Quizbowl: The Case for Incremental Question Answering

Pedro Rodriguez pedro@cs.umd.edu
Shi Feng shifeng@cs.umd.edu
Department of Computer Science
University of Maryland at College Park
College Park, MD

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA

He He
Department of Computer Science, Courant Institute
New York University
New York, NY

Jordan Boyd-Graber
Department of Computer Science
University of Maryland at College Park
College Park, MD

Editor:

Abstract

Quizbowl is a scholastic trivia competition that tests human knowledge and intelligence; additionally, it supports diverse research in question answering (QA). A Quizbowl question consists of multiple sentences whose clues are arranged by difficulty (from obscure to obvious) and uniquely identify a well-known entity such as those found on Wikipedia. Since players can answer the question at any time, an elite player (human or machine) demonstrates its superiority by answering correctly given as few clues as possible. We make two key contributions to machine learning research through Quizbowl: (1) collecting and curating a large factoid QA dataset and an accompanying gameplay dataset, and (2) developing a computational approach to playing Quizbowl that involves determining both *what* to answer and *when* to answer. Our Quizbowl system has defeated some of the best trivia players in the world over a multi-year series of exhibition matches. Throughout this paper, we show that collaborations with the vibrant Quizbowl community have contributed to the high quality of our dataset, led to new research directions, and doubled as an exciting way to engage the public with research in machine learning and natural language processing.

Keywords: Factoid Question Answering, Sequential Decision-Making, Natural Language Processing

1. Introduction

For over fifty years factoid question answering competition through trivia games have been a fun and popular way for humans to intellectually compete against each other. From

At its premiere, the librettist of this opera portrayed a character who asks for a glass of wine with his dying wish. That character in this opera is instructed to ring some bells to summon his love. At its beginning, a man who claims to have killed a serpent has a padlock put on his mouth because of his lying. The plot of this opera concerns a series of tests that Tamino must undergo to rescue Tamina from Sorastro. For 10 points, name this Wolfgang Mozart opera titled for an enchanted woodwind instrument.
Answer: The Magic Flute

Figure 1: A typical Quizbowl question with clues that are initially difficult, but become progressively easier until a giveaway at the end of the question. Players answer as soon as they know the answer so as a result the earlier they answer the more knowledgeable they are. For example, answering after the first sentence indicates the player recognizes the librettist (Emanual Schikaneder) and knows that they played Papageno in *The Magic Flute* (die Zauberflöte). Answering at the end of the question only requires surface knowledge of Mozart's opera works.

television early shows such as *Dotto*, to more recent television shows such as *Jeopardy!* and *Who Wants to be a Millionaire*, to popular board games like *Trivial Pursuit*, to scholastic competition like Quizbowl, trivia games have long held human interest. Trivia is motivating because it encourages us to gain new knowledge, recall relevant past knowledge, and reason over that knowledge. These skills—knowledge representation and reasoning—are known to be AI-complete (Yampolskiy, 2013); the intense research interest in factoid question answering (FQA) is thus unsurprising. We argue that the research community would benefit tremendously from the lessons that the trivia community—in particular the large and vibrant Quizbowl community—has learned about competition format and question writing. We show that collaboration with this community has resulted in a better format for evaluating machine progress in FQA.

In Quizbowl, questions are posed *incrementally*—word by word—and players must *interrupt* the question when they know the answer (Figure 1). Thus, it rewards players who can answer with less information than their opponents. Unlike other question answering tasks, players must simultaneously think about what is the most likely answer and at every word decide whether it is better to answer or wait for more information. To succeed, players and machines alike must be proficient at answering questions, maintaining accurate estimates of their confidence, and factoring in the abilities of their opponents. The combination of these skills makes Quizbowl challenging for machine learning algorithms.

Quizbowl was originally crafted by a dedicated and skilled community for scholastic trivia competition (Section 2). It is the product of a symbiosis between tournament organizers and competitors to create fun, interesting competitions that reliably select the most knowledgeable players. This annual cycle of question writing is the principal contributor to the diversity and size of our dataset (Section 3). We refer to this dataset as the QANTA dataset since (in our opinion) **Q**uestion **A**nswering is **N**ot a **T**rivial **A**ctivity.[1]

1. Dataset available at `http://datasets.qanta.org`.

Playing Quizbowl requires deciding *what* to answer with (Section 5) and *when* to answer (Section 6). Our final contribution is a simple framework that combines independent systems for each of these sub-tasks (Section 4). Despite its simplicity, our implementation of this framework is competitive with the best players. However, our implementation (or even) framework is not a final solution to Quizbowl: the goal of this article is to provide a foundation for future research in this engaging domain.

Section 8 showcases the many ways we have used Quizbowl as a platform for simultaneously advancing machine learning and natural language processing research (NLP) and educating the public about the limits of machine learning and NLP. The primary way we accomplish this is through live events—usually co-located with national high school tournaments—where humans and machines compete against each other. From a research perspective this provides a way to evaluate, in realistic settings, the progress against humans. In Sections 9 and 10 we discuss ongoing and future research such as human-in-the-loop adversarial question writing and humans playing cooperatively with machines.

2. Why Quizbowl?

When discussing machine learning and trivia, the elephant in the room is always IBM's tour-de-force match (Ferrucci et al., 2010) against Ken Jennings and Brad Rutter on Jeopardy! Rather than ignore the obvious comparisons, we take this on directly and use the well-known Jeopardy! context—which we gratefully acknowledge as making our own work possible—as a point of comparison to argue why Quizbowl as a question answering framework is a better differentiator of skill between participants, be they human or machine (Sections 2.1 and 2.2).[2]

Much of what makes Quizbowl a better differentiator is a consequence of its evolution over the course of its fifty year history.[3] Many of the challenges the NLP community faces in collecting good question answering datasets—such as avoiding predictive yet useless patterns in data or incorporating multi-hop reasoning—were first encountered by trivia aficionados. We distill these lessons, describe the craft of question writing that makes Quizbowl a compelling question answering task (Section 2.3), and enumerate some NLP challenges required to truly solve Quizbowl (Section 2.4). We conclude by framing Quizbowl as a hybrid task between question answering and sequential decision-making (Section 2.5).

2.1 What is a Buzzer Race?

The scapegoat for every Jeopardy! loser and the foundation of every Jeopardy! winner is the **buzzer** (Harris, 2006). A buzzer is a small handheld device that players press to signal that they can correctly respond to a clue. The fundamental difference between Jeopardy! and Quizbowl—and what makes Quizbowl more suitable for research—is how clues are revealed and how players use the buzzer.

2. Boyd-Graber et al. (2012) introduced Quizbowl as a factoid question answering task, Iyyer et al. (2015) further developed algorithms for answering questions, and He et al. (2016) improved live play. This article drops all artificial limitations, significantly expands the dataset, and evaluates models in offline, online, and live environments.
3. US service members during World War II (Taylor et al., 2012) played a game similar to Quizbowl originally devised by Ron Reid. Following the war several popular television shows such as University Challenge, Delco Hi-Q, College Bowl, and *It's Academic* popularized the format.

Jeopardy! is a television show and uses the buzzer to introduce uncertainty, randomness, and thus excitement for the viewer at home. In Jeopardy!, players can only use the buzzer when the moderator has finished reading the question.[4] If players attempt to use the buzzer before the question is finished, they are locked out and prevented from answering the question for a fraction of a second (an eternity in the fast-paced game of Jeopardy!).

This presents a significant advantage to Watson in its match against two opponents with their feeble human thumbs and reflexes, as Jeopardy! uses the buzzer to determine who among those who know the answer *has the fastest reflexes*. While Watson gets an electronic signal when it was allowed to buzz, the two humans watch for a light next to the Jeopardy! game board to know when to buzz. Thus, Watson—an electronic buzzing machine—snags the first choice of questions, while the two humans fight over the scraps. In Jeopardy! reflexes are almost as important as knowledge. Next we show how the structure of Quizbowl questions, and how the implementation of buzzing in Quizbowl makes reflexes only sparingly important.

2.2 Pyramidality and Buzzers

In contrast, Quizbowl is a game honed by trivia enthusiasts which uses buzzers as a tool to determine *who knows the most about a subject*. This is possible because the questions are *interruptable*. Unlike Jeopardy!, players can interrupt the questions when they know the answer (recall questions are multi-sentence in Quizbowl). This would make for bad television (people like to play along at home and cannot when they cannot hear the whole question), but makes for a better trivia game.

This alone is insufficient however; if an easy clue appears early in the question then knowing hard clues later in the question is irrelevant. Questions that can be answered with only a fraction of their input are a bad foundation for research (Sugawara et al., 2018). Quizbowl addresses this problem by structuring questions *pyramidally*. In pyramidal questions, clues are incorporated so that harder, more obscure information comes first in the question, and easier, more obvious information comes at the end of the question. As a result, when a player answers before their opponents it implies that they are more knowledgeable than their opponents.

This also makes Quizbowl an attractive research domain. The giveaways are often easy for computers too: they are prominent on Wikipedia pages and have appeared in many questions. Thus, it is easy for computers to answer most questions *at some point*: Quizbowl is not an impossibly difficult problem. The challenge then becomes to answer the questions *earlier*, using more obscure information and higher-order reasoning.

Humans who play Quizbowl have the same yearning; they can answer most of the questions, but they want to collect enough facts to buzz in just a little earlier. They keep practicing, playing questions and going to tournaments to slowly build up enough skill and knowledge to improve. Quizbowl is engineered for this to be a rewarding experience.

The same striving can motivate researchers: it does not take much to buzz in a word earlier. As small incremental improvements accumulate, we can have more robust, comprehensive question answering systems. And because Quizbowl has a consistent evaluation framework, it is easy to see whose hard work has paid off.

4. In Jeopardy! terminology is reversed so that a moderator reads clues termed *answers* to which players must supply the correct *question*. To avoid confusion, we follow standard terminology.

Thus, the form of Quizbowl questions—the product of decades of refining how to measure the processing and retrieval of information of humans—represents an effective way to measure machine question answering. We next describe the cultural norms of question writing in the Quizbowl community that contribute to making it a challenging task for humans and machines alike.

2.3 The Craft of Question Writing

The goal of Quizbowl is to reward "real" knowledge. This goal is the product of a long history that has resulted in community norms that have evolved the competition into a thriving, carefully designed trivia ecosystem. By adopting these conventions developed over decades of trial and error, machine learning can adopt these best practices for question answering evaluation.

Instead of catering to popularity, Quizbowl is guided by its community. Rather than engineering exciting upsets (*a la* Jeopardy!), Quizbowl ensures that whoever knows more about a topic will be the one to answer questions on that topic. This goal has engineered not just the pyramidal question structure but other aspects of question crafting.

Every year question writers in the community focus on creating high quality questions that are novel and pyramidal. New questions are written every year to discourage the roughly 10,000 students who compete in Quizbowl from memorizing questions as opposed to learning deeply about a topic.[5] Regional competition questions are written by participants; championship competition questions are written by professionals hired by either the Academic Competition Federation (ACF), National Academic Quiz Tournaments (NAQT), or the Partnership for Academic Competition Excellence (PACE). As a whole, question writers at all levels have more knowledge and experience in crafting good questions than crowd-workers who typically generate question answering datasets.

To help maintain the quality and integrity of competition, the community has developed a set of question writing guidelines which are well aligned with rewarding generalizable machine learning models: avoiding ambiguity, ensuring correctness, and allowing for fair comparisons between teams (Lujan and Teitler; Vinokurov; Maddipoti).

The first step in writing a new question is deciding what it should be about. In Quizbowl, all answers must be uniquely identifiable named entities such as—but not limited to—people, places, events, and literary works. These answers are similar to those from other factoid question answering datasets such as SimpleQuestions (Bordes et al., 2015), SearchQA (Dunn et al., 2017), and TriviaQA (Joshi et al., 2017). This applies to both answers of questions and guesses provided by players during a game.[6] Fortunately, most Quizbowl answers strings are exact or fuzzy matches to a Wikipedia page (Section 3.4.1). As a result, one way to frame the machine learning problem—although certainly not the only way—is classification over entities represented by Wikipedia pages. This makes the challenge of eliminating ambiguity and ensuring correctness of answers in Quizbowl tractable and accurate.

Fair comparison between competitors is a product of pyramidality. For example, the first sentence of Figure 1—also known as the lead in—while obscure, uniquely identifies a single

5. Participation has roughly doubled every year since 2008 and is accelerating.
6. When a player provides an ambiguous—but plausible—answer moderators often *prompt* the player to elaborate, such as clarifying which Francis they mean: the philosopher, artist, or pope.

opera; an expert can recognize the opera. Questions that are misleading early on are scorned and derided in online discussions after a tournament as "neg bait" or a "hose";[7] Thus, writers ensure that all clues are uniquely identifying. By the end of the question, someone who knew of The Magic Flute would be able to answer.

The entirety of questions are carefully crafted, not just the lead-in. Middle clues should reward knowledge but not be too easy: if a clue appears too often or is too prominent in the subject's Wikipedia page, it's considered a "stock" clue and should be avoided. By the end of a question, the final giveaway clue should be direct and well-known enough so that if the player has heard of the answer they will correctly answer the question. From the machine learning perspective, the avoidance of over-used keywords in early clues discourages humans and machines from answering solely based on superficial pattern matching.

This is the product of a complicated and nuanced social dynamic in the Quizbowl community. Top teams and novice teams play on the same questions; the questions should be fun and fair for all. The pyramidal structure ensures that top teams use their deep knowledge and quick thinking to buzz on the very first clues, but novice teams are entertained and learning until they get to an accessible clue. Just about everyone answers all questions (it is considered a failure of the question writer if the question "goes dead" without an answer).

Quizbowl is not just used to test knowledge; it also helps discover new information and as a result diversifies questions ("oh, I didn't know the connection between the band the Monkees and correction fluid!").[8] While most players will not recognize the first clue (otherwise the question wouldn't be pyramidal), it should be interesting and connect to things the player would care about. For example, in our Magic Flute question, we learn that the librettist appeared in the premiere, a neat bit of trivia that we can tuck away once we learn the answer. The culture of incorporating new facts and novel combinations of facts aligns well with improving generalization in question answering datasets.

Altogether, these cultural norms makes Quizbowl a compelling way to compare the effectiveness of question answering models to human players. Next we shift towards linking some challenges in Quizbowl to some in NLP.

2.4 Quizbowl for Natural Language Processing Research

Figure 1 showcases several features and NLP challenges common to many Quizbowl questions. Clues are initially obscure and become progressively more recognizable (*pyramidality*); however, each sentence uniquely identifies the answer. The player with the most knowledge can answer the question earlier and "win" the question.

For a computer to answer the question early requires solving difficult natural language inferences like coreference resolution and entity linking. In our example the computer must recognize that "the librettist" refers to Schikaneder. To accomplish this it needs to know about the link (coreference resolution), and recognize that it is relevant to the question (entity linking). Accurate coreference is known to improve question answering systems (Stuckardt, 2003).

7. "Negging" refers to interrupting a question with a wrong answer; while wrong answers do happen, it is accepted that a response with a valid chain of reasoning behind it should be accepted. If a question admits multiple viable answers, it is considered poorly written.
8. Bette Nesmith Graham, the mother of Monkees band member Michael Nesmith, invented correction fluid in 1956.

Many FQA systems, including ours for Quizbowl, use Wikipedia as an additional knowledge source. A system using Wikipedia should parse this link from the sentence "Schikaneder was the librettist, composer, and principal singer" from Schikaneder's Wikipedia page. However, extracting this link requires accurate coreference and anaphora resolution, which is itself a difficult problem in NLP (Ng, 2017) that Guha et al. (2015) argue is particularly challenging in Quizbowl datasets.

Even with a good knowledge base, entity linking is not trivial (Shen et al., 2015). Traditional challenges in entity linking include disambiguation between similar entities (e.g., Michael Jordan the professor versus the basketball player), and different surface forms (New York City versus NYC). In Quizbowl there are additional challenges since referring expressions tend to be longer. Using our example from Figure 1, take the character Tamino: while he is mentioned by name, it is not until after he has been referred to multiple times obliquely ("a man who claims to have killed a serpent"). Understanding the question required a combination of coreference resolution and entity linking.

Inference like in the clue about "the librettist" is often called *higher-order reasoning*. Questions that require only a single lookup in a knowledge base or a single IR query are uninteresting and mean that only a miniscule fraction of all possible questions could be answered. Interest in multi-hop question answering led to the creation WikiHop through templates (Welbl et al., 2018) and HotPotQA through crowdsourcing (Yang et al., 2018). The first sentences in Quizbowl questions are the most difficult clues because they often incorporate surprising, quirky relationships that require skill and reasoning to recognize and disentangle.

Finally, even the final clue (called a "giveaway" because it's so easy for humans) could pose issues for a computer. Connecting "enchanted woodwind instrument" to The Magic Flute requires solving wordplay. While not all questions have all of these features, these features are typical of Quizbowl questions and showcase the richness of the problem.

2.5 Quizbowl as a Machine Learning Task

Players, human or machine, show their breadth and depth of knowledge by not only answering correctly but answering *before* their opponent. It is therefore crucial to answer Quizbowl question with the least amount of information; but how does one determine when enough information is known? Thus, Quizbowl is a hybrid of question answering and sequential decision-making.

Specifically, Quizbowl is a word-by-word question answering task where players—human or machine—must decide at each time step (word) whether to answer, and if so with what answer. Determining what to answer with is a factoid question answering task which we frame as high dimensional multi-class classification. Deciding when to answer is a binary sequential decision-making task: at each time step the agent must wait for more information or buzz in then provide a guess.

The primary challenge in framing Quizbowl as a question answering task lies in defining the form of answers and a mechanism for determining if a specific candidate answer is correct. Early work in QA (Kupiec, 1993) defined the answer set as noun phrases extracted from online encyclopedias. We use a similar concept, but we instead use the *titles* of all Wikipedia articles as our closed answer set. Although this includes any of the nearly six million pages

in English Wikipedia, in practice the number of classes represented in the training data is closer to 25,000.

An alternative framing for question answer tasks is "machine reading" popularized by SQuAD (Rajpurkar et al., 2016). The model sees a context paragraph and a question; it highlights a span of text in the context paragraph that answers the question. Unlike classification, there is no canonical set of answers to compare against; instead, SQuAD's evaluation compares the overlap of words in the candidate answer and the ground truth answer.[9] We focus on building classification-based systems that do not use context paragraphs, but in Section 9.1 discuss—as future work—how to incorporate context as in TriviaQA (Joshi et al., 2017).

We frame Quizbowl as classification and evaluate with accuracy-based metrics for several reasons. First, although this is in the technical sense closed domain factoid question answering, many and diverse entities of interest are represented (Section 3.2.3). Second, Quizbowl answer strings are typically near-exact matches to Wikipedia titles after Quizbowl specific syntax—such as instructions to the moderator—are removed. As a result, most answers in our dataset do have corresponding Wikipedia articles; only a minority of answers do not (Section 3.4.1).[10] This makes determining a canonical answer and comparing it to candidate answers trivial since there is zero ambiguity about which entity is being referred to. Thus, since models answer from a closed answer set we evaluate with accuracy-based metrics.

The sequential decision-making task—buzzing—is related to cost-sensitive learning. Cost sensitive learning factors in the cost of discovering feature values as well as the cost of errors. Zubek and Dietterich (2002) and Chai et al. (2004) study cost sensitive learning in the medical domain where a doctor must diagnose a patient, but each medical test has a cost. In Quizbowl the cost is directly related to difference between the expected utility of the agent seeing more words, and its opponent seeing more words. For example, if they agent is very certain that its speculative answer is correct, then there is little to be gained by waiting and giving its opponent an opportunity to find a crucial clue.

We evaluate the performance of our systems through a combination of standalone comparisons (Section 7.1) and simulated Quizbowl matches (Section 7.3). For standalone evaluation we incrementally feed systems new words and record their responses. Using these responses we generate accuracy based statistics as a function of position in the question. While standalone evaluations are useful for developing systems, the best way to compare systems and humans is with evaluations that mimic Quizbowl tournaments.

To play simulated games, we assume that a match consists of a sequence of questions called a packet. Each question is revealed incrementally to the players (machine or human) until one decides to buzz in with an answer.[11] If the answer is correct the player gains ten points. Otherwise, they lose five points and their opponent has an opportunity to answer.[12] Once all the questions in the packet have been played, the player with the most points wins. In our full evaluations we have systems play matches against each other round-robin style,

9. Two metrics are used. The first, Exact Match, checks if the highlighted span matches ground truth exactly. The second, F1, computes the F1 score between the bag-of-words candidate and ground truth answers.

10. For example, although most questions refer to novels or their authors—which mostly do have articles, some questions ask about characters—which often do not; for example, "Bean" in Ender's Game.

11. Ties—which are infrequent—are broken randomly.

12. Their opponent can answer provided the full text, and without penalty even if their answer is wrong.

play simulated matches against humans using the gameplay dataset, and play live exhibition matches against teams of accomplished trivia players.

A recurring theme is our mutually beneficial collaboration with the Quizbowl community: host outreach exhibitions (Section 8), annotate data (Section 9.2), play with and against our systems (Section 10.4), and collect the qanta dataset. This community created this rigorous format for question answering over decades and continues to help understand and measure the question answering abilities of machines.

3. Qanta Dataset

This section describes the two parts of the qanta dataset. The first component of our dataset is a large bank of over 100,000 human authored trivia questions from Quizbowl tournaments dating back to 1997. The second part of our dataset is a set of 3.9 million records of humans playing Quizbowl online where each record corresponds to a human playing one question. The first part of this section shows how we collected each of these datasets (Section 3.1).

In the second half of this section we analyze both datasets. First, we show that the dataset of questions is large compared to other fqa tasks, especially in number of tokens. Next, we show that the questions are syntactically diverse (Section 3.2.1); this is done through an analysis based on probabilistic context free grammars. We conclude the question analysis by showing the diversity in topics (Section 3.2.2) and answers (Section 3.2.3). Following this, we show more details of the gameplay dataset to show that it is also a large and diverse in the types of players—such as aggressive versus passive or risky versus safe (Section 3.3).

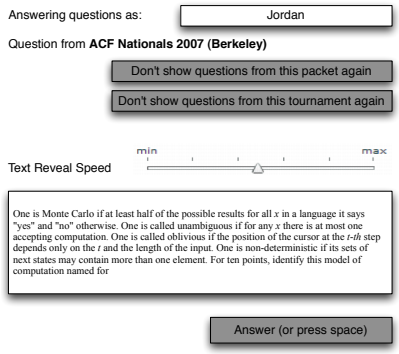
Lastly, we describe the preprocessing applied to the dataset. This primarily involves associating answer strings with Wikipedia titles (Section 3.4), and dividing the data into partitions for training, developing, and testing systems (Section 3.4.2).

3.1 Dataset Sources

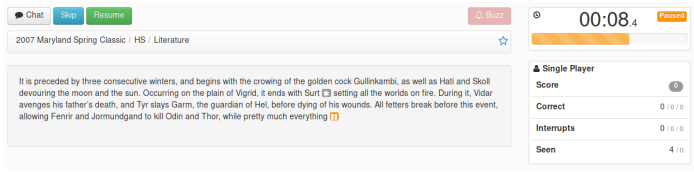
Every year about 10,000 new Quizbowl questions are written for invitational tournaments, regional qualifiers, and national championships. New questions must be written every year since students often practice with questions from prior competitions. Tournament organizers want to discourage students from memorizing clues rather than focusing on knowledge understanding. We build the qanta dataset by using questions from community Quizbowl sites.[13] Combined, this is over 100,000 questions from tournaments held between 1997 and 2018, encompassing local high school competitions to college championships.

With these questions as training data we can build a qa system, but playing Quizbowl requires more than answering questions; systems play against opponents, so they must decide when to answer. If a system waits too long for more information to make answering correctly more likely then its opponent may answer first; similarly, if a system accepts a higher degree of uncertainty in its answer, it is more likely to be penalized for being wrong. We address this challenge by collecting data on how humans play—such as if and when they answer specific questions.

13. We collect questions from `http://quizdb.org` and `http://protobowl.com`. Both sites obtain questions from tournaments.



(a) Our 2012 interface was the first way to play Quizbowl online.



(b) The Quizbowl interface used to collect the gameplay dataset from. It makes various improvements in interface design and also enables realtime play against other humans.

Figure 2: Our interface and a popular modern interface for playing Quizbowl online. Both interfaces reveal questions word-by-word until a player interrupts the system, and makes a guess.

To collect human play data on Quizbowl we built the first interface for playing Quizbowl online, and now partner with its direct successor (`http://protobowl.com`) to collect the data at a much larger scale. On both platforms users play questions from prior tournaments. In our original 2012 interface, shown in Figure 2a, words in the question are revealed one-by-one until the player attempts to answer the question.[14]

The data are collected using the interface in Figure 2b, which improves on our original website by adding realtime play against other players (instead of leaderboard only), and an improved user interface. Every time a question is played we record what word the player buzzed on, their answer, and whether their answer was correct. At time of publication we have collected 3.9 million records from over ten thousand users. We call this the **gameplay dataset**, and further describe how we use it in Section 3.3.

3.2 Number and Diversity of Quizbowl Questions

We begin our description of the QANTA dataset with its size. Table 1 compares QA datasets whose questions were written by humans. For reference we include the number of QA pairs for prior versions of the QANTA dataset. We adopt this notation going forward as we will update the dataset on an annual basis as new questions are written for Quizbowl competitions. We compare the size of these datasets through the number of question-answer pairs, number of sentences, and number of tokens in question text.

Because the questions are pyramidal (Section 2.3), the questions tend to be longer than other datasets. Each Quizbowl question has four to six sentences. Because nearly every sentence has enough information for a player to answer the question, each QANTA instance can be broken into many sentence-answer pairs (This helps our model in Section 5). Aside from SearchQA (Dunn et al., 2017), a Jeopardy! based dataset, the QANTA dataset is the largest factoid QA dataset publicly available in number of question-answer pairs (120K),

14. On the first day 7000 questions were played, and by the end of the two week experiment 43000 questions were played by 461 users.

Dataset	QA Pairs	Tokens
SimpleQuestions (Bordes et al., 2015)	100K	.614M
TriviaQA (Joshi et al., 2017)	95K	1.21M
SQuAD 1.0 (Rajpurkar et al., 2016)	100K	.988M
SearchQA (Jeopardy!) (Dunn et al., 2017)	216K	4.08M
QANTA 2012 (Boyd-Graber et al., 2012)	46K / 8K	1.07M
QANTA 2014 (Iyyer et al., 2014)	158K / 31K	3.93M
QANTA 2018 **(This Work)**	**650K / 120K**	**11.4M**

Table 1: A size comparison shows that the QANTA dataset is larger than most question answering datasets in number of QA pairs (120K). However, for most Quizbowl instances each sentence in a question can considered a QA pair so the true size of the dataset is closer to 440K QA pairs. In Section 5 using sentence level QA pairs for training greatly improves model accuracy. The QANTA dataset is also significantly larger than all other QA datasets in of number of tokens in question texts. Numbers for QANTA 2012 and 2013 only include publicly available data.

and is over three times as large as SearchQA in number of sentence-answer pairs (650K). In addition to having more examples, questions in Quizbowl are longer overall and have longer sentences (Figure 3). While the QANTA dataset is compelling in sheer size—which will increase every year—it is also crucial that machine learning datasets are diverse.

3.2.1 Syntactic Diversity

Quizbowl is a syntactically diverse dataset with dense coreferences (Guha et al., 2015) and complex structure. This section argues that Quizbowl is more syntactically diverse than other factoid question answering datasets. Syntactic diversity is desirable because different, yet equivalent, framings of the same question should be handled equally well by trained models, but are often not (Iyyer et al., 2018). Additionally, diversity discourages models from building (invalid) correlations between specific syntactic structures (artifacts) and specific answers. Throughout we assume that a dataset's syntactic diversity correlates with the number of unique constituency parses.

First we we generate constituency parses for each question with Stanford CoreNLP (Manning et al., 2014; Bauer, 2014).[15] Since we are interested in syntactic diversity—diversity based on structure of language, not in choice of vocabulary—we exclude terminals; additionally if we did not exclude terminals then every unique question would trivially be a unique parse. A reasonable next step would be to compare the number of unique parses.

However, a drawback of using only the number of unique parses is that it does not consider coarse versus granular syntactic diversity. Consider the two sentences in Figure 4: under this scheme they are counted as distinct parses despite being closely related since the top portions of their parse trees are identical. At a course level the parses are the same while at the granular level they are different. Rather than balancing the importance of coarse versus granular diversity, we compare diversity across different depths of parse trees. Specifically,

15. We use the shift-reduce parser in version 3.9.1 of Stanford CoreNLP.

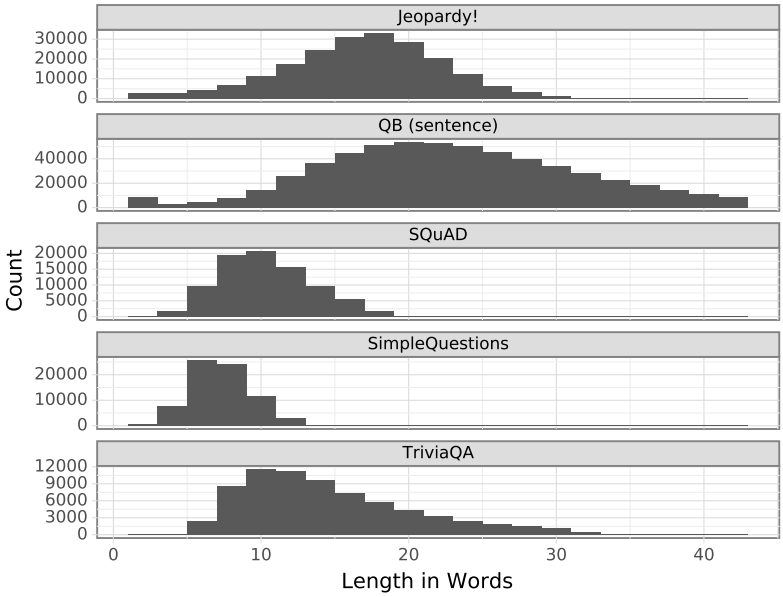


Figure 3: Size of question answering datasets. Questions in the QANTA dataset have longer sentences than any other dataset. The instances from SimpleQuestions, SQuAD, and TriviaQA are comparatively short which makes it less likely that they are as diverse of Quizbowl or Jeopardy!. For each dataset we compare the lengths of questions rather than paired context paragraphs; to avoid the histogram being overly skewed we remove the top 5% of examples by length from each dataset.

we compute the number of unique parses when only considering non-terminals up to every depth d in the tree from one to ten. In the example from Figure 4 the parses from trees of maximum depths one, two, and three are identical; it is not until depth four in the tree that their structures differ. In our analysis, parse trees from depths one, two, and three count as one unique parse, and parse trees with depth four or more count as two unique parses. If a dataset is diverse, it should show diversity across a wide range of maximum tree depths.

With this framework in place we compute the number of unique parses and average number of unique parses per sentence versus truncation depth for each dataset. Figure 5 shows that both competitive trivia datasets—Quizbowl and Jeopardy!—are more diverse on average, and Quizbowl has significantly more unique constituency parses. These results are consistent across any truncation depth. Combined this shows that Quizbowl is syntactically diverse, and that trivia datasets (Quizbowl, TriviaQA, and Jeopardy!) tend to have more diverse examples.

To further validate these findings, we compute the average entropy of an example's syntactic structure. Finding this is equivalent to computing the probabilistic context free grammar (PCFG) of each dataset, and then computing the average entropy of a question

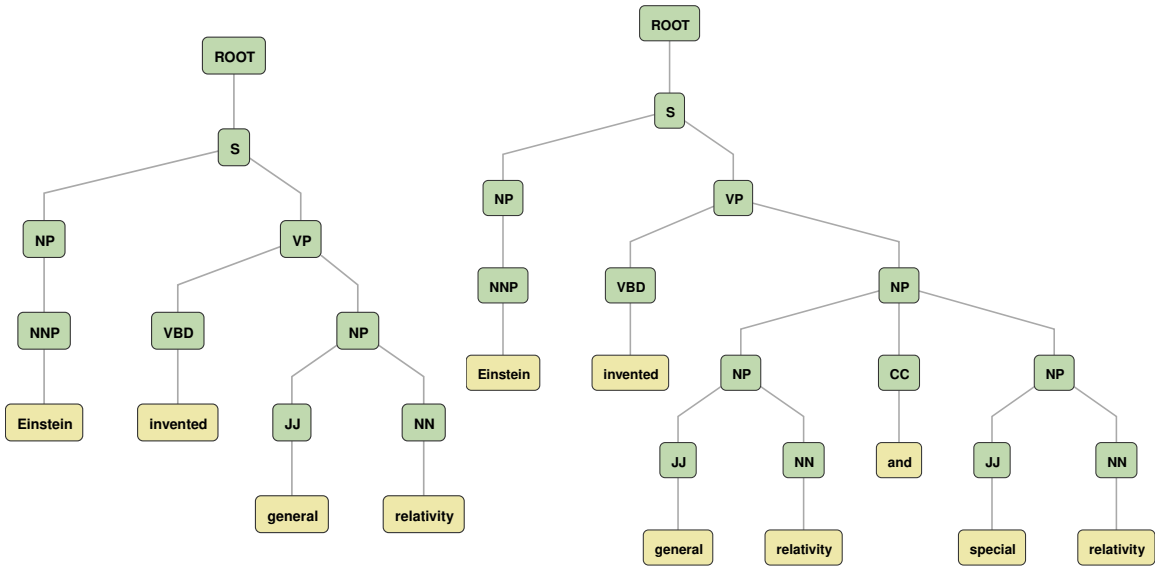


Figure 4: Here we show the constituency parse trees for two similar sentences. The syntactic structure of the sentences is identical until depth $d = 4$. In our analysis we compute the number of unique parses as a function of d, ranging from one to ten, on several QA datasets.

Dataset	PCFG Entropy
TriviaQA	5.64
QANTA 2018 (This Work)	5.48
SearchQA	5.41
SQuAD 1.0	5.30
SimpleQuestions	4.53

Table 2

under this PCFG.[16] With these definitions the entropy is

$$\mathbb{H}\left[P(a \to b)\right] = -\sum_{i=1}^{n} P(a_i \to b_i) \log P(a_i \to b_i)$$

The entropy of the PCFG for each dataset is in Table 2. From this comparison we reach conclusions similar to those from Figure 5; the QANTA dataset has more diverse and sophisticated examples than datasets crowdsourced online. This is not surprising as questions from the QANTA datasets, TriviaQA, and SearchQA (Jeopardy!) are often written by domain experts, and sometimes even professional writers.

16. To derive the PCFG we use the constituency parses to compute the conditional probability of a transition $a \to b$, the probability of a transition a, and finally the unconditional probability of a transition $a \to b$. Each of these probabilities are defined as $P(a \to b \,|\, a) = \frac{Count(a \to b)}{Count(a)}$, $P(a) = \frac{Count(a)}{\sum_{a' \in A} Count(a')}$, and $P(a \to b) = P(a \to b \,|\, a) P(a)$

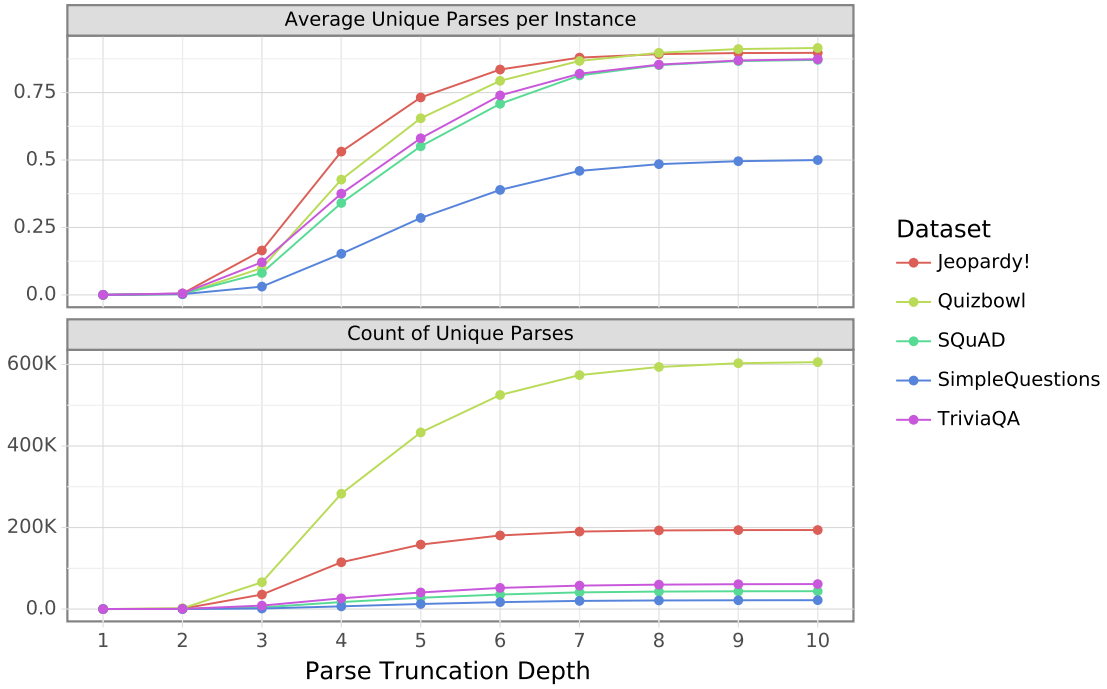


Figure 5: The syntactic diversity of QA datasets measured through their constituency parses. More diverse datasets should have more total unique parse trees (bottom plot), and unique parses per instance (top plot). To avoid minor variations deeper in parse trees from impacting results too heavily we plot counts and averages for a range of constituency tree truncation depths. At each truncation depth, we prune any part of the tree which is deeper than the specified depth before computing uniqueness of parses. Consider $S \rightarrow$ NP VP. Under truncation depth one this becomes S while under truncation depth two it remains the same.

3.2.2 Topical Diversity

Topical diversity is a goal shared between researchers creating datasets and organizers of Quizbowl tournaments. Quizbowl organizers ensure topical diversity in tournaments by defining desired distributions over categories and sub-categories and then writing to match. As a side effect, every Quizbowl question has an assigned category and sub-category. Figure 6 shows the aggregate category distribution over areas such as history, literature, science, and fine arts. For the largest categories we show the sub-category distribution in Figure 7. Taken together, Quizbowl is a topically diverse dataset across broad categories and finer grained sub-categories. This emphasizes that to do well players and systems need to have both breadth and depth of knowledge.

3.2.3 Answer Diversity

Quizbowl questions are also diverse in the kinds of entities that appear as answers (in total about 25K entities are present in the training data). A dataset which is topically diverse, but only asks about people is not ideal. Using the Wikidata knowledge graph we obtain the

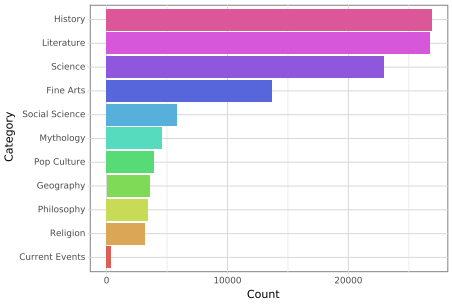


Figure 6: Questions in Quizbowl cover most if not all academic topics taught in school, and focus on history, literature, science, the fine arts, and social sciences.

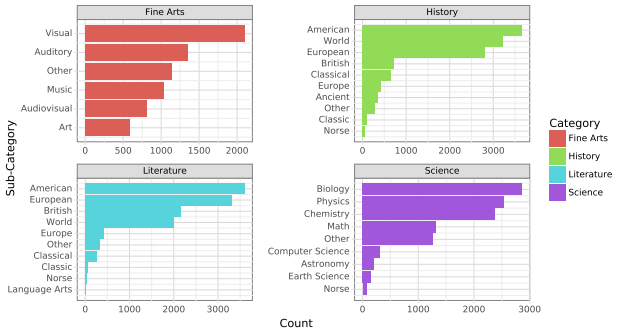


Figure 7: Even within a single category, questions cover a range of topics. Topically the dataset is biased towards American and European topics in literature and history.

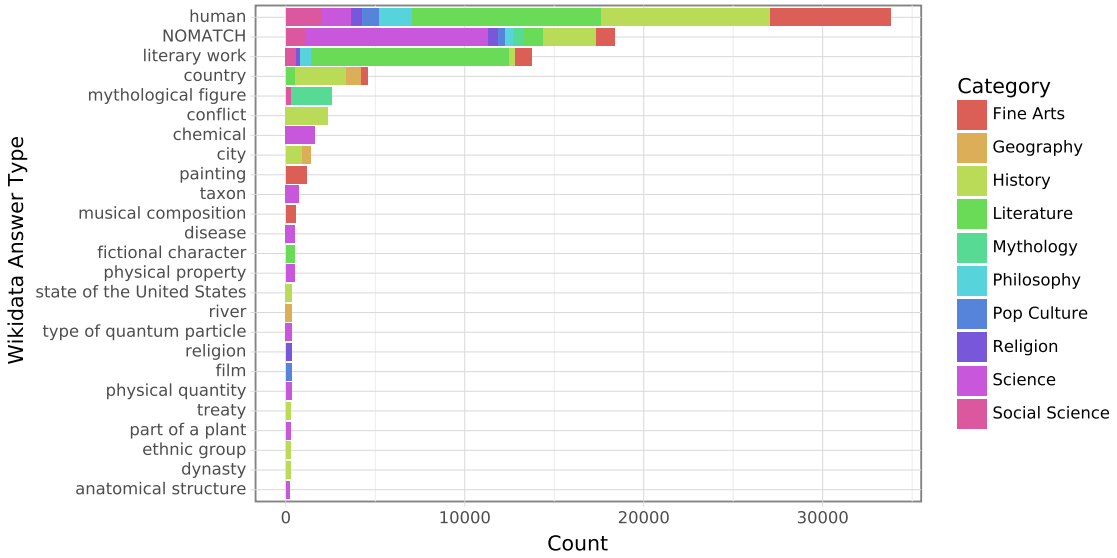


Figure 8: Distribution of answer types according to wikidata.org "instance of" attribute. Most answer types are topically diverse. We exclude answers which did not have an "instance of" attribute and condense several similar attributes.

type of each answer and plot frequencies in Figure 8.[17] Most questions ask about people (human), but with a broad diversity among other types. The special category "NOMATCH" characterizes answers which did not have a matched type.

Taken as a whole, these three analyses show that Quizbowl is syntactically, topically, and answer-wise diverse. To Quizbowl aficionados this is unsurprising; the primary educational

17. The knowledge graph is from `http://wikidata.org`, and we use the "instance of" as an answer's type. We collapse similar types into larger categories as well.

goal of Quizbowl is to encourage students to improve their mastery over wide ranges of knowledge. We now turn to details about the gameplay dataset.

3.3 Gameplay Dataset

Using only the question dataset is enough for training question answering systems, but not for training agents to play Quizbowl. Although standard QA systems may know *what* to guess, they would not know *when* to guess. The gameplay dataset allows us to characterize how humans play Quizbowl, and enable agents to play simulated games against humans so that they can learn effective buzzing strategies.

With this data we can simulate gameplay of machines versus human players. For every word in the question we query the machine for its action—wait or buzz—and check what the human would have done at on same word. By doing this repeatedly during training machines can learn optimal buzzing strategies based on their own uncertainty, the questions, and their opponent's prior behavior (He et al., 2016).

The QANTA gameplay dataset contains 3.9 million records of $10,576$ humans answering one of $128,988$ Quizbowl questions. The records are collected using the interface in Figure 2b. Each record in this dataset tells us how a player answered a question—both the guess and the buzzing position. In Table 3 the user correctly guessed "Atlanta" at word forty-seven. If an agent played against this player they would need to answer correctly before word forty-seven to win.

Unfortunately, various biases exist in the raw gameplay data; limitations of the Quizbowl platform can lead to over- or under-estimation of player ability. First, since the question pool is finite (although it is growing), a player might see a question multiple times, and they might get the answer correct immediately using memorization instead of knowledge. These records can lead us to overestimate player ability, so we only keep the first record for each question-player pair based on the timestamp. Secondly, some records do not come from actual Quizbowl players, but from random browsing on the site; to avoid underestimating player ability, we remove users who answered fewer than twenty questions.

A good quizbowler needs to be both accurate and quick, so we measure player ability by both average accuracy and buzzing position (percentage of the question revealed), shown as the two axes in Figure 9. An average player buzzes with 65% of the question shown, and achieves about 60% accuracy.

The gameplay dataset is valuable for evaluating and developing our computer system. Directly using the game records, we run our system against human players in simulated games (Section 7.3). Based on the inferred player ability, we create a metric for system comparison (Section 7.1.2). We also train our model to decide when to buzz using this dataset (Section 6).

3.4 Preprocessing

This section outlines the process for producing our final published datasets available at `http://datasets.qanta.org`. We primarily address how we match answers to Wikipedia pages (Section 3.4.1), and our process for assigning questions to training, development, and test folds.

Date	Thu Oct 29 2015 08:55:37 GMT-0400 (EDT)
UID	9e7f7dde8fdac32b18ed3a09d058fe85d1798fe7
QID	5476992dea23cca90550b622
Position	47
Guess	atlanta
Result	True
Question text	This Arcadian wounded a creature sent to punish Oeneus for improperly worshipping Artemis and killed the centaurs Rhaecus and Hylaeus...

Table 3: An entry from the gameplay dataset where the player correctly guesses "Atlanta" at word 47. The entry QID matches with the PROTO_ID field in the question dataset where additional information is stored such as the source tournament and year.

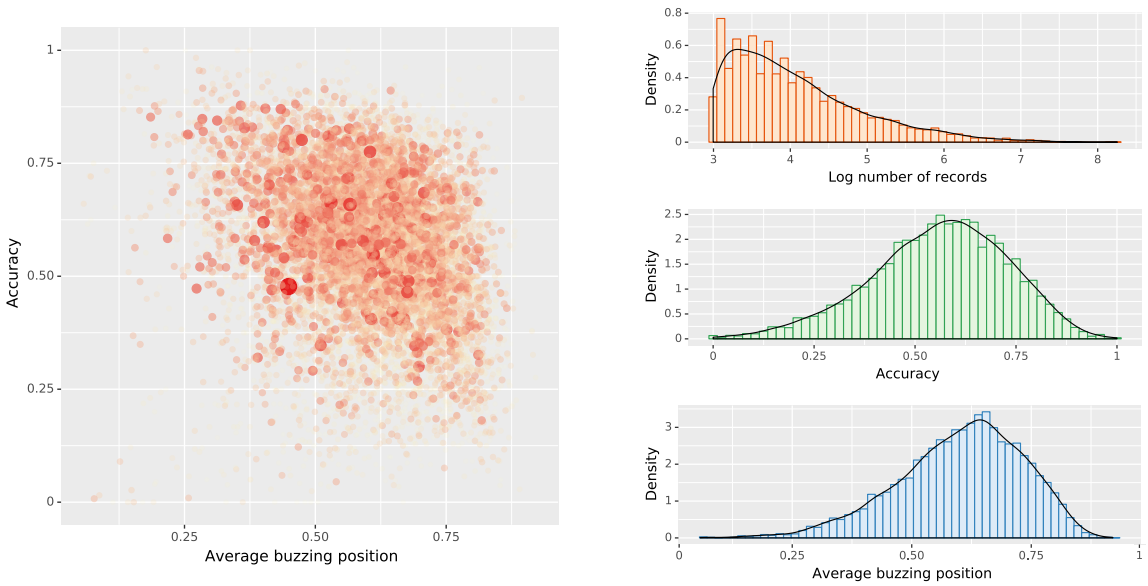


Figure 9: Left: each protobowl user is represented by a dot, positioned by average accuracy and buzzing position; size and color indicate the number of questions answered by each user. Right: distribution of number of questions answered, accuracy, and buzzing position of all users. An average player buzzes with 65% of the question shown, and achieves about 60% accuracy.

3.4.1 MATCHING QUIZBOWL ANSWERS TO WIKIPEDIA PAGES

Quizbowl questions nearly always ask about things which can be unambiguously identified by a distinct Wikipedia page.[18] However, answers in the dataset often contain Quizbowl specific notation that make exact matching more difficult such as instructions for acceptable alternative answers (Second Vatican Council or Vatican II), answers which are unambiguous

18. We use the english Wikipedia dump created on 4/18/2018, and preprocess it with `https://github.com/attardi/wikiextractor`.

given the question but not just the answer (Thomas Hutchinson the governor versus the scholar), and inconsequential differences (pluralization, foreign language names, capitalization, and explanatory notes).

To address these challenges we design a matching process that pairs answer strings to Wikipedia titles in two steps. Table 4 shows a random sample of matches generated with this process. In the first step we generate mutations of the original answer such as removing braces or all content in brackets. In the second step we attempt to exactly match the mutated string to a Wikipedia page. Most of the examples from Table 4 match exactly by removing braces and ignoring capitalization. If we still cannot find a match then we test mutations of Wikipedia titles such as removing parenthetical words ("governor", "physics", and "mechanics"). After this we create a list of questions that still do not have matches, and use this as our starting point for manual annotation.

To annotate these answer strings we use a combination of crowdsourcing in the Quizbowl community, and annotation by us, the authors. In previous versions of this dataset we pooled answer strings from all folds, and annotated the answer strings from most to least frequent up to a lower bound frequency. While this reduces the annotation burden, it indirectly leaks significant information about the test questions to the training questions. The existence of an annotation pair in the test set makes it easier for similar answer strings to be matched in the training data (the training set answers are not exhaustively matched). In short, a test set match can trigger additional training data for that answer to be found. The mere existence of training data for this class significantly increases the likelihood of systems answering correctly.

We correct this error by separating the answer string pool for training and test questions. Although this results in more annotation work, it avoids information leakage. While reviewing our annotation procedure we noticed another source of bias. Recall that we do not exhaustively annotate the training data. In our initial annotation we did not fully annotate the test data, and by doing so introduced a bias towards easier-to-annotate questions in the test set. To eliminate this bias—and make it as similar to playing a Quizbowl tournament as possible—we annotated every question in the test set.[19] In total we paired 119,093 out of 132,849 with Wikipedia titles. We describe the matching process further in Appendix A.4.

3.4.2 Dataset Folds and Annual Updates

We divide qanta dataset questions into training, development, and test folds based on the competitiveness and year of the source tournament. Since championship tournaments typically have the highest quality questions we use questions from championship tournaments 2015 and onward as development and test sets.[20] All other questions are used as the training set.

This folding scheme maximizes the quality of Quizbowl questions in the development and test sets while the temporal separation reduces the likelihood of data leaks. Separating the training, development, and test sets temporally was also used to build SearchQA from Jeopardy! questions (Dunn et al., 2017). This also forms the basis for our plan to annually

19. Specifically, we either pair each test set answer strings with a Wikipedia title or mark it as not having a corresponding Wikipedia title.

20. We use questions from acf Regionals, acf Nationals, acf Fall, pace nsc, and nasat from 2015 onward for development and test sets.

Original Quizbowl Answer	Matched Wikipedia Page
Nora Helmer	A_Doll's_House
{Gauss}'s law for the electric field	**No Mapping Found**
Thomas Hutchinson	Thomas_Hutchinson_(governor)
linearity	Linearity
{caldera}s	Caldera
William Holman {Hunt}	William_Holman_Hunt
{plasma}s	Plasma_(physics)
{Second Vatican Council} [or {Vatican II}]	Second_Vatican_Council
{Jainism}	Jainism
{Electronegativity}	Electronegativity
Hubert Selby, Jr.	Hubert_Selby_Jr.
(The) Entry of Christ into Brussels (accept equivalents due to translation)	Christ's_Entry_Into_Brussels_in_1889
Depictions of Speech [accept equivalents]	**No Mapping Found**
stress	Stress_(mechanics)

Table 4: A random sample of QB answer strings and their matched Wikipedia pages. Answer mappings are easy to obtain accurately since most failures in exact matching are due to Quizbowl specific syntax that can be accounted for by rule based matching. Combined with manual annotation to find common non-exact matches, this process succeeds on 119,093 of 132,849.

update the test data to prevent overfitting to the test set: new questions become the test set, test set questions become the development set, and development questions are pushed into the training data. This will aid in avoiding the common problem of overconfidence of a model's generalization (Patel et al., 2008), and has the free benefit of incorporating questions about current events.

Not all questions in the dataset have associated gameplay data—instances of humans playing the question—so not all questions can be used simultaneously to train systems on *what* and *when* to answer. We standardize the use of our dataset by sub-dividing questions in each fold—train, dev, and test—depending on if they have associated gameplay data. Table 5 shows the divisions of each fold; each train, dev, or test fold is assigned to be used for either determining *what* to answer (guessing) or *when* to answer (buzzing). Questions in "guess" folds are used for developing question answering systems as in Section 5. Questions in the "buzz" folds are used for developing agents that decide when to answer as in Section 6. See Appendix A.3 for more details on our fold assignment process.

Fold	Total
train + guess	96221
train + buzz	16706
dev + guess	1055
dev + buzz	1161
test + guess	2151
test + buzz	1953
unassigned	13602
All	132849

Table 5: We assign each question in our dataset to either the train, development, or test fold. Questions in the development and test folds come from national championship tournaments which typically have the highest quality questions. The development and test folds are temporally separated from the train and development folds to avoid leakage. Questions in each fold are assigned a "guess" or "buzz" association depending on if they have gameplay data; questions without that can only be used for training systems what to guess, not when to guess. Unassigned refers to questions for which we could not map their answer strings to Wikipedia titles or there did not exist an appropriate page to match to. Details of the fold assignment process are in Appendix A.3.

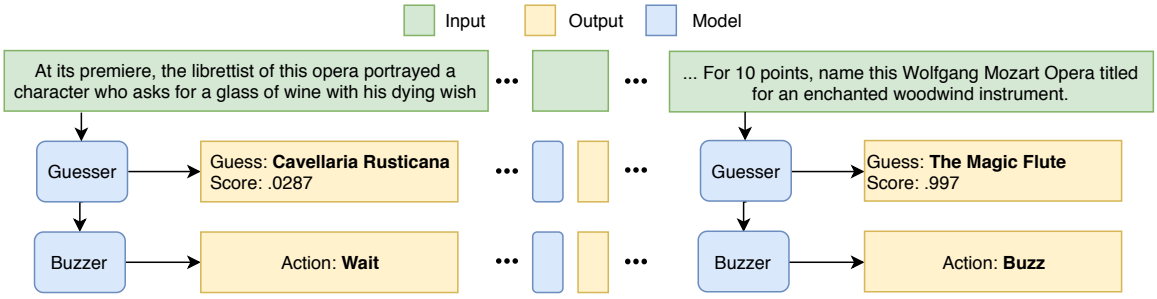


Figure 10: The QANTA framework for playing Quiz Bowl with semi-independent guesser and buzzer models. After each word in the input is revealed the guesser model outputs its best guesses. The buzzer uses these in combination with positional and gameplay features to decide whether to take the buzz or wait action. The guesser is trained as a question answering system that provides guesses given the input text seen so far. Buzzers use the guesser's score, positional features, and human gameplay data to predict the whether the buzz or wait action provides the highest reward.

4. Deciding When and What to Answer

One could imagine many machine learning systems that could play Quizbowl: an end-to-end reinforcement learning system or a heavily pipelined system that determines category, answer type, answer, and decides when to buzz. Without making any value judgement on the *right* answer, our approach divides the task into two subsystems: **guessing** and **buzzing**. This approach mirrors IBM Watson's[21] two system design (Tesauro et al., 2013). The first system answers questions, and the second decides when to buzz.

We refer to the question answering task of deciding *what* to answer as guessing, and the task of deciding *when* to answer as buzzing (Figure 10). In our systems, guessing is based solely on question text, and buzzing is based on the best guess, its score, and the game state (e.g., how many words have been read, changes in guess scores, and more).

Concretely, at test time the guessing model outputs its best guess and score at each time step (word), while the buzzing model uses the score and game state to determine whether to buzz or wait. The guessing model is trained as a question answering model with the question dataset (Section 5). The buzzing model is trained by using the predictions of trained guessing models and gameplay data (Section 6).

Machines playing Quizbowl by guessing and buzzing semi-independently is convenient from an engineering perspective: it simplifies model training and is easier to debug. More importantly, it allows us and subsequent researchers to focus on a sub-task of their choosing or the task as a whole. If you are interested in only question answering, focus on the guesser. If you are interested in multiagent cooperation or confidence estimation, focus on the buzzer.

Following discussion of our guessing (Section 5) and buzzing (Section 6) systems we describe our evaluations and results in Section 7.1. Section 8 summarizes the outcomes of our live, in-person, exhibition matches against some of the best trivia players in the world.[22]

5. Guessing Quizbowl Answers

Guessing answers to questions is a factoid question answering task and the first part of our systems that play Quizbowl (Figure 10). We frame the question answering sub-task in Quizbowl as high dimensional multi-class classification over Wikipedia page entities (i.e., answers are entities defined by distinct Wikipedia pages). Here we describe several ways to guess answers: information retrieval methods (Section 5.1), linear models (Section 5.2), and neural network models (Section 5.3). In addition to these—and motivated by data sparsity for many answers—we incorporate Wikipedia text as additional training data (Section 5.4).

5.1 Explicit Pattern Matching with Information Retrieval

The first model family we consider are traditional information retrieval (IR) methods. Although early clues avoid keyword usage, giveaways often have many such as "Wolfgang Mozart" and "wood wind instrument". TF-IDF (Rajaraman and Ullman, 2011) based IR are well suited to matching these keywords and are a strong baseline (Section 7.1).

21. In Watson the second system also determines wagers on Daily Doubles, wagers on Final Jeopardy, and chooses the next question (e.g., history for $500)
22. The systems in these exhibitions are largely prior iterations of those we describe in this work.

To frame this as search problem in IR we treat guessing as document retrieval. For each answer $A_i \in \mathcal{A}_{train}$ in the Quizbowl training data we create a document D_i that represents the answer. Each document contains a text field that indexes the concatenation of all training questions for its answer.[23] At test time questions are queries and search for the highest scoring document D_i and return the corresponding A_i as the answer. We implement this with Elastic Search (which uses Apache Lucene) and use Okapi BM25 for computing match scores (Robertson and Walker, 1994). This implementation of BM25 computes the score of each document as

$$\sum_i IDF(q_i) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{fieldLen}{avgFieldLen})} \tag{1}$$

$$IDF(q_i) = \ln(1 + \frac{docCount - f(q_i) + .5}{f(q_i) + .5}) \tag{2}$$

where q_i are words in the question (query), D is a training document, $f(q_i, D)$ represents how many times q_i occurs in document D, and $f(q_i)$ is the number of documents that contain q_i. b and k_1 are hyper parameters.

As Equations 1 and 2 make explicit, this model is firmly based on deterministic, surface level pattern matching. It is therefore effective when these patterns are predictive of the answer. However, this reliance on pattern matching often is not enough to answer early in questions. For example, in the first sentence from Figure 1 the author intentionally avoids keywords ("a character who asks for a glass of wine with his dying wish"). Purely IR methods, while effective, are limited since they rely on keywords and are deterministic. In contrast statistical machine learning methods can learn representations that better fit Quizbowl questions. Thus, we move on to machine learning methods that have a chance at addressing some of these shortcomings.

5.2 Trainable Pattern Matching with Linear Models

The first trainable methods we consider are linear models with n-gram based features. Conceptually these models apply surface level pattern matching, but rather than have a matching function explicitly defined like in IR we leave the model to determine the best weights for each word. Since this is a high dimensional multi-class classification problem (approximately 25,000 distinct answers in training set), a typical one-versus-all approach is computationally expensive. We instead use a logarithmic time algorithm for one-versus-all classification (Agarwal et al., 2014; Daumé et al., 2017). Input features are a combination of n-grams and skip-grams.[24] For example, the feature vectors for the phrase "at its premiere" with bigram features and skip-grams would have ones in the positions representing "at", "its", "premier", "at its", "its premiere", and "at <skip> premier".

This model is limited since it only considers linear relationships between n-gram terms and that n-grams are based on bag of words. We now move to neural models that use more sophisticated forms of composition than bag of words.

23. In our hyper parameter sweep we consider document per answer (best results), one document per training example, different values for BM25 coefficients, and the default Lucene practical scoring function.
24. The order of words to use is treated as a hyper parameter with its best value determined by grid search jointly with learning parameters.

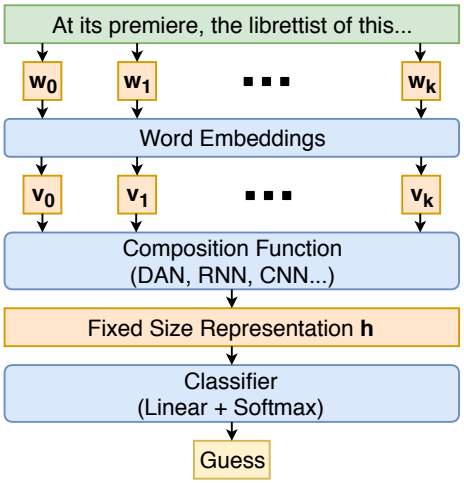


Figure 11: All our neural models feed their input to an embedding function, then a composition function, and finally a classification function. The primary variation across our models is the choice of composition function used to compute a fixed size representation from its variable length input.

5.3 Neural Network Models

The final family of methods we consider for Quizbowl question answering are neural methods. We describe the shared components of our neural models such as general architectures and training details and then compare their composition functions.

Figure 11 shows the three-part architecture of our neural models. First, an embedding layer to map individual words to numerical representations. Second, a composition function to combine sequences of words into a fixed size representation. And third, a classification function to compute the most likely answer given the fixed size representation.

The embedding function takes as input question q_i and its associated sequence of k_i input tokens $[w_1^{(i)}, \ldots, w_k^{(i)}]$. These are converted into a sequence of word embeddings $\boldsymbol{v}^{(i)} = [v_1^{(i)}, \ldots, v_k^{(i)}]$ which input to the composition function. We detail our choice of embedding initializations in Section 5.3.4.

First we consider unordered composition with deep averaging networks (Iyyer et al., 2015). This is the most similar to the IR and linear model since it is based on bag of words features. Next, we consider sequence based composition functions such as recurrent (Elman, 1990), long short-term (Hochreiter and Schmidhuber, 1997; Palangi et al., 2016, LSTM), and gated recurrent (Cho et al., 2014, GRU) networks. We compare unidirectional and bidirectional versions of these networks (Schuster and Paliwal, 1997). We leave comparing against other sequence models such as attention based models (Vaswani et al., 2017) and large pre-trained language models (Peters et al., 2018; Devlin et al., 2018) to future work.

The final component of the model uses the output of the composition function to compute the most likely answer with a classification function. The first part of the classification function is a linear layer which projects the question representation h to the dimension of the training answer set $\mathcal{A}_{train}$. The second part is a softmax function over the output of

the linear layer so that the network output is a categorical distribution over answers. The classification function outputs the answer with highest probability. In the remainder of this section we describe our use of deep unordered and ordered compositions functions, details of our neural architecture, and training details such as sentence level versus paragraph level training.

5.3.1 Unordered Composition with Deep Averaging Networks

dans are a simple, effective, and efficient method for Quizbowl question answering. Despite their complete disregard for word order, they have achieved competitive results compared to more sophisticated models on classification tasks such as sentiment analysis (Iyyer et al., 2015). Our first neural model uses a deep unordered composition function via a deep averaging network (dan). Although there are many cases where word order and syntax matter, many questions are answerable using only key phrases. For example, predicting the mostly likely answer to the bag of words "inventor, relativity, special, general" is easy; they are strongly associated with Albert Einstein.

The dan computes an initial hidden state $\mathbf{h_0}$ for question q_i with k_i ords as the average of their embeddings:

$$\mathbf{h_0} = \frac{1}{k} \sum_{i \in \{1,\dots,k_i\}} \mathbf{v_i} \tag{3}$$

It computes the output representation $\mathbf{h}$ by passing $\mathbf{h_0}$ through n feed-forward layers using the recurrence $\mathbf{h_i} = f(\mathbf{W^i} \cdot \mathbf{h_{i-1}} + \mathbf{b_i})$ where $\mathbf{W^i}$ and $\mathbf{b_i}$ are parameters of the model. The function f can be any element-wise function. Unsurprisingly using non-linearities for f—specifically variants of rectified linear units—achieves the best performance. Section 7.1 shows that although dans are not the most accurate model, they offer an attractive trade off between accuracy and computation cost.

5.3.2 Ordered Composition with Recurrent Neural Networks

Next we consider order-aware composition functions such as recurrent neural networks. dans are explicitly order-unaware which limits their theoretical expressiveness. Although this can be mitigated with higher order n-grams, they are still incapable of capturing long range dependencies. In contrast, order-aware models—rnns, lstms, and grus—can model long range dependencies in supervised tasks (Linzen et al., 2016).

In our rnn-based models, we input each word k_i and use the output states as the hidden representation. In bidirectional models we concatenate forward and backward hidden states. Our order-aware model's composition function computes its hidden state

$$\mathbf{h} = GRU(\mathbf{v_{1:k_i}}). \tag{4}$$

with a bidirectional gru (Cho et al., 2014). With our models defined we next describe our training procedure.

5.3.3 Using Sentences as Training Examples

Using full questions as examples leads models to focus on late, easy clues instead of early, difficult clues which makes answering early challenging. Additionally, a model using only the

long input of full questions may not be able to handle short single sentence inputs correctly. Instead of using questions as training examples we use each sentence in a question as a training example. This forces the model to extract more signal from each training example rather than only focus on indicative keywords from giveaway clues at the end of questions. The disadvantage of this approach is that models would not be able to learn to resolve co-references across sentences during training; fortunately many sentences have self-contained clues.

We empirically validate that the first factors are more important by comparing question training, sentence training, and variable length training.[25] Although the end-of-question accuracy in the question training scheme improves, the accuracy near the start approaches zero. Answering early in questions is far more important than minor gains at the end so we use sentence training in all experiments.

5.3.4 Training and Architecture Details

For each model we use 300-dimensional word embeddings initialized with GloVe for words in the vocabulary and randomly initialized embeddings otherwise.[26] In training we use dropout (Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015), adam (Kingma and Ba, 2015), early stopping, and learning rate annealing. We use elu in all non-linearities following fully connected feed-forward layers.

We optimize hyper parameters by running each setting three times with different random seeds and record the parameter settings corresponding to the top development set score. Next, we then run these parameters an five (additional) times to. Section 7.1 reports test set results aggregated over these five trials.

5.4 Wikipedia as Additional Training Data

A significant fraction of answers have very few or zero training examples; this predictably decreases accuracy on such answers. Nearly half of the questions in the dataset have only one training example. These answers are often infrequently asked about topics such as the Dark matter halo or relevant to current events such as the ETA (separatist group)[27] Figure 12 shows the distribution of training examples per class in Quizbowl with a similar phenomena observed in TriviaQA (Appendix B). Section 7.2.2 empirically shows that accuracy is significantly lower for these answers.

Although Wikipedia text is of a different distribution than questions, using it as additional training data is better than too little training data for specific answers. For example, there are zero training questions about Rocket, but a test question refers to Konstantin Tsiolkovksy's development of rocket theory.[28] However, this fact is mentioned on the Wikipedia page

25. We create k training examples from a question comprised of k sentences. Each example includes the text from the start position up to and including sentence k.
26. Randomly initialized embeddings use a normal distribution with mean zero and standard deviation one. We also compared initialization with Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Joulin et al., 2016).
27. Although the ETA was founded in 1959, the official disbanding of its political structure in 2018 likely prompted a question to be written about it. As our training data does not include 2018 its unsurprising that there are zero training examples.
28. Question 105837

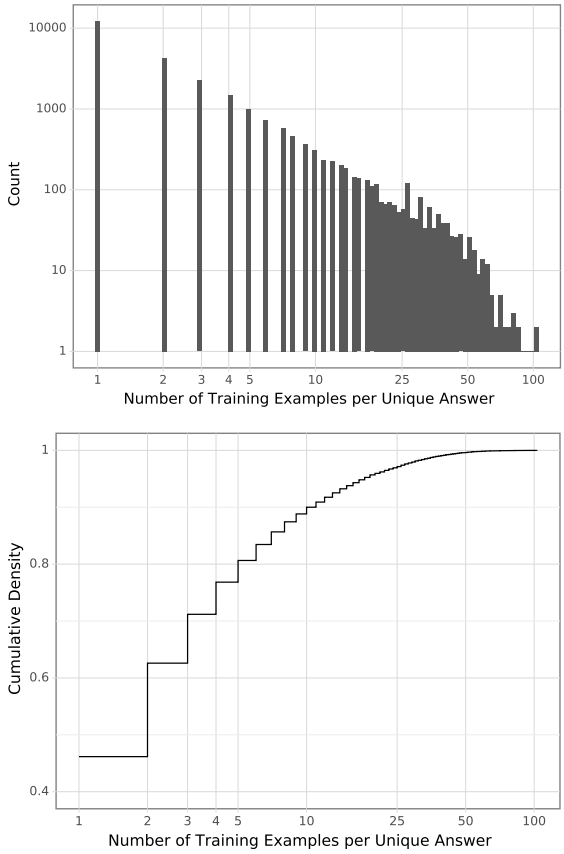


Figure 12: The distribution of training examples per unique answer (left) is heavily skewed. The most frequent answer occurs about 100 times (Japan). Nearly half of the questions in the dataset have one training example and just over sixty percent have either one or two training examples. We show in Section 7.2.2 that this scarcity of training examples per unique answer severely degrades performance.

on rockets so with Wikipedia as training data answering this question is plausible. Each Quizbowl answer is paired with a Wikipedia page so we use the text of that page as additional training data.[29] Experiments in Table 6 show that this simple method is effective, but does not work equally well across models. This section shows how we incorporate Wikipedia as training data in each model.

5.4.1 Wikipedia Text in ir Models

We incorporate Wikipedia text into our ir models by indexing it as an additional text field. The indexed text is the concatenation of every question with that answer. To incorporate Wikipedia text, we index an additional text field whose content is the text of the Wikipedia page. When the ir model is asked a question (issued a search query) it sums the textual

29. Yoshida (2018) investigates other methods for incorporating Wikipedia text during model training.

match scores of the Quizbowl and Wikipedia text and returns the answer with the highest score.

5.4.2 Wikipedia Text in Machine Learning Models

In our machine learning models we incorporate Wikipedia text as training examples. Much as IR models "learn" by indexing text, machine learning models "learn" from examples so we incorporate Wikipedia text as additional training examples. In our work we sentence tokenize the first paragraph of Wikipedia pages and use each sentence as a training example; the label of each sentence is the Wikipedia page it is from. For example, the first sentence from rocket's Wikipedia page "A rocket is a missile, spacecraft, aircraft, or other vehicle that obtains thrust from a rocket engine" would be a training example with the label "Rocket".

At first this method did not work since it violates a fundamental property in Quizbowl questions: the answer to the question is never mentioned in the text of the question. At test time we anecdotally observed that even if our model originally had a correct answer to a question, inserting the answer text caused our models to change their answer. To avoid this, we tokenize the Wikipedia title and remove all references to those tokens from the Wikipedia text. Using the example from Figure 1, the words "Magic" and "Flute" are removed from the text of its Wikipedia page.[30]

Although certainly not exhaustive, these set of models serve as strong baselines for the question answering component of Quizbowl. Section 10 identifies areas for future modeling work; throughout the rest of this work however we focus on completing a description of our approach to playing Quizbowl by combining guessers and buzzer (Section 6). Following this we describe how we evaluate these systems independently (Section 7.1), jointly (Section 7.3), offline (Section 7.1.2), and live (Section 8).

6. Buzzing

Excellence at Quizbowl requires answering accurately with as little information as possible. It is crucial, for humans and computers alike, to accurately measure their confidence and buzz as early as possible without being overly aggressive. The first part of our system, the guesser, optimizes for guessing accuracy; the second part, the buzzer, focuses on deciding when to buzz. Since questions are revealed word-by-word the buzzer makes a binary decision at each word: buzz and answer with the current best guess, or wait for more clues.

The outcome of this action depends on the answers from both our guesser and the opponent. If we buzz with the correct answer before the opponent can do so, we win 10 points; but if we buzz with an incorrect answer, we lose 5 points immediately, and since we cannot buzz again, the opponent can wait till the end of the question to answer, which might cost us 10 extra points in the competition.

Before we discuss our strategy to buzzing, let's consider a buzzer with perfect knowledge of whether the guesser is correct or not, but does not know anything about the opponent: a *locally optimal* buzzer. This buzzer would buzz as soon as the guesser gets the answer correct. A stronger buzzer exists: an omnipotent buzzer with perfect knowledge of what the

30. The first sentence of the page is "The Magic Flute is an operate in two acts by Wolfgang Amadeus Mozart to a German libretto by Emanuel Schikaneder".

opponent will do; it would exploit the opponent's weaknesses: delay the buzz to wait for the opponent to make a mistake.

The buzzer we develop in this paper targets a locally optimal strategy: we focus on predicting the correctness of the guesser and do not model the opponent. This buzzer is effective: it both defeats players in our gameplay dataset (Section 3.3) and playing against real human players (Section 8). The opponent modeling extension has been explored by previous work, and we discuss it in Section 9.

6.1 A Classification Approach to Buzzing

We formulate our buzzer as a binary classifier. At each time step, it looks at the sequence of guesses that the guesser has generated so far, and make a binary decision of whether to buzz or to wait.

Under the locally optimal assumption, the ground truth action at each time step equals the correctness of the top guess: it should buzz if and only if the current top guess is correct. The buzzer can thus be seen as an uncertainty estimator (Hendrycks and Gimpel, 2017) of the guesser.

The guesses create a distribution over all possible answers. If this distribution faithfully reflects the uncertainty of guesses, the buzzer could be a simple "if–then" rule: buzz as soon as the guesser probability for any guess gets over a certain threshold. This *threshold* system is our first baseline, and we tune the threshold value on a held-out dataset.

However, this doesn't work because the confidence of neural models is ill-calibrated (Guo et al., 2017). Our neural network guesser often outputs a long tail distribution over answers concentrated on the top few guesses, and the confidence score of the top guess is often higher than the actual uncertainty (the chance of being correct). To counter these issues, we extract features from the top ten guesser scores train a classifier on top of them. Some important features include a normalized version of the top ten scores and the gap between them; the full list of features can be found in the Appendix.

There is also important temporal information; for example, a steady increase in the score indicates that the guesser is certain about the top guess. To capture this kind of information, we compare the current guesser scores with the previous time steps and extract features such as the change in the score associated with the current best guess, and whether the ranking of the current top guess changed in this time step. The full list of temporal features can be found in the Appendix.

To summarize, at each time step, we extract a feature vector, including current and temporal features, from the sequence of guesses generated by the guesser so far. We implement the classifier with both fully connected Multi-layer Perceptron (MLP) and with Recurrent Neural Network (RNN). The classifier outputs a score between zero and one indicating the estimated probability of buzzing. Following the locally optimal assumption, we use the correctness of the top guess as ground truth action: buzz if correct and wait if otherwise We train the classifier with logistic regression; during testing, we buzz as soon as the buzzer outputs a score greater than 0.5. Both models are implemented in Chainer (Tokui et al., 2015); we use hidden size of 100, and LSTM as the recurrent architecture. We train the buzzer on the "buzzertrain" fold of the dataset, which does not overlap with the training set of the guesser, for 20 epochs with Adam optimizer (Kingma and Ba, 2015). Both buzzers achieve

test accuracy of above 80%, however, the classification accuracy does not directly translate into the buzzer's performance as part of the pipeline, which we look at next.

7. Offline Evaluation

A primary contribution of this work is describing our methodology for offline evaluation of guessing and buzzing models independently and jointly. The goal of our evaluation is to incorporate the incremental part of Quizbowl which is inherent to live play of Quizbowl. Section 7.1 introduces the metrics we use for evaluating guessing systems. Section 7.1.2 describes a metric—Expected Wins (EW)—that combines accuracy based metrics with the sequential decision-making aspect of Quizbowl while staying independent of buzzing models by assuming access to an oracle buzzer. Following a detailed error analysis (Section 7.2), Section 7.3 evaluates buzzing models by replacing this oracle buzzer with our proposed buzzing models.

7.1 Guesser Evaluation

Ideally, we would compare systems in a head-to-head competition: which can buzz first on a question (Sections 7.3 and 8). However, we divide our system into guessing and buzzing (as other systems might choose to do): can we still evaluate which guessing system is better on the pyramidal structure of quiz bowl questions (Section 2.2)?

Intuitively, a system that consistently buzzes correctly earlier in the question is better than a system that buzzes late in the question; any metric chosen should reflect this. A second requirement of our evaluation is that it not use a buzzing model so that we can independently evaluate the guesser.

7.1.1 Accuracy for Guessing Evaluation

The easiest and most common method for evaluating closed domain question answering methods is accuracy over all questions in the test set. Although we report accuracy on full questions, doing well on this metric should be considered a minimum bar of achievement since full questions are intentionally written to be answered easily (Section 2.3).

We also report model accuracy after observing only the first sentence of the question. We choose this since it is the hardest question answering sub-task in Quizbowl. This is a result of 1) this first point in the question where the answer is guaranteed to be discernible, and 2) following this point the question only gets easier. Although both of these metrics are useful in the development of question answering systems for Quizbowl, they are not enough to know how models models would fare against human opponents.

7.1.2 Computing Expected Wins Versus Humans for Guessing Evaluation

Evaluating through accuracy at the start and end of questions ignores how a system would fare against humans, and—once past the first sentence—does not incentive answering as early as possible. Our final metric is based on a score function f that both rewards early answering generally, and answering before human opponents. By creating this score function from historical gameplay data we incorporate performance against humans without explicitly requiring an opponent. The metric—expected wins (EW)—computes the expectation of the

score function f using system s over randomly sampled questions $q \in \mathcal{Q}$ and opponents $o \in \mathcal{O}$ (Equation 5). $p(q, o)$ represents the joint probability of playing question q against opponent o. The maximum score is one and the minimum score is zero.

$$EW(s) = \mathbb{E}_{(q,o)\in\mathcal{Q}\times\mathcal{O}}\left[f(s, q, o)\right] = \sum_{(q,o)\in\mathcal{Q}\times\mathcal{O}} p(q, o) f(s, q, o) \tag{5}$$

The score function in Equation 6 is the sum of points accumulated along all character positions j in the question. The character-wise score function can be broken down into two components: an indicator function $\mathbb{1}\left[s, q, j\right]$ and weight $w(t)$. The former takes value one when the system's guess at character j is correct, and the latter is an empirical estimate of the probability that a random opponent answers a random question before a fraction t of the question has been observed (Equation 7).[31]

$$f(s, q, o) = \sum_{j=1}^{k} f(s, q, o, j) = \sum_{j=1}^{k} \mathbb{1}\left[s, q, j\right] w\left(\frac{j}{k}\right) \tag{6}$$

The empirical estimate

$$w(t) = 1 - \frac{N_t}{N} \tag{7}$$

is computed from the gameplay dataset where N is the total number of question-player records and N_t is the number of question-player records where the human answered correctly before position t. We simplify Equation 5 to Equation 8 by substituting terms, making an independence assumption that $p(q, o) = p(q)p(o)$, and assuming that $p(q)$ and $p(o)$ are uniform distributions.

$$EW(s) = \frac{1}{|\mathcal{Q}\times\mathcal{O}|} \sum_{(q,o)\in\mathcal{Q}\times\mathcal{O}} \sum_{j=1}^{k} \mathbb{1}\left[s, q, j\right] w\left(\frac{j}{k}\right) \tag{8}$$

$$= \frac{1}{|\mathcal{Q}||\mathcal{O}|} \sum_{o\in\mathcal{O}} \sum_{q\in\mathcal{Q}} \sum_{j=1}^{k} \mathbb{1}\left[s, q, j\right] w\left(\frac{j}{k}\right) \tag{9}$$

$$= \frac{1}{|\mathcal{Q}|} \sum_{q\in\mathcal{Q}} \sum_{j=1}^{k} \mathbb{1}\left[s, q, j\right] w\left(\frac{j}{k}\right) \tag{10}$$

The EW over the test set can be computed exactly by iterating over each question, computing $\mathbb{1}\left[s, q, j\right]$ by querying the system for guesses and checking correctness, and using the empirical estimate for $w(t)$.[32]

The EW metric can easily be extended to evaluate combinations of guessing and buzzing systems. Instead of defining the indicator $\mathbb{1}\left[s, q, j\right]$ as whether a system's guess at position j is correct, we define it as whether the system's guess is correct *and* it decides to buzz in. The indicator used for guessing evaluation is equivalent to an oracle buzzer that only buzzes when the guess is correct. Here we use an oracle since the focus is on evaluating guessing systems; in Section 6 we compare using the oracle buzzer to differing buzzing systems.

31. We assume that at test time systems behave deterministically.
32. Since $w(t)$ is a step function we use a use a cubic polynomial fit to the data instead.

		Accuracy (%)						
		Start		End		Expected Wins		Time
Model	Wiki	Top	Mean	Top	Mean	Eager	Stable	(min)
Linear	yes	1.58	1.4±0.3	9.25	8.67±0.7	10.3	4.96	10
IR	no	6.37	6.37	54.4	54.4	43.8	38.6	2
IR	yes	7.95	7.95	54.7	54.7	45.9	39.6	4
DAN	no	8.00	8.4±0.4	54.0	54.1±0.2	44.2	37.8	81
DAN	yes	7.76	7.9±0.4	56.3	56.1±0.2	45.4	39.0	101
RNN	no	10.5	9.9±0.4	59.7	59.0±0.9	50.9	35.5	582
RNN	**yes**	**10.1**	**10.4±0.6**	**61.0**	**60.8±0.7**	**51.2**	**34.7**	**564**

Table 6: **Best model is bolded**. Each model is run five times with different random initializations. For the accuracy metrics we show the top score, mean score, standard deviation of scores. Expected win scores and training time are from the top model. The RNN models achieve the best results, but compared to DAN models take significantly longer to train for 2.3% absolute improvement. Using Wikipedia as training data often helps, but is not a universal trend.

7.1.3 Guesser Performance Comparison

We evaluate our proposed methods using start accuracy, end accuracy, and expected wins (Table 6). We report an our primary version of EW (eager), and an alternative version of EW (stable).[33] All models struggle with answering at the start of the question with the best score achieving only about 11% accuracy. This is unsurprising as the first sentence contains the most difficult clue, and they are difficult for even the best human players. The story is quite different at the end of the question since nearly all questions have giveaway clues. These clues are specifically designed to make the question extremely easy to answer so accuracy here should be near perfect. Despite this the best model achieves only 61% accuracy so there is much room for improvement. The EW scores for all models fall in between, but trend closer towards end accuracies. The difference in eager versus stable EW scores indicates that many although models can answer early correctly they often change their answer.

Except for linear n-grams based models, all models were reasonably competitive with each other. Although the neural models are best best, the IR model is a surprisingly strong baseline especially considering its relative implementation simplicity and superior speed. Between the DAN and RNN, the RNN achieves the best accuracy, but takes seven times as long to train for absolute accuracy boosts ranging from .03 to .07. We now move our attention to inspecting the similarities and differences in model behavior.

7.2 Identifying Sources of Error

Arguably the most crucial aspect of developing machine learning models is studying their behavior through error analysis and model inspection at both aggregate and instance levels.

33. Stable only awards points if the current answer and all subsequent answers are correct. Eager awards points anytime the current answer is correct, even if the model is incorrect at later time steps.

In Section 7.2.1 we show that neural and IR models tend to make similar correct and incorrect predictions indicating that they both have an over-reliance of key-phrase matching. Section 7.2.2 establishes the scarcity of training data for many answers as a major source of error, and identifies which models benefit from using Wikipedia sentences as additional training examples. In Section 7.2.3 we manually break down test set errors.

7.2.1 Neural Models Over-rely on Pattern Matching

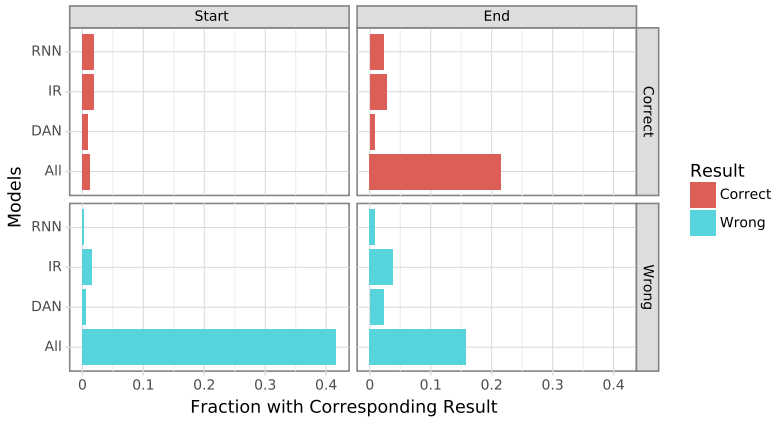


Figure 13: We break down results for each question dependent on whether all models were wrong, right, or when individual models were the only ones to be correct or wrong. All indicates all models were correct or wrong, and *model* indicates only that model was correct or wrong. Overall accuracies on—for example—the RNN is recovered by combining the bars for "Correct All", "Correct RNN", "Wrong IR", and "Wrong DAN". From this plot we glean that models even of differing families (neural vs information retrieval) agree on most predictions.

Neural and IR models tend to make similar predictions (Figure 13); when one model is wrong the other is likely also wrong, and similarly when a model is correct the other is likely also correct. The relative largeness of the "All" bars in Figure 13 shows this. When models do disagree the IR is usually the dissenter (sum "IR" bars). Even then the disagreement is less than 5% and 10% at the start and end of question. These disagreements are infrequent enough to conclude that the prediction behavior of neural and IR models do not significantly diverge.

Since IR systems are explicitly pattern matching based, this hints that neural Quizbowl models may merely be sophisticated pattern matching systems, which is an observation made about neural models on other tasks as well (Jia and Liang, 2017; Rajpurkar et al., 2018; Feng et al., 2018). Next we take advantage of the similarity in predictions to show at the instance level that this hypothesis is consistent.

For our instance level analysis we randomly sample examples of correct and incorrect predictions. First we randomly sample a question that all models answer correctly after the first sentence. In Figure 14 we show the first sentence of the test question, a matching sentence from a training question, and the answer. In the figure we highlight a high match key phrase that triggered the answer in the IR model and is likely responsible for triggering

Test Question (first sentence):
A holder of this title commissioned a set of miniatures to accompany the story collection Tales of a Parrot.
Training Question (matched fragment):
A holder of this title commissioned Abd al-Samad to work on miniatures for books such as the Tutinama and the Hamzanama.
Answer: Mughal Emperors

Figure 14: A test question that was answered correctly by all models after the first sentence; a normally very difficult task for both humans and machines. We suspected that since all models—specifically the IR one—answered correctly that there was a training question phrase with key-phrase overlap with the first sentence. We confirmed this was the case and show the training question that triggered the IR system to answer correctly with overlapping terms highlighted.

the answer in the neural models as well. This example indicates that despite the low accuracy at starting positions that at least some questions that are correctly answered early may be due to accidental overlap between phrases in the training and test set questions. One direction for future work may be to use IR-based models to find questions with this problem and remove them from the test set to make scores properly reward multi-hop reasoning early on in questions versus lucky spurious pattern matching.

The natural next step would be to consider a randomly sampled question that the RNN answered correctly, but that the IR model did not get correct (Figure 15). Since neural models are difficult to interpret we attempt a manual interpretation by examining the twenty seven training questions for the answer. Our initial analysis reveals that the most frequently occurring words in the training data are "phenomenon" (twenty three times), "model" (seventeen times), "equation" (thirteen times), "numerically" (once), and "tensor" (once). If we remove these words or substitute them with synonyms then perhaps the model will not maintain the correct answer especially given that the confidence score is already low.

To test the hypothesis that the model is over-reliant on these terms we first removed the term "phenomenon" from the question and immediately find that the model changes its answer to Ising model (a mathematical model of ferromagnetism in statistical mechanics). If we instead substitute the term for with synonyms such as "occurrence", "event", and "observable event" the answers are also incorrect. Similarly if "model" is replaced by "representation" the RNN also produces incorrect answers. From this we conclude that at least for this question unstable key-phrase matching is responsible for producing the correct answer, and that the model is not robust to these semantically equivalent modifications.

This style of analysis falls under a larger area of research by Ribeiro et al. (2018) on semantically equivalent adversarial rules (SEARS). The aim of SEARS is to create rules for mutating textual inputs such that the meaning is unchanged. In the example from Figure 15 we use manually created, example specific synonym rules to test the model's robustness to SEARS attacks on this example. We leave extending SEARS attacks at the dataset level to future work. We discuss present and future research directions for using adversarial-based

Test Question (first sentence):
This phenomenon is resolved without the help of a theoretical model in costly DNS methods, which numerically solve for the rank-2 tensor appearing in the RANS equations.
Answer: Turbulence **Score** (rnn): .0113
Synonym Attacks: phenomenon → event, model → representation

Figure 15: This example shows an instance where only the rnn model answers correctly (importantly the ir model answers incorrectly). To test the robustness of the model to semantically equivalent input modifications we use sears-based (Ribeiro et al., 2018) synonym attacks and find that when any of the sample rules are used the answer is no longer correct. While it is good that the score reflects this instability, the aggregated accuracy metrics do not reflect this. However it is likely that a buzzer model would not buzz on this answer which highlights that one benefit of evaluating guessers and buzzers together is that it integrates model uncertainty into the evaluation.

methods to improve the qanta dataset in Section 10. In summary our neural models are not robust to attacks targeting over-reliance on key terms and phrases.

7.2.2 Errors Caused by Data Sparsity

A major source of error in our models is the dearth of training data for many answers (sparsity). The most egregious version of these errors are the 17.9% of test questions with zero corresponding training examples. The problem persists beyond this as many answer have very few training examples. We confirm this in Section 5.4 by showing that the distribution of training example counts versus unique answer follows a power law (Figure 12). We take a step further and show in Figure 16 that the sparsity of training data is a substantial source of errors across all models. We also show that this sparsity adversely affects at least fifty percent of test set questions.

Figure 16 also shows the effect of our Wikipedia-based data augmentation technique on accuracy. The ir systems improve substantially in all cases, and neural models in only a few. This is likely due to ir systems being more robust to mixed domain data than neural models. Despite the mixed domain data all model accuracies improve by the end of question. This indicates that in the limit of long text that the divergence between domains is small enough for even neural models to improve, even if that is not the case on short, single sentence text. Since this data augmentation technique shows promise we discuss in Section 10 research directions that use methods from domain adaptation, low resource cross-lingual classification, and few-shot learning to make the out-of-domain Wikipedia data more useful. We now continue our error analysis by examining errors that cannot easily be accounted for due lack of training data.

7.2.3 Error Breakdown

We conclude our error analysis by manually inspecting and breaking down the errors made by the rnn model at the start and end of questions. Of the 2151 questions in the guesstest set,

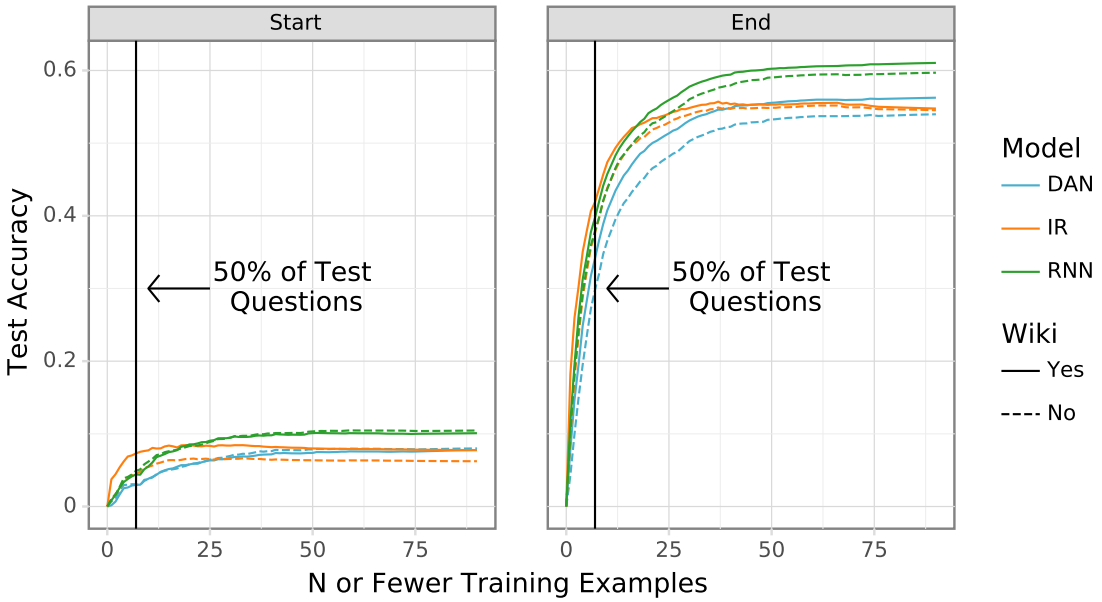


Figure 16: Model accuracy is strongly affected by the number of training examples corresponding to each test question at both the start and end of question. This is a significant source of issue since accuracy on at least 50% of test questions—those with seven or less training examples—is significantly lower for all models. A possible mitigation against the data scarcity issue is data augmentation with Wikipedia which proved effective when combined with the IR model. The case for their use in neural models is not nearly as strong since it appears to improve accuracy at the end of questions, at the expense of accuracy near the beginning.

386 have zero training examples leaving 1765 questions that are answerable by our models. Of these remaining questions the RNN answers 1540 incorrectly after the first sentence and 481 at the end of the question. To avoid errors due to data scarcity we further filter to questions with at least 25 training examples. The number of errors on this subset at start and end of question is 289 and 36. Table 7 lists reasons for model errors on a random sample of 50 errors from the start of question and all 36 errors from the end of question.

The predominant sources of error are when the model predicts the correct answer type (e.g., person, country, place), but chooses the incorrect member of that type. This accounts for errors such as choosing the wrong person, country, place, or event. The RNN especially confuses countries as its a source of errors at both start and end of questions. Manual inspection of the questions indicates that questions in which two countries are involved—such as a question regarding the Adam Onis Treaty between Spain and the United States—are particularly difficult as in Figure 17. That the relative absence of incorrect answer type errors at the end of questions is attributable to the tendency of late clues including the answer type (such as "name this country…"). Our primary takeaway from inspecting errors is that models are not hopeless as they are often correct on one aspect of the answer.

Error Reason	Start Count	End Count
Wrong Country	11	17
Wrong Person	16	2
Wrong Place	1	5
Wrong Type	15	5
Wrong Event	0	1
Nonsense	7	2
Annotation	1	4

Table 7: An error breakdown for questions with at least twenty five training examples. To analyze errors at the start of questions we randomly sample fifty errors, and for end of question we examining all thirty six errors. End of question errors principally focus around wrong country guesses such as in Figure 17 where the model answers United States instead of Spain. Errors at the start of the question are more diverse. The most common error is guessing the correct answer type, but not the specific member of that type; examples of this error class include answering Albert Einstein instead of Alan Turing, or Iowa instead of Idaho.

Test Question: This country seized four vessels owned by Captain John Meares, which were registered in Macau and disguised with Portuguese flags, starting a dispute over fishing rights. To further negotiations with this country, Thomas Jefferson signed the so-called "Two Million Dollar Act." This country agreed not to police a disputed spot of land, which was subsequently settled by outlaws and "Redbones", and which was called the "Neutral Ground." This country was humiliated by England in the Nootka Crisis. Harman Blennerhassett's farm on an island in the Ohio River was intended as the launching point of an expedition against this European country's possessions in a plan exposed by James Wilkinson. This country settled navigation rights with the United States in Pinckney's Treaty, which dealt with the disputed "West" section of a colony it owned. For 10 points, name this European country which agreed to the Adams-Onis Treaty handing over Florida.
Guess: United States **Answer**: Spain

Figure 17: Although the answer to this question is Spain, many of the terms and phrases mentioned are correlated with the United States. We believe that this is the reason that the RNN model answered with United States instead of the correct answer Spain. This is one of many examples where the model answers with the correct answer type (country), but incorrect member of that type.

In the process of manual error breakdown we also find five annotation errors where the assigned Wikipedia answer did not match the true answer. This low number of errors further validates the robustness of our answer mapping process.

Model	ACC	EW	Score
Threshold		0.013	-9.98
MLP	0.840	0.272	-2.31
RNN	0.849	0.302	-1.01
Optimal	1.0	0.523	2.19

Table 8: The accuracy (ACC), expected wins (EW), and Quizbowl score (Score) of each buzzer on the validation set. Both MLP and RNN outperform the static threshold baseline by a large margin, but there is still a considerable gap from the optimal buzzer.

7.3 Evaluating the Buzzer

We first evaluate our buzzer against the locally optimal buzzer which buzzes as soon as the guesser gets the answer correct. However, this can be overly ambitious and unrealistic since the guesser is not perfectly stable: it can get the answer correct by chance, then vacillate between several candidates before settling down to the correct answer. To account for this instability, we find the first position that the guesser stabilizes to the correct answer and set it as the optimal buzzing position. To be exact, we start at the last position that the guess is correct, go backwards until the guess is incorrect and consider this the locally optimal buzzing position; we set the ground truth to all positions before this to zero, and all positions after it to one.

We use the same guesser (RNN) in combination with different buzzers, and quantitatively compare their performance using the expected wins metric introduced in Section 7.1.2. Table 8 summarizes the results. Both MLP and RNN buzzers outperform the static threshold baseline by a large margin, but there is still a considerable gap between RNN and the optimal buzzer.

A lower performance—buzzing at suboptimal positions—means the buzzer is either too aggressive or not aggressive enough. To characterize their weaknesses, we compare the buzzers' behavior over time in Figure 18. The static threshold buzzer is too aggressive, especially early in the questions. This behavior to some extent resonates with the observation that the confidence of neural models needs calibration Guo et al. (2017). The difference between MLP and RNN is small but RNN is less likely to be overly aggressive early in the question.

For a more fine-grained analysis, we use the gameplay dataset (Section 3.3) to simulate games where our system plays against individual human players, which allows allow us to break down the buzzer's behavior on the level of individual questions. Based on the ability of the guesser, the games questions are first categorized based on the *possibility*: "possible" means the guesser gets the answer correct before the opponent answers correctly; in the other case it is impossible for the buzzer to do anything to beat the opponent. Based on this categorization, Figure 19 further breaks down the performance by the outcomes: the RNN is less likely to be overly aggressive in both possible and impossible cases.

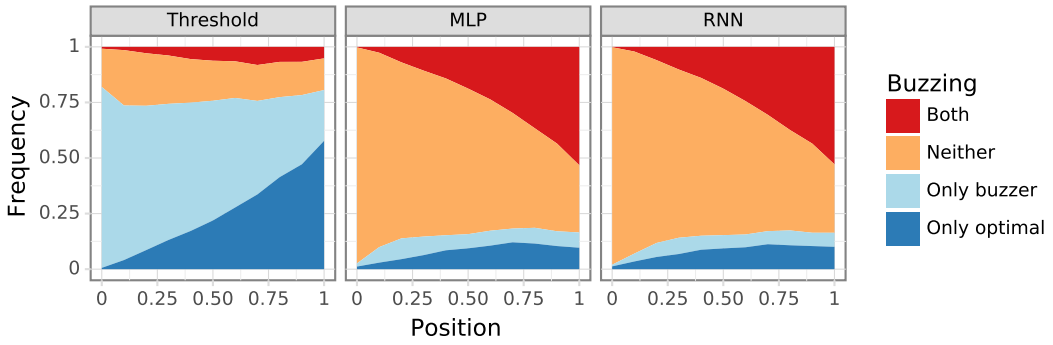


Figure 18: Comparing buzzers' behavior over time against the optimal buzzer; we want to maximize the red areas and minimize the blue areas. The static threshold baseline is overly aggressive, especially at earlier positions in the question; MLP and RNN both behaves reasonably well, and the aggressiveness of RNN is slightly more balanced early on in the question.

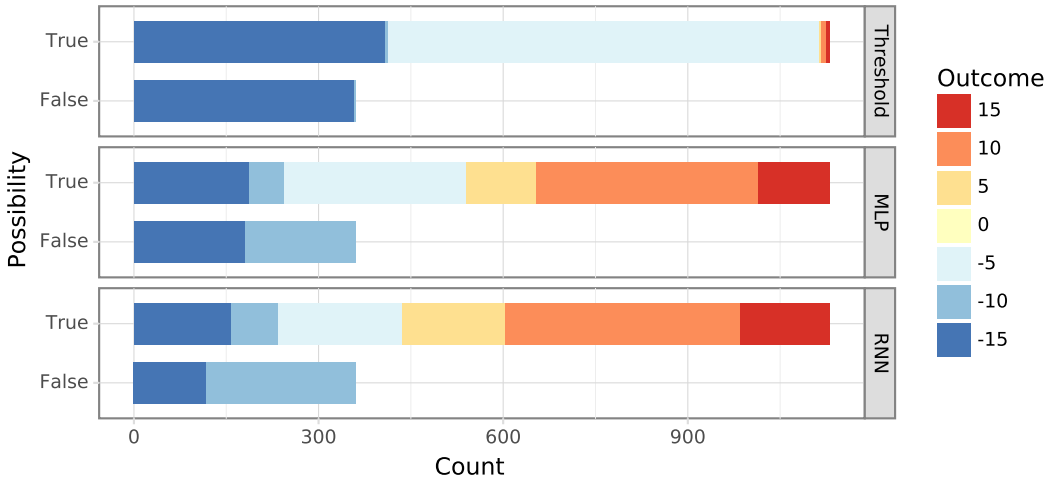


Figure 19: Breaking down the buzzer's performance on the individual question level. Impossible question means there is nothing the buzzer can do to beat the opponent. It is clearer that RNN performs better than MLP, making fewer mistakes of being overly aggressive.

8. Live Exhibition Events

No amount of thorough experimentation and analysis of machine learning systems can match the public interest and rubber-meets-the-road practicalities of live matches between humans and machines. IBM's Watson in Jeopardy!, Deep Blue in chess, and Google's AlphaGo in Go were tremendous scientific achievements, and the degree to which their success impacted the public is similarly impressive. In the case of chess and Go they transformed how the games are played through insight gained from the collaboration of humans and machines.

In a similar spirit we have hosted eight live events since 2015 where we showcase our research to the general public, and have humans and machines compete against each other.[34] Except for our NIPS 2015 Best Demonstration against ML researchers, our system's opponents have been strong trivia players. Their achievements include victories in numerous national Quizbowl championships (high school and college), Jeopardy!, and similar trivia competitions.

Our inaugural event in 2015 at the Quizbowl High School National Competition Tournament (HSNCT) pitted an early and vastly different version of our system against a team of tournament organizers in a match that ended in a tie.[35] Later that year a similar system defeated Ken Jennings of Jeopardy! fame at the University of Washington, but lost convincingly (145–345) at HSNCT 2016. The subsequent year at HSNCT 2017 our newly designed system, a predecessor to the IR guesser and RNN buzzer in this work, narrowly defeated its opponents (260–215). Although this impressive result appears to follow the trend of machines improving until they defeat skilled humans, it is far from the whole story.

In parallel with these events, we hosted events where teams—humans and machines—were selected from open competition. Our first of these style events was hosted as part of a NAACL 2016 workshop on question answering. Before the event local high school teams competed against each other, and researchers submitted their machine systems which also played simulated matches against each other. At the event the best human and machine teams played against each other with the high school team defeating an early version of Studio OUSIA's system (Yamada et al., 2017, 2018).[36] In 2017 we hosted a similar workshop at NIPS where an improved version of OUSIA's system yet again defeated its machine competition, but this time also defeated the invited human team. In 2018 we hosted two competitions featuring questions that avoid over-used clues; during the spring event the same OUSIA system lost (30–300).

Events and collaborations like these show that Quizbowl is more than just another question answering task. By engaging with the Quizbowl community to digitize Quizbowl questions in a machine readable form we not only made our research possible, but enabled the ecosystem of tools that students now rely on to practice with before competitions. In the next step towards deeper collaboration with this community we are building ways for humans and machines to cooperate in competition (Feng and Boyd-Graber, 2019) and in writing questions (Wallace et al., 2018). We accomplish all this while simultaneously providing ways for students of all ages to engage with and benefit from research through our live exhibition events.

9. Related Work

The number and variety of question answering tasks and datasets has exploded in recent years. Here we discuss the challenges they try to address, and compare with Quizbowl, focusing on question complexity and the context required to answer. We show that Quizbowl cannot be solved with simple pattern matching, as it requires complex language and global factual context. It also challenges us to accurately measure the models' confidence so that it can be used in the decision to wait or buzz.

34. Videos of our events are available at `http://events.qanta.org`.
35. Our software did not handle ties correctly and terminated instead of playing tiebreaker questions.
36. The OUSIA system embeds words and entities separately, and uses a DAN-based architecture over these.

9.1 Datasets and Tasks

The least complex types of questions are often called "simple questions". Examples of this type of task include WebQuestions (Berant et al., 2013), SimpleQuestions (Bordes et al., 2015) and WikiMovies (Miller et al., 2016) which use knowledge graphs and templates to automatically generate large datasets. The drawback of this approach is that questions require very little reasoning to answer (as little as one fact), and the language is not diverse since its generated from templates. As a consequence SimpleQuestion is now nearly solved—after considering ambiguity in the data bounds performance at 83.4%—with standard deep learning methods achieving 78.1% accuracy (Petrochuk and Zettlemoyer, 2018). In Quizbowl the closest comparison to these "simple questions" are the final giveaway clues in questions which are specifically designed to be easy so that novice players can get them correct. Unlike these tasks, answering on the giveaway clue is seen as a minimum bar, and even the best models we analyzed fall short here.

As we have shown for Quizbowl, using trivia questions for question answering datasets increases their diversity and quality. Outside the Quizbowl community trivia has many other large communities with troves of already written questions. Taking advantage of this SearchQA is built from Jeopardy! questions (Dunn et al., 2017), TriviaQA is built from fourteen trivia sites (Joshi et al., 2017), and Quasar-T is built from a set of questions collected by a reddit user (Dhingra et al., 2017). In contrast, the qanta dataset is built from questions previously used in competitive Quizbowl play. All or most of the questions in these datasets can be framed as either entity classification—the same way we frame the Quizbowl task—or as reading comprehension over supporting documents mined from Wikipedia and the web.

Framing factoid question answering as reading comprehension through span selection in documents traces back to TrecQA (Voorhees and Tice, 2000). More recent datasets following this framework include WikiQA (Yang et al., 2015), SQuAD (Rajpurkar et al., 2016), WikiReading (Hewlett et al., 2016), ms marco (Nguyen et al., 2016), and NewsQA (Trischler et al., 2017). These datasets are differentiated from other reading comprehension datasets—such those asking questions about non-factual stories—since their questions predominantly are answerable without the context provided a clever enough human or model.

Quizbowl can also be framed as reading comprehension if certain challenges are solved. The first is that pragmatically it is difficult to—at scale—accurately retrieve and verify that supporting evidence contains the information required by the question.[37] This is especially an issue in Quizbowl since the clues from early in the question often require multiple pieces of disparate knowledge. To motivate progress towards creating verified evidence for Quizbowl we release candidate evidence for each question using an ir system.

The second challenge is also problematic in SQuAD. If the evidence is guaranteed to contain the answer then the task is easier since it can be reduced to multiple choice over the small number of entities in the evidence. Rajpurkar et al. (2018) show that as a result SQuAD models only need to select the span most related to the question to attain high accuracy. Rajpurkar et al. make SQuAD 2.0 more difficult by including unanswerable yet plausible questions to force models to calibrate their confidence much as a buzzer like does in Quizbowl. A similar mechanism could be employed in Quizbowl by mixing verified and

37. As a convenience for system builders we still provide unverified supporting evidence using methods from information retrieval.

unverified evidence so that models must calibrate their trust that the evidence contains the answer.

9.2 Human-in-the-Loop Adversarial Examples

Quizbowl players are motivated to write better questions, and interpretations of questions with machine learning tools can help with this. Wallace et al. (2018) developed a question writing interface that exposes—in real time—model predictions and interpretations of those predictions to users. Writers use these interpretations to improve questions by avoiding clues that are too easy or that have been used before. These more robust questions are also adversarial examples; although, unlike prior work that focuses on automatic generation of adversarial examples (Jia and Liang, 2017; Iyyer et al., 2018; Ribeiro et al., 2018), Wallace et al. (2018) use a human-in-the-loop process. Questions from this work were used in the live competitions in 2018 discussed in Section 8.

A major motivation for exposing model predictions to humans generating datasets is to improve the generalizability of the dataset. This is an increasingly salient challenge as several tasks have been shown to be easier than supposed due to artifacts; these include natural language inference (Gururangan et al., 2018), visual question answering (Goyal et al., 2017), simple factoid question answering (Petrochuk and Zettlemoyer, 2018), and reading comprehension (Chen et al., 2016; Kaushik and Lipton, 2018; Sugawara et al., 2018). Engaging with the Quizbowl community to write better questions is one way to develop and test model interpretations, visualizations, and interfaces for robust dataset creation.

9.3 Interruptable QA and Model Calibration

Knowing when we can trust a model's predictions is an increasingly studied and important challenge in machine learning research. Traditionally one could use the model's score to know when to trust it. However, using prediction scores from state-of-the-art deep learning models is unreliable (Guo et al., 2017; Feng et al., 2018); they are often overly confident when they should not be. For many settings—such as predicting medical diagnoses—*it is not enough for models to be right*; they need to reliably estimate the likelihood their prediction is correct.

In Quizbowl knowing the answer is just as important as knowing *when* you are right. Although model calibration is now part of SQuAD (Rajpurkar et al., 2018), it has always been an essential part of playing Quizbowl. In this work we address model calibration separately by dividing the task into guessing and buzzing; the buzzing model calibrates the predictions of the guessing model. Quizbowl is a natural fit for research in calibration question answering models.

9.4 Multi-Hop QA

Another trend in question answering is an emerging focus on tasks that specifically require multi-step—also known as multi-hop—reasoning. In WikiHop the dataset is automatically constructed using Wikipedia, Wikidata and WikiReading, and they pose the task as multi-hop reasoning across several documents (Welbl et al., 2018). HotPotQA is a similarly structured task, but was created via crowdsourcing rather than automatically (Yang et al., 2018).

In Quizbowl, questions regarding novels often incorporate multi-hop reasoning questions. For example, the start of one question states, "He left unfinished a novel whose title character forges his father's signature to get out of school and avoids the draft by feigning a desire to join." In this question and many others a character in a novel is described, and the quantity of interest is the author; in this case Thomas Mann. Focusing on solely the first sentences of Quizbowl questions is one way to get diverse examples of multi-hop reasoning for work in improving multi-hop reasoning models.

Despite the explosion and evolution of question answering datasets Quizbowl remains compellingly unique, and addresses several concerns which datasets are only now beginning to target. This is no accident. Quizbowl was not created with machine learning research, natural language processing research, or TV entertainment in mind; it was created as a means to motivate students to learn about our world, and show their skill through fair competition that rewards true mastery of human knowledge. The Quizbowl task and QANTA dataset are beneficiaries of decades of refinement in asking good trivia questions.

10. Future Work

Here we identify research directions that are particularly well suited to the QANTA dataset and Quizbowl. Rather than focus on general research directions in question answering we identify areas that best take advantage of Quizbowl's unique aspects. Specifically we focus on directions using its interruptible and pyramidal nature, the perpetual influx of new and diverse questions from annual tournaments, and the supportive community interested in making Quizbowl even more supportive of scholastic learning and achievement.

10.1 Pre-Trained Language Models

One direction we do not investigate in this work—but which should yield improvements—is the use of large pre-trained language models (Dai and Le, 2015). More recently implementations of this idea such as ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018) have achieved state-of-the-art across a variety of NLP tasks. We expect similarly positive results if applied to Quizbowl via either feature extraction or fine-tuning. Since the primary focus of this work is to frame Quizbowl as a machine learning task, introduce the datasets for the task, and carefully analyze the datasets we leave improving on our models with pre-trained language models to future work.

10.2 Generalization in Factoid Question Answering

Although some syntactic forms in Quizbowl may be overly specific, its ever growing size and diversity makes it attractive for studying more generalizable factoid question answering. The reasons for this are twofold. First, as discussed in Section 3, the QANTA dataset is already diverse in topics, syntax, and in range of answers (over twenty-five thousand considering all data folds). Second and more importantly though the dataset is growing year over year. Since 2007 the size of the dataset has over quadrupled, and the growth shows no sign of slowing down. Figure 20 shows this growth over the past twenty years.

As the dataset continues to grow it will demand that machines and humans broaden their knowledge of past events while also updating their knowledge with current events. Every

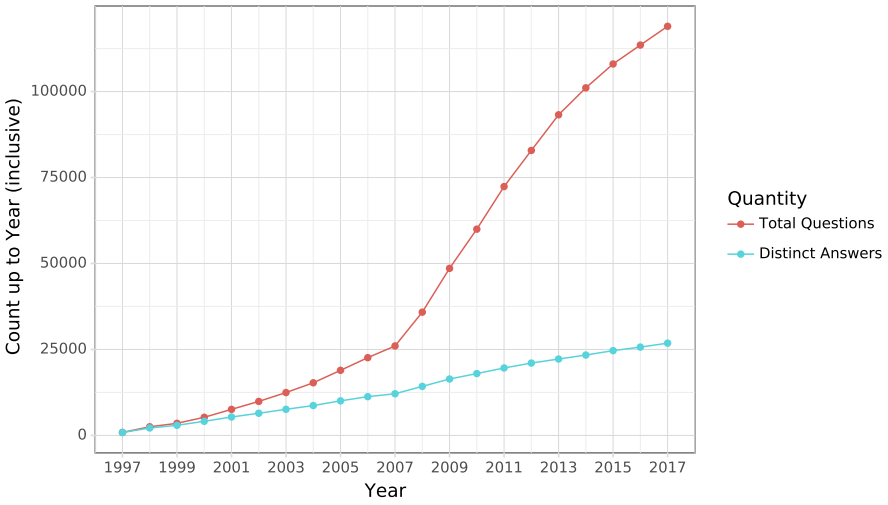


Figure 20: The growth of the QANTA dataset in number of questions and number of distinct answers over the past twenty years starting in 1997. The dataset has grown by at least 5,000 questions every year since 2010. All questions with matched answers are included, and we construct the plot by using the tournament year of each question. Independently, participation in Quizbowl (and thus number of students writing questions) has roughly doubled every year since 2008.

year presents an opportunity to test both how well models generalize to novel questions and how well they generalize to questions about current events. For example, in a 2017 exhibition match our model missed a question about the company responsible for driving down rocket launch costs (SpaceX); a phenomenon which only manifested itself several years prior. With this ever present influx of new questions every year we believe this opens new research directions in testing the generalization of models.

10.3 Few Shot Learning and Domain Adaptation

One well suited research direction with Quizbowl is adopting methods from domain adaptation and few shot learning for factoid question answering. A large source of error in our models is a scarcity of training examples for most answers (see Figures 12 and 16). Zero and few shot learning in NLP and Computer vision face similar challenges of using shared structure to improve the use of the scarce training data (Xian et al., 2018). Correcting errors induced by the scarcity of training examples is an exciting research direction.

One solution to the scarcity of training data is domain adaptation. In Section 5.4 we experiment with a crude data augmentation technique which we believe could be greatly improved with model based domain adaptation. From (frustratingly) easy methods to more sophisticated methods sharing there are a variety of options investigate and extend (Blitzer et al., 2007; Daume III, 2007; Kim et al., 2016; Chen and Cardie, 2018).

10.4 Explainable and Trustable Machine Learning

Developing explainable and trustable machine learning systems overlaps with the goals of machine learning researchers and the Quizbowl community. A simple example of this is buzzing. Since playing Quizbowl requires deciding when to buzz it naturally requires having accurate estimates of the model correctness; a task that many neural models fail at (Section 9.3). At bare minimum models, should be able to tell us when to trust them and when not to. Investigating topics such as this and model calibration through playing Quizbowl is an open research direction.

Humans can improve at games like Chess and Go by learning from machines; similarly, Quizbowl players can learn from and cooperate with machines as well. For example, Feng and Boyd-Graber (2019) built and evaluated interpretations of machine learning models based on how effective they were at improving human live play. A related research direction would be to use these interfaces and insights about models of human learning—such as the effectiveness of spaced repetition (Ebbinghaus, 1885)—to improve knowledge retention as Settles and Meeder (2016) do for language learning. Both of these directions fuse work in human computer interaction and interpretation of machine learning algorithms.

Our collaborative research in Quizbowl thus far is only a beginning. Quizbowl easily supports work in factoid question answering and sequential decision-making. We list several challenges in these based on our experiments, but there are certainly more. Beyond playing Quizbowl, there are opportunities in human-in-the-loop research that can improve interpretations of machine learning models while producing useful artifacts such as adversarial datasets. We hope that our work in establishing the qanta datasets and Quizbowl as a machine learning task empowers others to contribute to these and other future research.

11. Conclusion

This article introduces and argues for Quizbowl: an incremental question answering task. Solving Quizbowl questions requires sophisticated nlp such as resolving complex coreference, multi-hop reasoning, and understanding the relationships between a gigantic menagerie of entities that could be answers. Fundamental to answering Quizbowl questions is that the questions are incremental; this is both fun and good for research. It is fun because it allows for live, engaging competitions between humans and computers. This format—the product of refining human question answering competitions over decades—is also good for research because it allows for fair, comprehensive comparison of systems and iterative improvement as systems answer questions earlier and earlier.

To evaluate systems we use three methods: offline accuracy-based metrics adapted to the incremental nature of Quizbowl, simulated matches against machines and humans, and live exhibition matches. Although the best models achieve just over sixty percent accuracy at the end of questions, they only barely break ten percent near the start indicating there is much work to be done in improving models.

Improving Quizbowl models can incorporate many commonplace tasks in nlp other than question answering. Reasoning about entities can be improved through better named entity recognition, entity linking, coreference and anaphora resolution. Some of the more difficult clues in Quizbowl however presume that the player has read and integrated the content of books such as important plot points. Further work in reading comprehension

and summarization could help answer some of these questions. At a more general level, the extraction of information from external knowledge sources (such as books or Wikipedia) is important since the distribution of training examples per answer is heavily skewed and some new questions ask about current events. Improving Quizbowl models requires and can further motivate advances in these tasks and others.

However, the benefits to research go beyond format or specific sub-tasks, and extend to our symbiotic collaboration with the Quizbowl community. For example, our exhibition matches double as outreach events and opportunities to put machine systems to the test on previously unseen questions. Another area of active research is in collaborating with the Quizbowl community to further improve the quality of questions for humans and machines alike. Writers are empowered with machine learning tools to discover bad clues which helps create questions more interesting to humans that consequently better test the generalization of systems. In this the goals of the Quizbowl and communities align; we both seek to create datasets that discriminate different levels of language and knowledge understanding.

Quizbowl isn't just another dataset or task; it is a rich platform for NLP research that co-evolves with the Quizbowl community. From the digitization of Quizbowl questions to our pioneering online interface which created and popularized online Quizbowl play. From cooperative play between humans and machines to providing tools for better writing better question we have built a symbiotic relationship that continues to yield productive research while moving the Quizbowl community forward. We hope that new unforeseen research directions will continue emerging from this collaboration while simultaneously giving back to the Quizbowl community through new and exciting ways of engaging with state-of-the-art research in machine learning and natural language processing.

Acknowledgements

Rodriguez and Boyd-Graber are supported by National Science Foundation Grants IIS1320538 and IIS1822494. Feng is supported under subcontract to Raytheon BBN Technologies by DARPA award HR001-15-C-0113. Amazon Web Services Cloud Credits for Research provided computational resources that supported many of our experiments and internal infrastructure. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

Many individuals contributed their ideas and work to this paper for which we are eternally grateful. Davis Yoshida was influential in developing the Wikipedia data augmentation method and contributed to many insightful discussions. Others who have been intellectually involved in our work with Quizbowl include Brianna Satinoff, Anupam Guha, Danny Bouman, Varun Manjunatha, Hal Daume III, Leonardo Claudino, and Richard Socher. We also thank the members of the CLIP lab at the University of Maryland for helpful discussion.

Next, we appreciate those that improved this paper through comments and edits on the manuscript. In particular, we thank Yogarshi Vyas, Joseph Barrow, Alvin Grissom, and Hal Daume III for their valuable editorial feedback.

Nathan Murphy and R. Robert Hentzel have been incredibly supportive of using Quizbowl for outreach with the public; they have multiple times hosted our exhibition events at the high school national championship tournaments. We also thank the participants that played against machine systems in these events; these include Ophir Lifshitz, Auroni Gupta, Jennie

Yang, Vincent Doehr, Rahul Keyal, Kion You, James Malouf, Rob Carson, Scott Blish, Dylan Minarik, Niki Peters, Colby Burnett, Ben Ingram, Alex Jacob, and Kristin Sausville. Ken Jennings played against our system at an event hosted by Noah Smith at the University of Washington. Ikuya Yamada and Studio OUSIA entered their systems which competed against human teams at several of our events.

Finally, this work would never have been possible without the support of the Quizbowl community. We are grateful to the original authors of questions, and those who helped us collect our current dataset. The maintainers of `quizdb.org` and `protobowl.com` allowed us to use their websites to build our dataset. The first versions of work in Quizbowl used a dataset collected by Shivaram Venkataraman. National Academic Quiz Tournaments, LLC provided access to their proprietary questions which we used in prior iterations of our systems.

References

Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15:1111–1133, 2014.

John Bauer. Shift-reduce constituency parser, 2014. URL `https://nlp.stanford.edu/software/srparser.html`.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy S. Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*, 2013.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the Association for Computational Linguistics*, 2007.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of Empirical Methods in Natural Language Processing*, 2012.

Xiaoyong Chai, Lin Deng, Qiang Yang, and Charles X. Ling. Test-cost sensitive naive bayes classification. *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 51–58, 2004.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the Association for Computational Linguistics*, 2016.

Xilun Chen and Claire Cardie. Multinomial adversarial networks for multi-domain text classification. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2015.

Hal Daumé, Nikos Karampatziakis, John Langford, and Paul Mineiro. Logarithmic time one-against-some. In *Proceedings of the International Conference of Machine Learning*, 2017.

Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the Association for Computational Linguistics*, 2007.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*, 2017.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

H. Ebbinghaus. *Memory: a contribution to experimental psychology.* Teachers College, Columbia University, New York, NY, USA, 1885.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

Shi Feng and Jordan Boyd-Graber. What can ai do for me: Evaluating machine learning interpretations in cooperative play. In *International Conference on Intelligent User Interfaces*, 2019.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31:59–79, 2010.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision*, 127:398–414, 2017.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2015.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference of Machine Learning*, 2017.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Bob Harris. *Prisoner of Trebekistan: a decade in Jeopardy!* 2006.

He He, Jordan L. Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *Proceedings of the International Conference of Machine Learning*, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. Wikireading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the Association for Computational Linguistics*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9: 1735–1780, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference of Machine Learning*, 2015.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*, 2015.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing*, 2017.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*, 2017.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

Divyansh Kaushik and Zachary Chase Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. Frustratingly easy neural domain adaptation. In *Proceedings of International Conference on Computational Linguistics*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Julian Kupiec. Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016.

Paul Lujan and Seth Teitler. Writing good quizbowl questions: A quick primer. `https://www.ocf.berkeley.edu/~quizbowl/qb-writing.html`. Accessed: 2018-12-04.

Subash Maddipoti. Subash maddipoti's tips on question writing. `https://acf-quizbowl.com/documents/subash-maddipotis-tips-on-question-writing/`. Accessed: 2018-12-04.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics System Demonstrations*, pages 55–60, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems*, 2013.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Proceedings of Empirical Methods in Natural Language Processing*, 2016.

Vincent Ng. Machine learning for entity coreference resolution: A retrospective look at two decades of research. In *Association for the Advancement of Artificial Intelligence*, 2017.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Kreidieh Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24:694–707, 2016.

Kayur Patel, James Fogarty, James A. Landay, and Beverly L. Harrison. Investigating statistical machine learning as a tool for software development. In *International Conference on Human Factors in Computing Systems*, 2008.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Michael Petrochuk and Luke S. Zettlemoyer. Simplequestions nearly solved: A new upper-bound and baseline approach. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Anand Rajaraman and Jeffrey David Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011. doi: 10.1017/CBO9781139058452.002.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the Association for Computational Linguistics*, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the Association for Computational Linguistics*, 2018.

Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, 1997.

Burr Settles and Brendan Meeder. A trainable spaced repetition model for language learning. In *Proceedings of the Association for Computational Linguistics*, 2016.

Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27: 443–460, 2015.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Roland Stuckardt. Coreference-based summarization and question answering: a case for high precision anaphor resolution. In *International Symposium on Reference Resolution*, 2003.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. What makes reading comprehension questions easier? In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

David Taylor, Colin McNulty, and Jo Meek. Your starter for ten: 50 years of university challenge, 2012. URL `https://www.bbc.co.uk/sounds/play/b01m49vh`.

Gerald Tesauro, David Gondek, Jonathan Lenchner, James Fan, and John M. Prager. Analysis of watson's strategies for playing jeopardy! *Journal of Artificial Intelligence Research*, 47: 205–251, 2013.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems*, 2015.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, 2017.

Jerry Vinokurov. How to write questions. `http://hsquizbowl.org/forums/viewtopic.php?f=30&t=3945`. Accessed: 2018-12-04.

Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

Eric Wallace, Pedro Rodriguez, Shi Feng, and Jordan Boyd-Graber. Trick me if you can: Adversarial writing of trivia challenge questions. 2018.

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

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics*, 5:397–411, 2017.

Ikuya Yamada, Ryuji Tamaki, Hiroyuki Shindo, and Yoshiyasu Takefuji. Studio ousia's quiz bowl question answering system. *arXiv preprint arXiv:1803.08652*, 2018.

Roman V. Yampolskiy. *Turing Test as a Defining Feature of AI-Completeness*, pages 3–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-29694-9. doi: 10.1007/978-3-642-29694-9_1.

Yi Yang, Wen tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2015.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Davis Yoshida. Domain adaptation for factoid question answering. Master's thesis, University of Colorado at Boulder, 2018. Applied Mathematics Graduate Theses & Dissertations.

Valentina Bayer Zubek and Thomas G. Dietterich. Pruning improves heuristic search for cost-sensitive learning. In *Proceedings of the International Conference of Machine Learning*, 2002.

Appendices

Appendix A. Preprocessing

This section provides a detailed description of preprocessing on the question dataset.

A.1 Aligning and De-duplicating Questions

Since we obtain machine readable versions of questions from two online sources it is necessary to ensure that we do not include the same question twice. We use the metadata associated with each question such as tournament and year. As part of our preprocessing we manually align the values of these fields.[38] We use these fields to ensure that questions for each tournament and year are included only once.

A.2 Textual Preprocessing

Models should define their own textual preprocessing so we only preprocess the text to remove Quizbowl specific artifacts. Most of these artifacts are instructions to the moderator or

38. We also align category and sub-category fields.

organizer such as "MODERATOR NOTE:", "Description Required", "15 pts:", or a reference to the category of the question; we use regular expression rules to remove these. Since we report results on accuracy after the first sentence in questions we also provide a set of canonical sentence tokenization indices computed using spacy.[39]

A.3 Fold Assignment

When we assign folds to Quizbowl questions we aim to create useful splits for guessing and buzzing while preserving the integrity of the development and test sets. Namely, when we create test and development folds we make the division into folds not depend on whether or not gameplay data exists. If it were the case that by making this unconditional assignment the number of questions with gameplay data is too small this would be a problem. We do not find this to be a problem however.

For test set questions this is easily accomplished by using an implicit quality filter and a temporal split; use only questions from national championship tournaments, questions from 2016 are used in the buzzing test set, and questions from 2017 and 2018 are used for the guessing test set. Following this we pair the test fold for buzzing with gameplay data, and are fortunate that the number of questions is not small.

To create the development sets we use questions from 2015 which are randomly split with equal probability into guessing and buzzing specific folds. Similarly to the test set we associate gameplay data after this assignment occurs to preserve its integrity against any bias that conditioning on having gameplay data would have.

For the training data we make a weaker attempt to eliminate bias in favor of ensuring that the training folds for guessing and buzzing are large enough. We first divide the training questions with an 80/20 split. Questions in the eighty percent split are assigned to the guessing fold. Each remaining question is assigned to the buzzing fold if it has gameplay data, otherwise it is assigned to the guessing fold. Figure 5 shows the result of this folding procedure.

A.4 Wikipedia Answer Matching

The automatic rule based part of this process is composed of two phases: an expansion phase in that produces variants of the answer text, and a match phase that determines when one of these variants is a match to a Wikipedia page. The rules in the expansion phase can be as simple as exact text match to expanding "The {Master of Flémalle} or Robert {Campin}" to "{Master of Flémalle}" and "Robert {Campin}". In this case, multiple matches result in "Robert Campin" being the answer page: after removing braces "The Master of Flémalle" Wikipedia redirects to "Robert Campin" and "Robert Campin" is also an exact match. When matches disagree we use the match that modified the original answer text the least.

There are inevitably cases where the automatic system fails to find a match, or finds the wrong match. Qualitatively these are often caused by disambiguation errors such as failing to differentiate between "Guernica" the city versus the painting by Picasso, small differences in answer strings, and when there is no suitable Wikipedia page. To correct or verify these errors we (the authors), and skilled members of the QB community (such as tournament

39. `https://spacy.io`

organizers and participants from our exhibition matches) manually annotated a significant fraction of the training data, and all the test data.

Rather than doing manual annotation of each question, we begin by defining mappings of answer strings to Wikipedia pages so that when that string occurs multiple times it does not require manual annotation for every occurrence of that answer in questions. However, this has the serious drawback that if the answer string is ambiguous then it may result in mislabeled answers. To avoid this problem we design a manual process whereby annotators update three sets of answer-to-Wikipedia mappings: unambiguous, ambiguous, and direct mappings.

Unambiguous annotations contain a list of answer strings that when seen map to a specific Wikipedia page. As the name implies, we only insert annotations here when the answer unambiguously identifies the corresponding Wikipedia page. Ambiguous annotations similarly contain a list of answer strings, but are paired with a list of disambiguation words. If the answer string is seen, at least one word is in the question text, and there are no other ambiguous matches, then it is mapped. For example, if the answer string is "amazon" and the question contains the word "river" then we assume "Amazon river" is the correct page while if the question mentions "bezos" then the correct page is "Amazon (company)". Finally, direct mappings match the answer for specific questions.

The last major design decision in this process addresses how we prevent information from the test data to leak into the training data. The root of the data leak issue is that the distribution of answers between training and test data often results in only approximately 80% of test set answers occurring in the training data. We observed this phenomena empirically in both our data and the distribution of answers from our numerous exhibition events. If all answer strings are naively combined, then mapped, this implies that the training data will be biased towards its answers containing an over abundance of test set answers. To avoid this problem we use direct question level mapping to fully annotate the test data.

Appendix B. TriviaQA Answer Frequency Distribution

Section 5.4 and specifically Figure 12 shows that the distribution of training examples for any given answer is heavily skewed in Quizbowl. Predictably degrades model performance on answers with scarce training data (Figure 16 in Section 7.2.2). Figure 21 shows that TriviaQA has a similarly skewed distribution in training examples per answer.

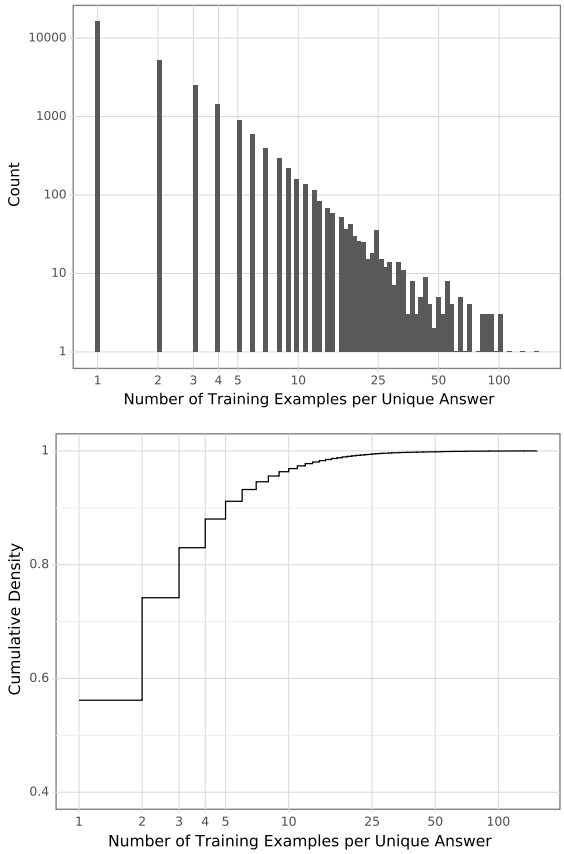


Figure 21: The figure shows that for TriviaQA—just as in Quizbowl—the distribution of training examples per unique answer is heavily skewed. Many errors in Quizbowl models are attributable to scarce training data for given answers; perhaps a similar phenomena occurs in TriviaQA models.