which we show in box and whisker plots (Figure 15.14).

The IRT Statistical Test

The IRT test differs in two substantial ways from other tests: (1) it does not assume that items are equally informative and (2) it does assume that the informativeness of items is a function of the subject's skill θ_j. In the literature, this is closely connected to *reliability* (**?**) and each item provides information about the location of θ_j; as we accumulate more evidence for the location of θ_j the error of estimation decreases. It is a well known result in IRT that standard error of estimate (SEE) $\sigma(\hat{\theta}|\theta)$ varies with respect to the agent location parameter θ (**?**, p. 30) and is connected to the Fisher information

$$I_i(\theta) = \frac{(p_i')^2}{p_i(1-p_i)} \tag{15.4}$$

of each item. For a 2PL model, information

$$I_i(\theta) = \gamma^2 p_i(1-p_i) \tag{15.5}$$

is maximized when $p_i = (1-p_i)$. Since Fisher information is additive, the information of the evaluation set is maximal when items have a 50% chance of being responded to correctly. As derived by **?**, p. 102, the standard error of estimation

$$\text{SEE}(\theta) = \sqrt{\frac{1}{\sum_i I_i(\theta)}}. \tag{15.6}$$

is computed by accumulating the information gained from each item. Given two subjects X and Y, one can use the probability distribution of

Ability	TODO, FIX ME!!!!	TODO, FIX ME!!!!	TODO, FIX ME!!!!
TODO, FIX ME!!!!	1.00	0.947	0.895
TODO, FIX ME!!!!	0.947	1.00	0.907
TODO, FIX ME!!!!	0.895	0.907	1.00

Table 15.2: Table entries are Kendall's τ rank correlation of IRT subject ability between rows and columns. Generally, the models agree on the ranking with the TODO, FIX ME!!!! and TODO, FIX ME!!!! having the strongest correlation.

	EM_{dev}	EM_{test}	$\text{Ability}_{\text{dev}}$	$\text{Ability}_{\text{test}}$
EM_{dev}	1.00	0.953	0.954	0.931
EM_{test}	0.953	1.00	0.944	0.947
$\text{Ability}_{\text{dev}}$	0.954	0.944	1.00	0.950
$\text{Ability}_{\text{test}}$	0.931	0.947	0.950	1.00

Table 15.3: Entries are Kendall's rank correlation between rows and columns. Scores are SQUAD Exact Match (EM) and TODO, FIX ME!!!! ability.

score differences

$$N(\theta_Y - \theta_X, \text{SEE}(\theta_X)^2 + \text{SEE}(\theta_Y)^2) \tag{15.7}$$

to compute the probability that the difference in skill is greater than two standard errors which corresponds to an $\alpha \leq .05$ significance level.

Multidimensional IRT Clustering

While we achieve strong held-out accuracy with 10 dimensional IRT (TODO, FIX ME!!!!), we had limited success in interpreting parameters. We use TSNE[19] plots overlayed with features like item accuracy, the question's Wikipedia page, if the question was answerable, length of questions, and topic model weights. Of these, item accuracy and answerability showed the most obvious patterns (Figure 15.15).

We repeated this approach with the multi-task question answering shared task MRQA (**?**). However, instead of using 10 dimensions we use 6 to match the number of development set tasks in MRQA. Although questions in NarrativeQA standout (Figure 15.16), there is not a discernible pattern amongst the other tasks. We leave more sophisticated methods for making multidimensional IRT models interpretable to future work.

Reproducibility Checklist

Here we provide reproducibility details to complement our source code (https://irt.pedro.ai).

Software and Parameters

All IRT models are implemented in PyTorch (**?**) and Pyro (**?**). Linear models are trained with Vowpal Wabbit (**?**). The topic model that generates features for the linear model uses Mallet (**?**).

[19] We use openTSNE (**?**) with default parameters.

Figure 15.14: P-values of the rank correlation difference for each sample size and trial in Figure 15.4. The inherent noise in dev set sampling makes inferring significance difficult (left); test set driven results (right) are more significant.

Figure 15.15: In SQuAD, TSNE shows a relationship between mean exact match (item accuracy) and answerability with respect to multidimensional difficulty and discriminability.

The number of IRT model parameters is proportional to the number of subjects m and the number of items n. The TODO, FIX ME!!!! has one parameter per subject and one per item. The TODO, FIX ME!!!! has one parameter per subject and two per item. The TODO, FIX ME!!!! has one parameter per subject and three per item. The TODO, FIX ME!!!! has ten parameters per subject and thirty per item.

Hyperparameters

We did not invest significant effort in hyper-parameter tuning the IRT models and instead used the defaults in the py-irt software[20] provided by **?**. The TODO, FIX ME!!!! , TODO, FIX ME!!!! , and TODO, FIX ME!!!! models were trained for 1000 epochs with no early stopping conditions and a learning rate of 0.1 with ADAM (**?**). The TODO, FIX ME!!!! model was trained for 2500 epochs and used 10 dimensions.

In the linear model, we used a Hyperopt-based (**?**) tool provided by Vowpal Wabbit[21] for hyper parameter search. For each LM, the tool spent 20 iterations optimizing the learning rate, L2 regularization, and number of bits against the logistic loss function. The learning rate was searched from 0.001 to 10 with loguniform sampling, L2 regularization from $1e - 8$ to 1, and bits from 20 to 23 as categorical variables.

The topic model that generated features for the linear model used mallet, and we followed the recommendations of the software to set hyper parameters.[22] Specifically, we used an optimization interval of 10, removed stop words, trained for 1000 iterations, and used a document-topic threshold of 0.05. Each document was comprised of the Wikipedia page title and the question text.

Computational Resources

The majority of experiments were conducted on a single workstation with an Intel i7-7700K CPU, 47GB of RAM, and an Nvidia 1080Ti. The average runtime for the TODO, FIX ME!!!! model on CPU is 113 seconds with a standard deviation of 2.31 over 5 trials. The average runtime of the TODO, FIX ME!!!! model on GPU is 110 seconds with a standard deviation of 0.5 over 5 trials.

Since each ranking stability experiment required (§15) re-training an TODO, FIX ME!!!! model on each subset, we parallelized this experiment on a CPU cluster where each trial received two CPU cores and 16GB of RAM. In total, this included 520 trials which corresponds to twice that many trained IRT models since one model is trained on

[20] github.com/jplalor/py-irt

[21] github.com/VowpalWabbit/vowpal_wabbit

[22] mallet.cs.umass.edu/topics.php

each subset of the data.

Figure 15.16: In MRQA, TSNE shows a relationship between whether the task is NarrativeQA with respect to multidimensional difficulty and discriminability.

Part III

Question Answering Future

16 Human–Machine Collaboration

If existing datasets and game show appearances aren't enough to tell whether humans or computers are better at answering questions, what can we do? While there is an argument for focusing on natural data, modern language models are changing not just what is possible computationally but changing the language itself. Thus, we need to select examples specifically to challenge computers. These examples are called *adversarial examples*, and this chapter presents how to gather them and how they can reveal the strengths and weaknesses of computer question answering.

17
Question Answering in Science Fiction

From humans and dieties posing each other questions to humans and computers facing off in the Turing test, these face-offs have also become a staple of science fiction. This chapter reviews human–computer question answering in 2001, Star Trek, The Terminator, Blade Runner, Futurama, and novels and what these depictions reveal about both human conceptions of artificial intelligence and how it might shape future deployments of question answering.

18
What AI Dystopias You Should be Afraid of

Beyond the imagining of science fiction, what are the actual downsides to the widespread availability and deployment of 'intelligent' AI? This chapter talks about AI's ability to amplify existing negative *human* tendencies and how—while dangerous and worthy of mitigation—don't demand us to halt the development of modern AI.

19
Computer Game Shows of the Future

The year is 2025, and there's a new gameshow that not only showcases the most advanced AI available but also keeps the public informed about the limitations and struggles of current technology. This chapter outlines the ten seasons of the show and how it tracks the development of machine intelligence, leading to the Turing Test.

Bibliography

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daume III. Besting the quiz master: Crowdsourcing incremental classification games. 2012.

Anthony Cuthbertson. Robots can now read better than humans, putting millions of jobs at risk. *Newsweek*, 2018. URL https://www.newsweek.com/robots-can-now-read-better-humans-putting-millions-jobs-risk-781393.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 2019. URL https://www.aclweb.org/anthology/N19-1246.

R. Dwan. *As Long as They're Laughing: Groucho Marx and You Bet Your Life*. Midnight Marquee Press, 2000. ISBN 9781887664363. URL https://books.google.ch/books?id=hO0KAQAAMAAJ.

Stephen Eltinge. Quizbowl lexicon, 2013. URL http://www.pace-nsc.org/pace-quizbowl-lexicon/.

Manaal Faruqui and Dipanjan Das. Identifying well-formed natural language questions. 2018.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3), 2010.

Flight Standards Service. *Aviation Instrutor's Handbook*, volume FAA-H-8083-9A. Federal Aviation Administration, Department of Transportation, 2008.

Morris Freedman. The fall of Charlie Van Doren. *The Virginia Quarterly Review*, 73(1):157–165, 1997. ISSN 0042675X. URL http://www.jstor.org/stable/26439004.

Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. 2019.

Mark E Glickman and Albyn C Jones. Rating the chess rating system. *Chance*, 12, 1999. URL http://scholar.google.de/scholar.bib?q=info:r4leZxFtllwJ:scholar.google.com/&output=citation&scisig=AAGBfm0AAAAAUm2XckUml2ofzJwFf4A_kI-bLVVMwfdI&scisf=4&hl=en&scfhb=1.

Merv Griffin. *Merv : making the good life last*. Thorndike Press, Waterville, Me, 2003. ISBN 0786253533.

Alvin Grissom II, He He, Jordan Boyd-Graber, and John Morgan. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. 2014.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C. Lipton. AmazonQA: A review-based question answering task. 2019.

Thomas M. Haladyna. *Developing and Validating Multiple-choice Test Items*. Lawrence Erlbaum Associates, 2004.

Bob Harris. *Prisoner of Trebekistan : a decade in Jeopardy*. Crown Publishers, New York, 2006. ISBN 0307339564.

Robert X. D. Hawkins, Andreas Stuhlmüller, Judith Degen, and Noah D. Goodman. Why do you ask? Good questions provoke informative answers. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, 2015.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. 2016.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. Quora question pairs, 2017. URL https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Ken Jennings. *Brainiac: adventures in the curious, competitive, compulsive world of trivia buffs*. Villard, 2006. ISBN 9781400064458.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. ISSN 0885-6125. doi: 10.1023/A:1007665907178.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. 2017. URL https://www.aclweb.org/anthology/P17-1147.

Divyansh Kaushik and Zachary C. Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. 2018.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. 2019. URL https://tomkwiat.users.x20web.corp.google.com/papers/natural-questions/main-1455-kwiatkowski.pdf.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. 2019.

Roger Levy. Integrating surprisal and uncertain-input models in online sentence comprehension: Formal techniques and empirical results. 2011.

Roger P. Levy, Florencia Reali, and Thomas L. Griffiths. Modeling the effects of memory on human online sentence processing with particle filters. 2008.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.eacl-main.86.

Zachary C. Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *Queue*, 17(1), February 2019. ISSN 1542-7730. doi: 10.1145/3317287.3328534. URL https://doi.org/10.1145/3317287.3328534.

Elaine Low. Jeopardy! goat ken jennings on trebek, trash talk and why he's hanging up his buzzer. *Variety*, January 2020.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. 2019.

Kenny Malone. How uncle Jamie broke Jeopardy. *Planet Money*, (912), May 2019.

David Marchese. In conversation: Alex trebek the jeopardy! icon on retirement, his legacy, and why knowledge matters. *Vulture*, November 2018.

Adam Najberg. Alibaba AI model tops humans in reading comprehension, 2018. URL https://www.alizila.com/alibaba-ai-model-tops-humans-in-reading-comprehension/.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

Pedro Rodriguez, Shi Feng, Mohit Iyyer, He He, and Jordan Boyd-Graber. Quizbowl: The case for incremental question answering. *CoRR*, abs/1904.04792, 2019. URL http://arxiv.org/abs/1904.04792.

Lisa Rogak. *Who is Alex Trebek? : a biography*. Thomas Dunne Books, an imprint of St. Martin's Publishing Group, New York, NY, 2020. ISBN 1250773660.

Idan Szpektor and Gideon Dror. From query to question in one click: Suggesting synthetic questions to searchers. 2013.

Gerald Tesauro, David C. Gondek, Jonathan Lencher, James Fan, and John M. Prager. 21, 2013.

Ellen M. Voorhees. *Evaluating Question Answering System Performance*, pages 409–430. Springer Netherlands, Dordrecht, 2008. ISBN 978-1-4020-4746-6. doi: 10.1007/978-1-4020-4746-6_13. URL https://doi.org/10.1007/978-1-4020-4746-6_13.

Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. 2000.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. Trick me if you can: Human-in-the-loop generation of adversarial question answering examples. *Transactions of the Association of Computational Linguistics*, 10, 2019a.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China, November 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1534. URL https://aclanthology.org/D19-1534.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. 2019.