The AI Behind Watson — The Technical Article

The 2010 Fall Issue of *AI Magazine* includes an article on "Building Watson: An Overview of the DeepQA Project," written by the IBM Watson Research Team, led by David Ferucci. Read about this exciting project in the most detailed technical article available. We hope you will also take a moment to read through the archives of *AI Magazine,* (../issues.php) and consider joining us at AAAI. To join, please read more at http://www.aaai.org/Membership/membership.php (http://www.aaai.org/Membership/membership.php) . The most recent online volume of *AI Magazine* is usually only available to members of the association. However, we have made an exception for this special article on Watson to share the excitement. Congratulations to the IBM Watson Team!

Building Watson: An Overview of the DeepQA Project

Written by *David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty*

Abstract

IBM Research undertook a challenge to build a computer system that could compete at the human champion level in real time on the American TV quiz show, Jeopardy. The extent of the challenge includes fielding a real-time automatic contestant on the show, not merely a laboratory exercise. The Jeopardy Challenge helped us address requirements that led to the design of the DeepQA architecture and the implementation of Watson. After three years of intense research and development by a core team of about 20 researchers, Watson is performing at human expert levels in terms of precision, confidence, and speed at the Jeopardy quiz show. Our results strongly suggest that DeepQA is an effective and extensible architecture that can be used as a foundation for combining, deploying, evaluating, and advancing a wide range of algorithmic techniques to rapidly advance the field of question answering (QA).

The goals of IBM Research are to advance computer science by exploring new ways for computer technology to affect science, business, and society. Roughly three years ago, IBM Research was looking for a major research challenge to rival the scientific and popular interest of Deep Blue, the computer chess-playing champion (Hsu 2002), that also would have clear relevance to IBM business interests.

With a wealth of enterprise-critical information being captured in natural language documentation of all forms, the problems with perusing only the top 10 or 20 most popular documents containing the user's two or three key words are becoming increasingly apparent. This is especially the case in the enterprise where popularity is not as important an indicator of relevance and where recall can be as critical as precision. There is growing interest to have enterprise computer systems deeply analyze the

breadth of relevant content to more precisely answer and justify answers to user's natural language questions. We believe advances in question-answering (QA) technology can help support professionals in critical and timely decision making in areas like compliance, health care, business integrity, business intelligence, knowledge discovery, enterprise knowledge management, security, and customer support. For researchers, the open-domain QA problem is attractive as it is one of the most challenging in the realm of computer science and artificial intelligence, requiring a synthesis of information retrieval, natural language processing, knowledge representation and reasoning, machine learning, and computer-human interfaces. It has had a long history (Simmons 1970) and saw rapid advancement spurred by system building, experimentation, and government funding in the past decade (Maybury 2004, Strzalkowski and Harabagiu 2006).

With QA in mind, we settled on a challenge to build a computer system, called *Watson,*[1] which could compete at the human champion level in real time on the American TV quiz show, *Jeopardy*. The extent of the challenge includes fielding a real-time automatic contestant on the show, not merely a laboratory exercise.

Jeopardy! is a well-known TV quiz show that has been airing on television in the United States for more than 25 years (see the *Jeopardy!* Quiz Show sidebar for more information on the show). It pits three human contestants against one another in a competition that requires answering rich natural language questions over a very broad domain of topics, with penalties for wrong answers. The nature of the three-person competition is such that confidence, precision, and answering speed are of critical importance, with roughly 3 seconds to answer each question. A computer system that could compete at human champion levels at this game would need to produce exact answers to often complex natural language questions with high precision and speed and have a reliable confidence in its answers, such that it could answer roughly 70 percent of the questions asked with greater than 80 percent precision in 3 seconds or less.

Finally, the *Jeopardy* Challenge represents a unique and compelling AI question similar to the one underlying DeepBlue (Hsu 2002) — can a computer system be designed to compete against the best humans at a task thought to require high levels of human intelligence, and if so, what kind of technology, algorithms, and engineering is required? While we believe the *Jeopardy* Challenge is an extraordinarily demanding task that will greatly advance the field, we appreciate that this challenge alone does not address all aspects of QA and does not by any means close the book on the QA challenge the way that Deep Blue may have for playing chess.

The *Jeopardy* Challenge

Meeting the *Jeopardy* Challenge requires advancing and incorporating a variety of QA technologies including parsing, question classification, question decomposition, automatic source acquisition and evaluation, entity and relation detection, logical form generation, and knowledge representation and reasoning.

Winning at *Jeopardy* requires accurately computing confidence in your answers. The questions and content are ambiguous and noisy and none of the individual algorithms are perfect. Therefore, each component must produce a confidence in its output, and individual component confidences must be combined to compute the overall confidence of the final answer. The final confidence is used to

determine whether the computer system should risk choosing to answer at all. In *Jeopardy* parlance, this confidence is used to determine whether the computer will "ring in" or "buzz in" for a question. The confidence must be computed during the time the question is read and before the opportunity to buzz in. This is roughly between 1 and 6 seconds with an average around 3 seconds.

Confidence estimation was very critical to shaping our overall approach in DeepQA. There is no expectation that any component in the system does a perfect job — all components post features of the computation and associated confidences, and we use a hierarchical machine-learning method to combine all these features and decide whether or not there is enough confidence in the final answer to attempt to buzz in and risk getting the question wrong.

In this section we elaborate on the various aspects of the *Jeopardy* Challenge.

The Categories

A 30-clue *Jeopardy* board is organized into six columns. Each column contains five clues and is associated with a category. Categories range from broad subject headings like "history," "science," or "politics" to less informative puns like "tutu much," in which the clues are about ballet, to actual parts of the clue, like "who appointed me to the Supreme Court?" where the clue is the name of a judge, to "anything goes" categories like "potpourri." Clearly some categories are essential to understanding the clue, some are helpful but not necessary, and some may be useless, if not misleading, for a computer.

A recurring theme in our approach is the requirement to try many alternate hypotheses in varying contexts to see which produces the most confident answers given a broad range of loosely coupled scoring algorithms. Leveraging category information is another clear area requiring this approach.

The Questions

There are a wide variety of ways one can attempt to characterize the *Jeopardy* clues. For example, by topic, by difficulty, by grammatical construction, by answer type, and so on. A type of classification that turned out to be useful for us was based on the primary method deployed to solve the clue. The bulk of *Jeopardy* clues represent what we would consider factoid questions — questions whose answers are based on factual information about one or more individual entities. The questions themselves present challenges in determining what exactly is being asked for and which elements of the clue are relevant in determining the answer. Here are just a few examples (note that while the *Jeopardy!* game requires that answers are delivered in the form of a question (see the *Jeopardy!* Quiz Show sidebar), this transformation is trivial and for purposes of this paper we will just show the answers themselves):

Category: General Science
Clue: When hit by electrons, a phosphor gives off electromagnetic energy in this form.
Answer: Light (or Photons)

Category: Lincoln Blogs
Clue: Secretary Chase just submitted this to me for the third time; guess what, pal. This time I'm accepting it.
Answer: his resignation

Category: Head North
Clue: They're the two states you could be reentering if you're crossing Florida's northern border.
Answer: Georgia and Alabama

Decomposition.

Some more complex clues contain multiple facts about the answer, all of which are required to arrive at the correct response but are unlikely to occur together in one place. For example:

Category: "Rap" Sheet
Clue: This archaic term for a mischievous or annoying child can also mean a rogue or scamp.
Subclue 1: This archaic term for a mischievous or annoying child.
Subclue 2: This term can also mean a rogue or scamp.
Answer: Rapscallion

In this case, we would not expect to find both "subclues" in one sentence in our sources; rather, if we decompose the question into these two parts and ask for answers to each one, we may find that the answer common to both questions is the answer to the original clue.

Another class of decomposable questions is one in which a subclue is nested in the outer clue, and the subclue can be replaced with its answer to form a new question that can more easily be answered. For example:

Category: Diplomatic Relations
Clue: Of the four countries in the world that the United States does not have diplomatic relations with, the one that's farthest north.
Inner subclue: The four countries in the world that the United States does not have diplomatic relations with (Bhutan, Cuba, Iran, North Korea).
Outer subclue: Of Bhutan, Cuba, Iran, and North Korea, the one that's farthest north.
Answer: North Korea

Decomposable *Jeopardy* clues generated requirements that drove the design of DeepQA to generate zero or more decomposition hypotheses for each question as possible interpretations.

Puzzles.

Jeopardy also has categories of questions that require special processing defined by the category itself. Some of them recur often enough that contestants know what they mean without instruction; for others, part of the task is to figure out what the puzzle is as the clues and answers are revealed (categories requiring explanation by the host are not part of the challenge). Examples of well-known puzzle categories are the Before and After category, where two subclues have answers that overlap by (typically) one word, and the Rhyme Time category, where the two subclue answers must rhyme with one another. Clearly these cases also require question decomposition. For example:

Category: Before and After Goes to the Movies
Clue: Film of a typical day in the life of the Beatles, which includes running from bloodthirsty zombie fans in a Romero classic.

Subclue 2: Film of a typical day in the life of the Beatles.
Answer 1: (*A Hard Day's Night*)
Subclue 2: Running from bloodthirsty zombie fans in a Romero classic.
Answer 2: (*Night of the Living Dead*)
Answer: A Hard Day's Night of the Living Dead

Category: Rhyme Time
Clue: It's where Pele stores his ball.
Subclue 1: Pele ball (soccer)
Subclue 2: where store (cabinet, drawer, locker, and so on)
Answer: soccer locker

There are many infrequent types of puzzle categories including things like converting roman numerals, solving math word problems, sounds like, finding which word in a set has the highest Scrabble score, homonyms and heteronyms, and so on. Puzzles constitute only about 2–3 percent of all clues, but since they typically occur as entire categories (five at a time) they cannot be ignored for success in the Challenge as getting them all wrong often means losing a game.

Excluded Question Types.

The *Jeopardy* quiz show ordinarily admits two kinds of questions that IBM and Jeopardy Productions, Inc., agreed to exclude from the computer contest: audiovisual (A/V) questions and Special Instructions questions. A/V questions require listening to or watching some sort of audio, image, or video segment to determine a correct answer. For example:

Category: Picture This
(Contestants are shown a picture of a B-52 bomber)
Clue: Alphanumeric name of the fearsome machine seen here.
Answer: B-52

Special instruction questions are those that are not "self-explanatory" but rather require a verbal explanation describing how the question should be interpreted and solved. For example:

Category: Decode the Postal Codes
Verbal instruction from host: We're going to give you a word comprising two postal abbreviations; you have to identify the states.
Clue: Vain
Answer: Virginia and Indiana

Both present very interesting challenges from an AI perspective but were put out of scope for this contest and evaluation.

The Domain

As a measure of the *Jeopardy* Challenge's breadth of domain, we analyzed a random sample of 20,000 questions extracting the lexical answer type (LAT) when present. We define a LAT to be a word in the clue that indicates the type of the answer, independent of assigning semantics to that

word. For example in the following clue, the LAT is the string “maneuver.”

Category: Oooh….Chess
Clue: Invented in the 1500s to speed up the game, this maneuver involves two pieces of the same color.
7 *Answer:* Castling

About 12 percent of the clues do not indicate an explicit lexical answer type but may refer to the answer with pronouns like “it,” “these,” or “this” or not refer to it at all. In these cases the type of answer must be inferred by the context. Here’s an example:

Category: Decorating
Clue: Though it sounds “harsh,” it’s just embroidery, often in a floral pattern, done with yarn on cotton cloth.
Answer: crewel

The distribution of LATs has a very long tail, as shown in figure 1. We found 2500 distinct and explicit LATs in the 20,000 question sample. The most frequent 200 explicit LATs cover less than 50 percent of the data. Figure 1 shows the relative frequency of the LATs. It labels all the clues with no explicit type with the label “NA.” This aspect of the challenge implies that while task-specific type systems or manually curated data would have some impact if focused on the head of the LAT curve, it still leaves more than half the problems unaccounted for. Our clear technical bias for both business and scientific motivations is to create general-purpose, reusable natural language processing (NLP) and knowledge representation and reasoning (KRR) technology that can exploit as-is natural language resources and as-is structured knowledge rather than to curate task-specific knowledge resources.

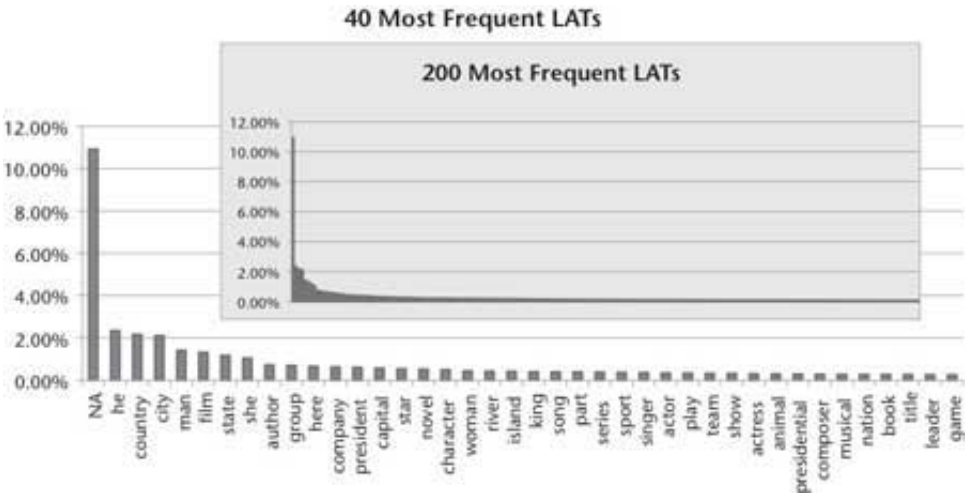

Figure 1. Lexical Answer Type Frequency.

The Metrics

In addition to question-answering precision, the system’s game-winning performance will depend on speed, confidence estimation, clue selection, and betting strategy. Ultimately the outcome of the public contest will be decided based on whether or not Watson can win one or two games against top-ranked

humans in real time. The highest amount of money earned by the end of a one- or two-game match determines the winner. A player's final earnings, however, often will not reflect how well the player did during the game at the QA task. This is because a player may decide to bet big on Daily Double or Final *Jeopardy* questions. There are three hidden Daily Double questions in a game that can affect only the player lucky enough to find them, and one Final *Jeopardy* question at the end that all players must gamble on. Daily Double and Final *Jeopardy* questions represent significant events where players may risk all their current earnings. While potentially compelling for a public contest, a small number of games does not represent statistically meaningful results for the system's raw QA performance.

While Watson is equipped with betting strategies necessary for playing full *Jeopardy*, from a core QA perspective we want to measure correctness, confidence, and speed, without considering clue selection, luck of the draw, and betting strategies. We measure correctness and confidence using precision and percent answered. Precision measures the percentage of questions the system gets right out of those it chooses to answer. Percent answered is the percentage of questions it chooses to answer (correctly or incorrectly). The system chooses which questions to answer based on an estimated confidence score: for a given threshold, the system will answer all questions with confidence scores above that threshold. The threshold controls the trade-off between precision and percent answered, assuming reasonable confidence estimation. For higher thresholds the system will be more conservative, answering fewer questions with higher precision. For lower thresholds, it will be more aggressive, answering more questions with lower precision. Accuracy refers to the precision if all questions are answered.

Figure 2 shows a plot of precision versus percent attempted curves for two theoretical systems. It is obtained by evaluating the two systems over a range of confidence thresholds. Both systems have 40 percent accuracy, meaning they get 40 percent of all questions correct. They differ only in their confidence estimation. The upper line represents an ideal system with perfect confidence estimation. Such a system would identify exactly which questions it gets right and wrong and give higher confidence to those it got right. As can be seen in the graph, if such a system were to answer the 50 percent of questions it had highest confidence for, it would get 80 percent of those correct. We refer to this level of performance as 80 percent precision at 50 percent answered. The lower line represents a system without meaningful confidence estimation. Since it cannot distinguish between which questions it is more or less likely to get correct, its precision is constant for all percent attempted. Developing more accurate confidence estimation means a system can deliver far higher precision even with the same overall accuracy.

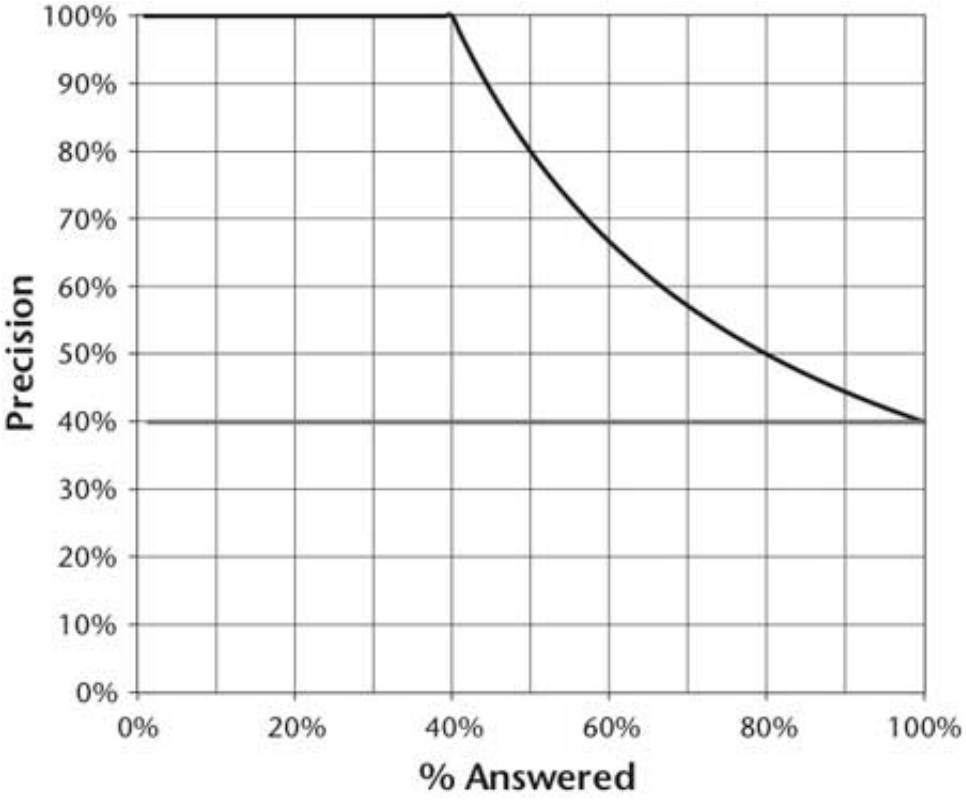

Figure 2. Precision Versus Percentage Attempted.
Perfect confidence estimation (upper line) and no confidence estimation (lower line).

The Competition: Human Champion Performance

A compelling and scientifically appealing aspect of the *Jeopardy* Challenge is the human reference point. Figure 3 contains a graph that illustrates expert human performance on *Jeopardy* It is based on our analysis of nearly 2000 historical *Jeopardy* games. Each point on the graph represents the performance of the winner in one *Jeopardy* game.[2] As in figure 2, the *x*-axis of the graph, labeled "% Answered," represents the percentage of questions the winner answered, and the *y*-axis of the graph, labeled "Precision," represents the percentage of those questions the winner answered correctly.

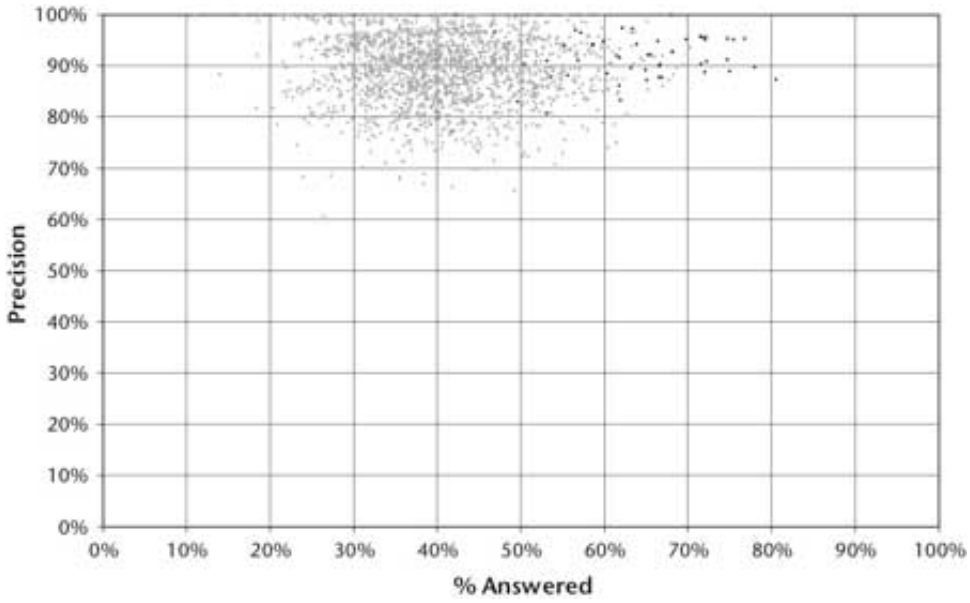

Figure 3. Champion Human Performance at Jeopardy.

In contrast to the system evaluation shown in figure 2, which can display a curve over a range of confidence thresholds, the human performance shows only a single point per game based on the observed precision and percent answered the winner demonstrated in the game. A further distinction is that in these historical games the human contestants did not have the liberty to answer all questions they wished. Rather the percent answered consists of those questions for which the winner was confident and fast enough to beat the competition to the buzz. The system performance graphs shown in this paper are focused on evaluating QA performance, and so do not take into account competition for the buzz. Human performance helps to position our system's performance, but obviously, in a *Jeopardy* game, performance will be affected by competition for the buzz and this will depend in large part on how quickly a player can compute an accurate confidence and how the player manages risk.

The center of what we call the "Winners Cloud" (the set of light gray dots in the graph in figures 3 and 4) reveals that *Jeopardy* champions are confident and fast enough to acquire on average between 40 percent and 50 percent of all the questions from their competitors and to perform with between 85 percent and 95 percent precision.

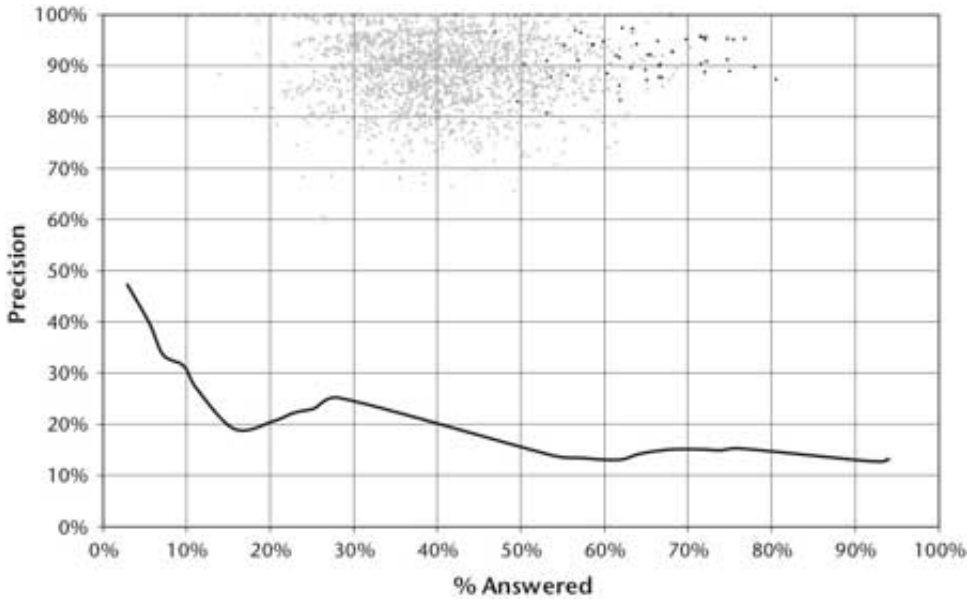

Figure 4. Baseline Performance.

The darker dots on the graph represent Ken Jennings's games. Ken Jennings had an unequaled winning streak in 2004, in which he won 74 games in a row. Based on our analysis of those games, he acquired on average 62 percent of the questions and answered with 92 percent precision. Human performance at this task sets a very high bar for precision, confidence, speed, and breadth.

Baseline Performance

Our metrics and baselines are intended to give us confidence that new methods and algorithms are improving the system or to inform us when they are not so that we can adjust research priorities.

Our most obvious baseline is the QA system called Practical Intelligent Question Answering Technology (PIQUANT) (Prager, Chu-Carroll, and Czuba 2004), which had been under development at IBM Research by a four-person team for 6 years prior to taking on the *Jeopardy* Challenge. At the time it was among the top three to five Text Retrieval Conference (TREC) QA systems. Developed in part under the U.S. government AQUAINT program[3] and in collaboration with external teams and universities, PIQUANT was a classic QA pipeline with state-of-the-art techniques aimed largely at the TREC QA evaluation (Voorhees and Dang 2005). PIQUANT performed in the 33 percent accuracy range in TREC evaluations. While the TREC QA evaluation allowed the use of the web, PIQUANT focused on question answering using local resources. A requirement of the *Jeopardy* Challenge is that the system be self-contained and does not link to live web search.

The requirements of the TREC QA evaluation were different than for the *Jeopardy* challenge. Most notably, TREC participants were given a relatively small corpus (1M documents) from which answers to questions must be justified; TREC questions were in a much simpler form compared to *Jeopardy* questions, and the confidences associated with answers were not a primary metric. Furthermore, the systems are allowed to access the web and had a week to produce results for 500 questions. The reader can find details in the TREC proceedings[4] and numerous follow-on publications.

An initial 4-week effort was made to adapt PIQUANT to the *Jeopardy* Challenge. The experiment

focused on precision and confidence. It ignored issues of answering speed and aspects of the game like betting and clue values.

The questions used were 500 randomly sampled *Jeopardy* clues from episodes in the past 15 years. The corpus that was used contained, but did not necessarily justify, answers to more than 90 percent of the questions. The result of the PIQUANT baseline experiment is illustrated in figure 4. As shown, on the 5 percent of the clues that PIQUANT was most confident in (left end of the curve), it delivered 47 percent precision, and over all the clues in the set (right end of the curve), its precision was 13 percent. Clearly the precision and confidence estimation are far below the requirements of the *Jeopardy* Challenge.

A similar baseline experiment was performed in collaboration with Carnegie Mellon University (CMU) using OpenEphyra,[5] an open-source QA framework developed primarily at CMU. The framework is based on the Ephyra system, which was designed for answering TREC questions. In our experiments on TREC 2002 data, OpenEphyra answered 45 percent of the questions correctly using a live web search.

We spent minimal effort adapting OpenEphyra, but like PIQUANT, its performance on *Jeopardy* clues was below 15 percent accuracy. OpenEphyra did not produce reliable confidence estimates and thus could not effectively choose to answer questions with higher confidence. Clearly a larger investment in tuning and adapting these baseline systems to *Jeopardy* would improve their performance; however, we limited this investment since we did not want the baseline systems to become significant efforts.

The PIQUANT and OpenEphyra baselines demonstrate the performance of state-of-the-art QA systems on the *Jeopardy* task. In figure 5 we show two other baselines that demonstrate the performance of two complementary approaches on this task. The light gray line shows the performance of a system based purely on text search, using terms in the question as queries and search engine scores as confidences for candidate answers generated from retrieved document titles. The black line shows the performance of a system based on structured data, which attempts to look the answer up in a database by simply finding the named entities in the database related to the named entities in the clue. These two approaches were adapted to the *Jeopardy* task, including identifying and integrating relevant content.

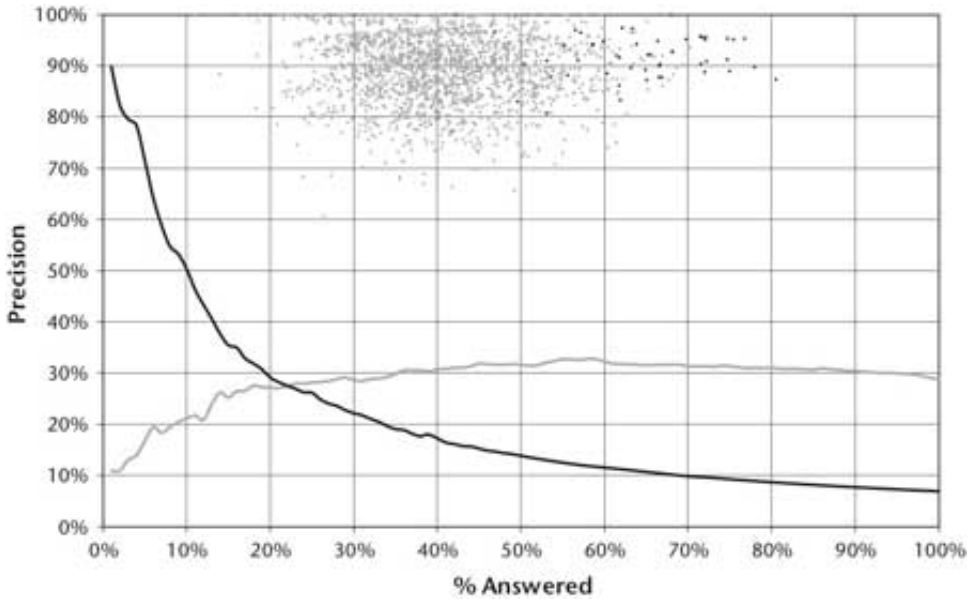

Figure 5. Text Search Versus Knowledge Base Search.

The results form an interesting comparison. The search-based system has better performance at 100 percent answered, suggesting that the natural language content and the shallow text search techniques delivered better coverage. However, the flatness of the curve indicates the lack of accurate confidence estimation.[6] The structured approach had better informed confidence when it was able to decipher the entities in the question and found the right matches in its structured knowledge bases, but its coverage quickly drops off when asked to answer more questions. To be a high-performing question-answering system, DeepQA must demonstrate both these properties to achieve high precision, high recall, and an accurate confidence estimation.

The DeepQA Approach

Early on in the project, attempts to adapt PIQUANT (Chu-Carroll et al. 2003) failed to produce promising results. We devoted many months of effort to encoding algorithms from the literature. Our investigations ran the gamut from deep logical form analysis to shallow machine-translation-based approaches. We integrated them into the standard QA pipeline that went from question analysis and answer type determination to search and then answer selection. It was difficult, however, to find examples of how published research results could be taken out of their original context and effectively replicated and integrated into different end-to-end systems to produce comparable results. Our efforts failed to have significant impact on *Jeopardy* or even on prior baseline studies using TREC data.

We ended up overhauling nearly everything we did, including our basic technical approach, the underlying architecture, metrics, evaluation protocols, engineering practices, and even how we worked together as a team. We also, in cooperation with CMU, began the Open Advancement of Question Answering (OAQA) initiative. OAQA is intended to directly engage researchers in the community to help replicate and reuse research results and to identify how to more rapidly advance the state of the art in QA (Ferrucci et al 2009).

As our results dramatically improved, we observed that system-level advances allowing rapid integration and evaluation of new ideas and new components against end-to-end metrics were

essential to our progress. This was echoed at the OAQA workshop for experts with decades of investment in QA, hosted by IBM in early 2008. Among the workshop conclusions was that QA would benefit from the collaborative evolution of a single extensible architecture that would allow component results to be consistently evaluated in a common technical context against a growing variety of what were called "Challenge Problems." Different challenge problems were identified to address various dimensions of the general QA problem. *Jeopardy* was described as one addressing dimensions including high precision, accurate confidence determination, complex language, breadth of domain, and speed.

The system we have built and are continuing to develop, called DeepQA, is a massively parallel probabilistic evidence-based architecture. For the *Jeopardy* Challenge, we use more than 100 different techniques for analyzing natural language, identifying sources, finding and generating hypotheses, finding and scoring evidence, and merging and ranking hypotheses. What is far more important than any particular technique we use is how we combine them in DeepQA such that overlapping approaches can bring their strengths to bear and contribute to improvements in accuracy, confidence, or speed.

DeepQA is an architecture with an accompanying methodology, but it is not specific to the *Jeopardy* Challenge. We have successfully applied DeepQA to both the *Jeopardy* and TREC QA task. We have begun adapting it to different business applications and additional exploratory challenge problems including medicine, enterprise search, and gaming.

The overarching principles in DeepQA are massive parallelism, many experts, pervasive confidence estimation, and integration of shallow and deep knowledge.

Massive parallelism: Exploit massive parallelism in the consideration of multiple interpretations and hypotheses.

Many experts: Facilitate the integration, application, and contextual evaluation of a wide range of loosely coupled probabilistic question and content analytics.

Pervasive confidence estimation: No component commits to an answer; all components produce features and associated confidences, scoring different question and content interpretations. An underlying confidence-processing substrate learns how to stack and combine the scores.

Integrate shallow and deep knowledge: Balance the use of strict semantics and shallow semantics, leveraging many loosely formed ontologies.

Figure 6 illustrates the DeepQA architecture at a very high level. The remaining parts of this section provide a bit more detail about the various architectural roles.

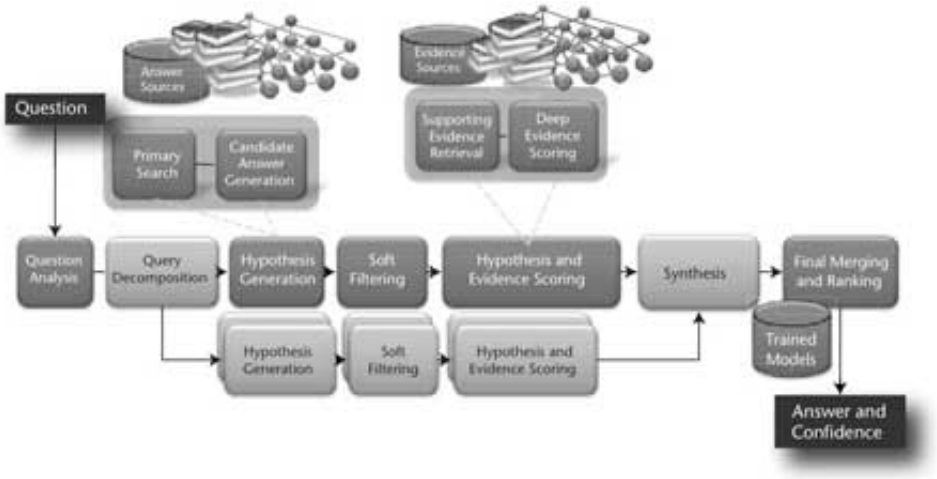

Figure 6. DeepQA High-Level Architecture.

Content Acquisition

The first step in any application of DeepQA to solve a QA problem is content acquisition, or identifying and gathering the content to use for the answer and evidence sources shown in figure 6.

Content acquisition is a combination of manual and automatic steps. The first step is to analyze example questions from the problem space to produce a description of the kinds of questions that must be answered and a characterization of the application domain. Analyzing example questions is primarily a manual task, while domain analysis may be informed by automatic or statistical analyses, such as the LAT analysis shown in figure 1. Given the kinds of questions and broad domain of the *Jeopardy* Challenge, the sources for Watson include a wide range of encyclopedias, dictionaries, thesauri, newswire articles, literary works, and so on.

Given a reasonable baseline corpus, DeepQA then applies an automatic corpus expansion process. The process involves four high-level steps: (1) identify seed documents and retrieve related documents from the web; (2) extract self-contained text nuggets from the related web documents; (3) score the nuggets based on whether they are informative with respect to the original seed document; and (4) merge the most informative nuggets into the expanded corpus. The live system itself uses this expanded corpus and does not have access to the web during play.

In addition to the content for the answer and evidence sources, DeepQA leverages other kinds of semistructured and structured content. Another step in the content-acquisition process is to identify and collect these resources, which include databases, taxonomies, and ontologies, such as dbPedia,[7] WordNet (Miller 1995), and the Yago[8] ontology.

Question Analysis

The first step in the run-time question-answering process is question analysis. During question analysis the system attempts to understand what the question is asking and performs the initial analyses that determine how the question will be processed by the rest of the system. The DeepQA approach encourages a mixture of experts at this stage, and in the Watson system we produce shallow

parses, deep parses (McCord 1990), logical forms, semantic role labels, coreference, relations, named entities, and so on, as well as specific kinds of analysis for question answering. Most of these technologies are well understood and are not discussed here, but a few require some elaboration.

Question Classification.

Question classification is the task of identifying question types or parts of questions that require special processing. This can include anything from single words with potentially double meanings to entire clauses that have certain syntactic, semantic, or rhetorical functionality that may inform downstream components with their analysis. Question classification may identify a question as a puzzle question, a math question, a definition question, and so on. It will identify puns, constraints, definition components, or entire subclues within questions.

Focus and LAT Detection.

As discussed earlier, a lexical answer type is a word or noun phrase in the question that specifies the type of the answer without any attempt to understand its semantics. Determining whether or not a candidate answer can be considered an instance of the LAT is an important kind of scoring and a common source of critical errors. An advantage to the DeepQA approach is to exploit many independently developed answer-typing algorithms. However, many of these algorithms are dependent on their own type systems. We found the best way to integrate preexisting components is not to force them into a single, common type system, but to have them map from the LAT to their own internal types.

The focus of the question is the part of the question that, if replaced by the answer, makes the question a stand-alone statement. Looking back at some of the examples shown previously, the focus of "When hit by electrons, a phosphor gives off electromagnetic energy in this form" is "this form"; the focus of "Secretary Chase just submitted this to me for the third time; guess what, pal. This time I'm accepting it" is the first "this"; and the focus of "This title character was the crusty and tough city editor of the *Los Angeles Tribune*" is "This title character." The focus often (but not always) contains useful information about the answer, is often the subject or object of a relation in the clue, and can turn a question into a factual statement when replaced with a candidate, which is a useful way to gather evidence about a candidate.

Relation Detection.

Most questions contain relations, whether they are syntactic subject-verb-object predicates or semantic relationships between entities. For example, in the question, "They're the two states you could be reentering if you're crossing Florida's northern border," we can detect the relation borders(Florida,?x,north).

Watson uses relation detection throughout the QA process, from focus and LAT determination, to passage and answer scoring. Watson can also use detected relations to query a triple store and directly generate candidate answers. Due to the breadth of relations in the *Jeopardy* domain and the variety of ways in which they are expressed, however, Watson's current ability to effectively use curated databases to simply "look up" the answers is limited to fewer than 2 percent of the clues.

Watson's use of existing databases depends on the ability to analyze the question and detect the relations covered by the databases. In *Jeopardy* the broad domain makes it difficult to identify the most lucrative relations to detect. In 20,000 *Jeopardy* questions, for example, we found the distribution of Freebase[9] relations to be extremely flat (figure 7). Roughly speaking, even achieving high recall on detecting the most frequent relations in the domain can at best help in about 25 percent of the questions, and the benefit of relation detection drops off fast with the less frequent relations. Broad-domain relation detection remains a major open area of research.

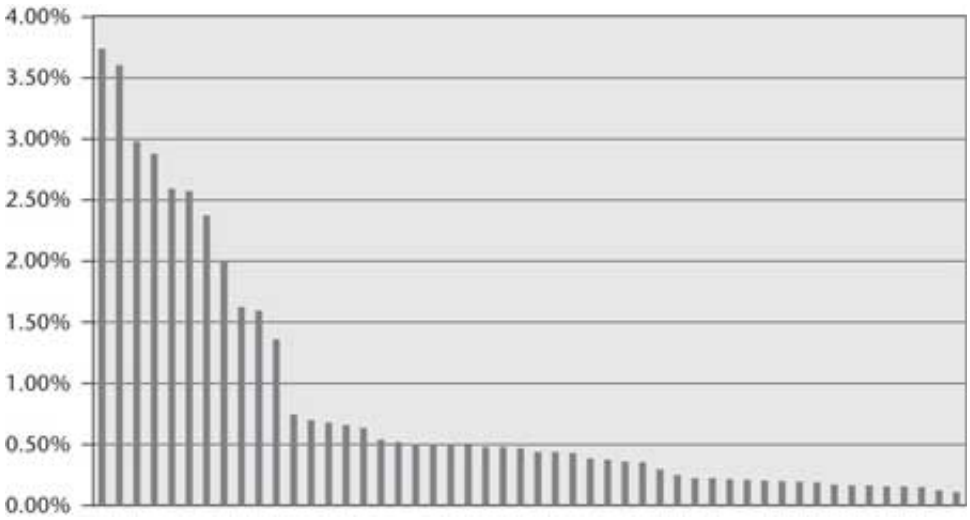

Figure 7. Approximate Distribution of the 50 Most Frequently Occurring Freebase Relations in 20,000 Randomly Selected Jeopardy Clues.

Decomposition.

As discussed above, an important requirement driven by analysis of *Jeopardy* clues was the ability to handle questions that are better answered through decomposition. DeepQA uses rule-based deep parsing and statistical classification methods both to recognize whether questions should be decomposed and to determine how best to break them up into subquestions. The operating hypothesis is that the correct question interpretation and derived answer(s) will score higher after all the collected evidence and all the relevant algorithms have been considered. Even if the question did not need to be decomposed to determine an answer, this method can help improve the system's overall answer confidence.

DeepQA solves parallel decomposable questions through application of the end-to-end QA system on each subclue and synthesizes the final answers by a customizable answer combination component. These processing paths are shown in medium gray in figure 6. DeepQA also supports nested decomposable questions through recursive application of the end-to-end QA system to the inner subclue and then to the outer subclue. The customizable synthesis components allow specialized synthesis algorithms to be easily plugged into a common framework.

Hypothesis Generation

Hypothesis generation takes the results of question analysis and produces candidate answers by searching the system's sources and extracting answer-sized snippets from the search results. Each candidate answer plugged back into the question is considered a hypothesis, which the system has to prove correct with some degree of confidence.

We refer to search performed in hypothesis generation as "primary search" to distinguish it from search performed during evidence gathering (described below). As with all aspects of DeepQA, we use a mixture of different approaches for primary search and candidate generation in the Watson system.

Primary Search.

In primary search the goal is to find as much potentially answer-bearing content as possible based on the results of question analysis — the focus is squarely on recall with the expectation that the host of deeper content analytics will extract answer candidates and score this content plus whatever evidence can be found in support or refutation of candidates to drive up the precision. Over the course of the project we continued to conduct empirical studies designed to balance speed, recall, and precision. These studies allowed us to regularly tune the system to find the number of search results and candidates that produced the best balance of accuracy and computational resources. The operative goal for primary search eventually stabilized at about 85 percent binary recall for the top 250 candidates; that is, the system generates the correct answer as a candidate answer for 85 percent of the questions somewhere within the top 250 ranked candidates.

A variety of search techniques are used, including the use of multiple text search engines with different underlying approaches (for example, Indri and Lucene), document search as well as passage search, knowledge base search using SPARQL on triple stores, the generation of multiple search queries for a single question, and backfilling hit lists to satisfy key constraints identified in the question.

Triple store queries in primary search are based on named entities in the clue; for example, find all database entities related to the clue entities, or based on more focused queries in the cases that a semantic relation was detected. For a small number of LATs we identified as "closed LATs," the candidate answer can be generated from a fixed list in some store of known instances of the LAT, such as "U.S. President" or "Country."

Candidate Answer Generation.

The search results feed into candidate generation, where techniques appropriate to the kind of search results are applied to generate candidate answers. For document search results from "title-oriented" resources, the title is extracted as a candidate answer. The system may generate a number of candidate answer variants from the same title based on substring analysis or link analysis (if the underlying source contains hyperlinks). Passage search results require more detailed analysis of the passage text to identify candidate answers. For example, named entity detection may be used to extract candidate answers from the passage. Some sources, such as a triple store and reverse dictionary lookup, produce candidate answers directly as their search result.

If the correct answer(s) are not generated at this stage as a candidate, the system has no hope of

answering the question. This step therefore significantly favors recall over precision, with the expectation that the rest of the processing pipeline will tease out the correct answer, even if the set of candidates is quite large. One of the goals of the system design, therefore, is to tolerate noise in the early stages of the pipeline and drive up precision downstream.

Watson generates several hundred candidate answers at this stage.

Soft Filtering

A key step in managing the resource versus precision trade-off is the application of lightweight (less resource intensive) scoring algorithms to a larger set of initial candidates to prune them down to a smaller set of candidates before the more intensive scoring components see them. For example, a lightweight scorer may compute the likelihood of a candidate answer being an instance of the LAT. We call this step soft filtering.

The system combines these lightweight analysis scores into a soft filtering score. Candidate answers that pass the soft filtering threshold proceed to hypothesis and evidence scoring, while those candidates that do not pass the filtering threshold are routed directly to the final merging stage. The soft filtering scoring model and filtering threshold are determined based on machine learning over training data.

Watson currently lets roughly 100 candidates pass the soft filter, but this a parameterizable function.

Hypothesis and Evidence Scoring

Candidate answers that pass the soft filtering threshold undergo a rigorous evaluation process that involves gathering additional supporting evidence for each candidate answer, or hypothesis, and applying a wide variety of deep scoring analytics to evaluate the supporting evidence.

Evidence Retrieval.

To better evaluate each candidate answer that passes the soft filter, the system gathers additional supporting evidence. The architecture supports the integration of a variety of evidence-gathering techniques. One particularly effective technique is passage search where the candidate answer is added as a required term to the primary search query derived from the question. This will retrieve passages that contain the candidate answer used in the context of the original question terms. Supporting evidence may also come from other sources like triple stores. The retrieved supporting evidence is routed to the deep evidence scoring components, which evaluate the candidate answer in the context of the supporting evidence.

Scoring.

The scoring step is where the bulk of the deep content analysis is performed. Scoring algorithms determine the degree of certainty that retrieved evidence supports the candidate answers. The DeepQA framework supports and encourages the inclusion of many different components, or scorers, that consider different dimensions of the evidence and produce a score that corresponds to how well evidence supports a candidate answer for a given question.

DeepQA provides a common format for the scorers to register hypotheses (for example candidate answers) and confidence scores, while imposing few restrictions on the semantics of the scores themselves; this enables DeepQA developers to rapidly deploy, mix, and tune components to support each other. For example, Watson employs more than 50 scoring components that produce scores ranging from formal probabilities to counts to categorical features, based on evidence from different types of sources including unstructured text, semistructured text, and triple stores. These scorers consider things like the degree of match between a passage's predicate-argument structure and the question, passage source reliability, geospatial location, temporal relationships, taxonomic classification, the lexical and semantic relations the candidate is known to participate in, the candidate's correlation with question terms, its popularity (or obscurity), its aliases, and so on.

Consider the question, "He was presidentially pardoned on September 8, 1974"; the correct answer, "Nixon," is one of the generated candidates. One of the retrieved passages is "Ford pardoned Nixon on Sept. 8, 1974." One passage scorer counts the number of IDF-weighted terms in common between the question and the passage. Another passage scorer based on the Smith-Waterman sequence-matching algorithm (Smith and Waterman 1981), measures the lengths of the longest similar subsequences between the question and passage (for example "on Sept. 8, 1974"). A third type of passage scoring measures the alignment of the logical forms of the question and passage. A logical form is a graphical abstraction of text in which nodes are terms in the text and edges represent either grammatical relationships (for example, Hermjakob, Hovy, and Lin [2000]; Moldovan et al. [2003]), deep semantic relationships (for example, Lenat [1995], Paritosh and Forbus [2005]), or both . The logical form alignment identifies Nixon as the object of the pardoning in the passage, and that the question is asking for the object of a pardoning. Logical form alignment gives "Nixon" a good score given this evidence. In contrast, a candidate answer like "Ford" would receive near identical scores to "Nixon" for term matching and passage alignment with this passage, but would receive a lower logical form alignment score.

Another type of scorer uses knowledge in triple stores, simple reasoning such as subsumption and disjointness in type taxonomies, geospatial, and temporal reasoning. Geospatial reasoning is used in Watson to detect the presence or absence of spatial relations such as directionality, borders, and containment between geoentities. For example, if a question asks for an Asian city, then spatial containment provides evidence that Beijing is a suitable candidate, whereas Sydney is not. Similarly, geocoordinate information associated with entities is used to compute relative directionality (for example, California is SW of Montana; GW Bridge is N of Lincoln Tunnel, and so on).

Temporal reasoning is used in Watson to detect inconsistencies between dates in the clue and those associated with a candidate answer. For example, the two most likely candidate answers generated by the system for the clue, "In 1594 he took a job as a tax collector in Andalusia," are "Thoreau" and "Cervantes." In this case, temporal reasoning is used to rule out Thoreau as he was not alive in 1594, having been born in 1817, whereas Cervantes, the correct answer, was born in 1547 and died in 1616.

Each of the scorers implemented in Watson, how they work, how they interact, and their independent impact on Watson's performance deserves its own research paper. We cannot do this work justice here. It is important to note, however, at this point no one algorithm dominates. In fact we believe DeepQA's facility for absorbing these algorithms, and the tools we have created for exploring their interactions and effects, will represent an important and lasting contribution of this work.

To help developers and users get a sense of how Watson uses evidence to decide between competing candidate answers, scores are combined into an overall evidence profile. The evidence profile groups individual features into aggregate evidence dimensions that provide a more intuitive view of the feature group. Aggregate evidence dimensions might include, for example, Taxonomic, Geospatial (location), Temporal, Source Reliability, Gender, Name Consistency, Relational, Passage Support, Theory Consistency, and so on. Each aggregate dimension is a combination of related feature scores produced by the specific algorithms that fired on the gathered evidence.

Consider the following question: Chile shares its longest land border with this country. In figure 8 we see a comparison of the evidence profiles for two candidate answers produced by the system for this question: Argentina and Bolivia. Simple search engine scores favor Bolivia as an answer, due to a popular border dispute that was frequently reported in the news. Watson prefers Argentina (the correct answer) over Bolivia, and the evidence profile shows why. Although Bolivia does have strong popularity scores, Argentina has strong support in the geospatial, passage support (for example, alignment and logical form graph matching of various text passages), and source reliability dimensions.

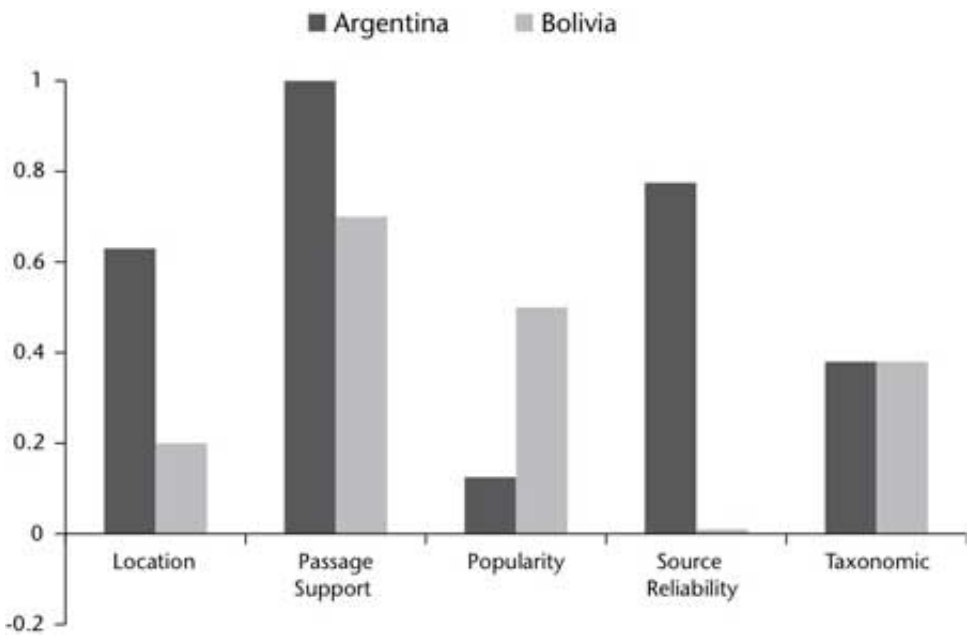

Figure 8. Evidence Profiles for Two Candidate Answers.
Dimensions are on the x-axis and relative strength is on the y-axis.

Final Merging and Ranking

It is one thing to return documents that contain key words from the question. It is quite another, however, to analyze the question and the content enough to identify the precise answer and yet another to determine an accurate enough confidence in its correctness to bet on it. Winning at *Jeopardy* requires exactly that ability.

The goal of final ranking and merging is to evaluate the hundreds of hypotheses based on potentially hundreds of thousands of scores to identify the single best-supported hypothesis given the evidence

and to estimate its confidence — the likelihood it is correct.

Answer Merging

Multiple candidate answers for a question may be equivalent despite very different surface forms. This is particularly confusing to ranking techniques that make use of relative differences between candidates. Without merging, ranking algorithms would be comparing multiple surface forms that represent the same answer and trying to discriminate among them. While one line of research has been proposed based on boosting confidence in similar candidates (Ko, Nyberg, and Luo 2007), our approach is inspired by the observation that different surface forms are often disparately supported in the evidence and result in radically different, though potentially complementary, scores. This motivates an approach that merges answer scores before ranking and confidence estimation. Using an ensemble of matching, normalization, and coreference resolution algorithms, Watson identifies equivalent and related hypotheses (for example, Abraham Lincoln and Honest Abe) and then enables custom merging per feature to combine scores.

Ranking and Confidence Estimation

After merging, the system must rank the hypotheses and estimate confidence based on their merged scores. We adopted a machine-learning approach that requires running the system over a set of training questions with known answers and training a model based on the scores. One could assume a very flat model and apply existing ranking algorithms (for example, Herbrich, Graepel, and Obermayer [2000]; Joachims [2002]) directly to these score profiles and use the ranking score for confidence. For more intelligent ranking, however, ranking and confidence estimation may be separated into two phases. In both phases sets of scores may be grouped according to their domain (for example type matching, passage scoring, and so on.) and intermediate models trained using ground truths and methods specific for that task. Using these intermediate models, the system produces an ensemble of intermediate scores. Motivated by hierarchical techniques such as mixture of experts (Jacobs et al. 1991) and stacked generalization (Wolpert 1992), a metalearner is trained over this ensemble. This approach allows for iteratively enhancing the system with more sophisticated and deeper hierarchical models while retaining flexibility for robustness and experimentation as scorers are modified and added to the system.

Watson's metalearner uses multiple trained models to handle different question classes as, for instance, certain scores that may be crucial to identifying the correct answer for a factoid question may not be as useful on puzzle questions.

Finally, an important consideration in dealing with NLP-based scorers is that the features they produce may be quite sparse, and so accurate confidence estimation requires the application of confidence-weighted learning techniques. (Dredze, Crammer, and Pereira 2008).

Speed and Scaleout

DeepQA is developed using Apache UIMA,[10] a framework implementation of the Unstructured Information Management Architecture (Ferrucci and Lally 2004). UIMA was designed to support interoperability and scaleout of text and multimodal analysis applications. All of the components in

DeepQA are implemented as UIMA annotators. These are software components that analyze text and produce annotations or assertions about the text. Watson has evolved over time and the number of components in the system has reached into the hundreds. UIMA facilitated rapid component integration, testing, and evaluation.

Early implementations of Watson ran on a single processor where it took 2 hours to answer a single question. The DeepQA computation is embarrassing parallel, however. UIMA-AS, part of Apache UIMA, enables the scaleout of UIMA applications using asynchronous messaging. We used UIMA-AS to scale Watson out over 2500 compute cores. UIMA-AS handles all of the communication, messaging, and queue management necessary using the open JMS standard. The UIMA-AS deployment of Watson enabled competitive run-time latencies in the 3–5 second range.

To preprocess the corpus and create fast run-time indices we used Hadoop.[11] UIMA annotators were easily deployed as mappers in the Hadoop map-reduce framework. Hadoop distributes the content over the cluster to afford high CPU utilization and provides convenient tools for deploying, managing, and monitoring the corpus analysis process.

Strategy

Jeopardy demands strategic game play to match wits against the best human players. In a typical *Jeopardy* game, Watson faces the following strategic decisions: deciding whether to buzz in and attempt to answer a question, selecting squares from the board, and wagering on Daily Doubles and Final *Jeopardy*.

The workhorse of strategic decisions is the buzz-in decision, which is required for every non–Daily Double clue on the board. This is where DeepQA's ability to accurately estimate its confidence in its answer is critical, and Watson considers this confidence along with other game-state factors in making the final determination whether to buzz. Another strategic decision, Final *Jeopardy* wagering, generally receives the most attention and analysis from those interested in game strategy, and there exists a growing catalogue of heuristics such as "Clavin's Rule" or the "Two-Thirds Rule" (Dupee 1998) as well as identification of those critical score boundaries at which particular strategies may be used (by no means does this make it easy or rote; despite this attention, we have found evidence that contestants still occasionally make irrational Final *Jeopardy* bets). Daily Double betting turns out to be less studied but just as challenging since the player must consider opponents' scores and predict the likelihood of getting the question correct just as in Final *Jeopardy*. After a Daily Double, however, the game is not over, so evaluation of a wager requires forecasting the effect it will have on the distant, final outcome of the game.

These challenges drove the construction of statistical models of players and games, game-theoretic analyses of particular game scenarios and strategies, and the development and application of reinforcement-learning techniques for Watson to learn its strategy for playing *Jeopardy*. Fortunately, moderate samounts of historical data are available to serve as training data for learning techniques. Even so, it requires extremely careful modeling and game-theoretic evaluation as the game of *Jeopardy* has incomplete information and uncertainty to model, critical score boundaries to recognize, and savvy, competitive players to account for. It is a game where one faulty strategic choice can lose the entire match.

Status and Results

After approximately 3 years of effort by a core algorithmic team composed of 20 researchers and software engineers with a range of backgrounds in natural language processing, information retrieval, machine learning, computational linguistics, and knowledge representation and reasoning, we have driven the performance of DeepQA to operate within the winner's cloud on the *Jeopardy* task, as shown in figure 9. Watson's results illustrated in this figure were measured over blind test sets containing more than 2000 *Jeopardy* questions.

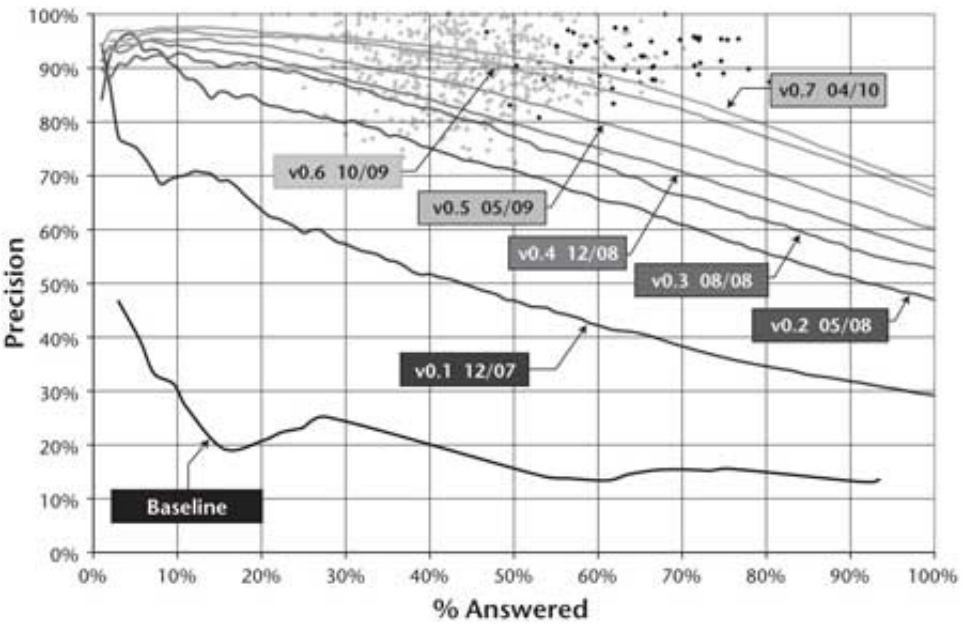

Figure 9. Watson's Precision and Confidence Progress as of the Fourth Quarter 2009.

After many nonstarters, by the fourth quarter of 2007 we finally adopted the DeepQA architecture. At that point we had all moved out of our private offices and into a "war room" setting to dramatically facilitate team communication and tight collaboration. We instituted a host of disciplined engineering and experimental methodologies supported by metrics and tools to ensure we were investing in techniques that promised significant impact on end-to-end metrics. Since then, modulo some early jumps in performance, the progress has been incremental but steady. It is slowing in recent months as the remaining challenges prove either very difficult or highly specialized and covering small phenomena in the data.

By the end of 2008 we were performing reasonably well — about 70 percent precision at 70 percent attempted over the 12,000 question blind data, but it was taking 2 hours to answer a single question on a single CPU. We brought on a team specializing in UIMA and UIMA-AS to scale up DeepQA on a massively parallel high-performance computing platform. We are currently answering more than 85 percent of the questions in 5 seconds or less — fast enough to provide competitive performance, and with continued algorithmic development are performing with about 85 percent precision at 70 percent attempted.

We have more to do in order to improve precision, confidence, and speed enough to compete with

grand champions. We are finding great results in leveraging the DeepQA architecture capability to quickly admit and evaluate the impact of new algorithms as we engage more university partnerships to help meet the challenge.

An Early Adaptation Experiment

Another challenge for DeepQA has been to demonstrate if and how it can adapt to other QA tasks. In mid-2008, after we had populated the basic architecture with a host of components for searching, evidence retrieval, scoring, final merging, and ranking for the *Jeopardy* task, IBM collaborated with CMU to try to adapt DeepQA to the TREC QA problem by plugging in only select domain-specific components previously tuned to the TREC task. In particular, we added question-analysis components from PIQUANT and OpenEphyra that identify answer types for a question, and candidate answer-generation components that identify instances of those answer types in the text. The DeepQA framework utilized both sets of components despite their different type systems — no ontology integration was performed. The identification and integration of these domain specific components into DeepQA took just a few weeks.

The extended DeepQA system was applied to TREC questions. Some of DeepQA's answer and evidence scorers are more relevant in the TREC domain than in the *Jeopardy* domain and others are less relevant. We addressed this aspect of adaptation for DeepQA's final merging and ranking by training an answer-ranking model using TREC questions; thus the extent to which each score affected the answer ranking and confidence was automatically customized for TREC.

Figure 10 shows the results of the adaptation experiment. Both the 2005 PIQUANT and 2007 OpenEphyra systems had less than 50 percent accuracy on the TREC questions and less than 15 percent accuracy on the *Jeopardy* clues. The DeepQA system at the time had accuracy above 50 percent on *Jeopardy*. Without adaptation DeepQA's accuracy on TREC questions was about 35 percent. After adaptation, DeepQA's accuracy on TREC exceeded 60 percent. We repeated the adaptation experiment in 2010, and in addition to the improvements to DeepQA since 2008, the adaptation included a transfer learning step for TREC questions from a model trained on *Jeopardy* questions. DeepQA's performance on TREC data was 51 percent accuracy prior to adaptation and 67 percent after adaptation, nearly level with its performance on blind *Jeopardy* data.

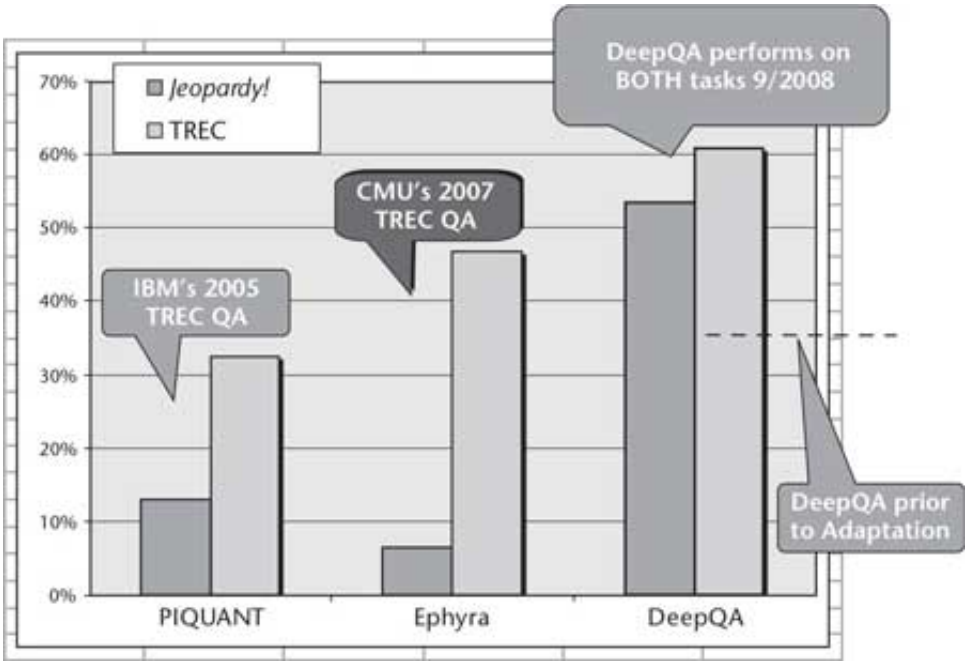

Figure 10. Accuracy on Jeopardy! and TREC.

The result performed significantly better than the original complete systems on the task for which they were designed. While just one adaptation experiment, this is exactly the sort of behavior we think an extensible QA system should exhibit. It should quickly absorb domain- or task-specific components and get better on that target task without degradation in performance in the general case or on prior tasks.

Summary

The *Jeopardy* Challenge helped us address requirements that led to the design of the DeepQA architecture and the implementation of Watson. After 3 years of intense research and development by a core team of about 20 researcherss, Watson is performing at human expert levels in terms of precision, confidence, and speed at the *Jeopardy* quiz show.

Our results strongly suggest that DeepQA is an effective and extensible architecture that may be used as a foundation for combining, deploying, evaluating, and advancing a wide range of algorithmic techniques to rapidly advance the field of QA.

The architecture and methodology developed as part of this project has highlighted the need to take a systems-level approach to research in QA, and we believe this applies to research in the broader field of AI. We have developed many different algorithms for addressing different kinds of problems in QA and plan to publish many of them in more detail in the future. However, no one algorithm solves challenge problems like this. End-to-end systems tend to involve many complex and often overlapping interactions. A system design and methodology that facilitated the efficient integration and ablation studies of many probabilistic components was essential for our success to date. The impact of any one algorithm on end-to-end performance changed over time as other techniques were added and had overlapping effects. Our commitment to regularly evaluate the effects of specific techniques on end-to-end performance, and to let that shape our research investment, was necessary for our rapid

progress.

Rapid experimentation was another critical ingredient to our success. The team conducted more than 5500 independent experiments in 3 years — each averaging about 2000 CPU hours and generating more than 10 GB of error-analysis data. Without DeepQA's massively parallel architecture and a dedicated high-performance computing infrastructure, we would not have been able to perform these experiments, and likely would not have even conceived of many of them.

Tuned for the *Jeopardy* Challenge, Watson has begun to compete against former *Jeopardy* players in a series of "sparring" games. It is holding its own, winning 64 percent of the games, but has to be improved and sped up to compete favorably against the very best.

We have leveraged our collaboration with CMU and with our other university partnerships in getting this far and hope to continue our collaborative work to drive Watson to its final goal, and help openly advance QA research.

Acknowledgements

We would like to acknowledge the talented team of research scientists and engineers at IBM and at partner universities, listed below, for the incredible work they are doing to influence and develop all aspects of Watson and the DeepQA architecture. It is this team who are responsible for the work described in this paper. From IBM, Andy Aaron, Einat Amitay, Branimir Boguraev, David Carmel, Arthur Ciccolo, Jaroslaw Cwiklik, Pablo Duboue, Edward Epstein, Raul Fernandez, Radu Florian, Dan Gruhl, Tong-Haing Fin, Achille Fokoue, Karen Ingraffea, Bhavani Iyer, Hiroshi Kanayama, Jon Lenchner, Anthony Levas, Burn Lewis, Michael McCord, Paul Morarescu, Matthew Mulholland, Yuan Ni, Miroslav Novak, Yue Pan, Siddharth Patwardhan, Zhao Ming Qiu, Salim Roukos, Marshall Schor, Dafna Sheinwald, Roberto Sicconi, Hiroshi Kanayama, Kohichi Takeda, Gerry Tesauro, Chen Wang, Wlodek Zadrozny, and Lei Zhang. From our academic partners, Manas Pathak (CMU), Chang Wang (University of Massachusetts [UMass]), Hideki Shima (CMU), James Allen (UMass), Ed Hovy (University of Southern California/Information Sciences Instutute), Bruce Porter (University of Texas), Pallika Kanani (UMass), Boris Katz (Massachusetts Institute of Technology), Alessandro Moschitti, and Giuseppe Riccardi (University of Trento), Barbar Cutler, Jim Hendler, and Selmer Bringsjord (Rensselaer Polytechnic Institute).

Notes

1. Watson is named after IBM's founder, Thomas J. Watson.

2. Random jitter has been added to help visualize the distribution of points.

3. www-nlpir.nist.gov/projects/aquaint.

4. trec.nist.gov/proceedings/proceedings.html.

5. sourceforge.net/projects/openephyra/.

6. The dip at the left end of the light gray curve is due to the disproportionately high score the search

engine assigns to short queries, which typically are not sufficiently discriminative to retrieve the correct answer in top position.

7. dbpedia.org.

8. www.mpi-inf.mpg.de/yago-naga/yago.

9. freebase.com.

10. incubator.apache.org/uima.

11. hadoop.apache.org.

References

Chu-Carroll, J.; Czuba, K.; Prager, J. M.; and Ittycheriah, A. 2003. Two Heads Are Better Than One in Question-Answering. Paper presented at the Human Language Technology Conference, Edmonton, Canada, 27 May–1 June.

Dredze, M.; Crammer, K.; and Pereira, F. 2008. Confidence-Weighted Linear Classification. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*. Princeton, NJ: International Machine Learning Society.

Dupee, M. 1998. *How to Get on Jeopardy! ... and Win: Valuable Information from a Champion*. Secaucus, NJ: Citadel Press.

Ferrucci, D., and Lally, A. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Langage Engineering* 10(3–4): 327–348.

Ferrucci, D.; Nyberg, E.; Allan, J.; Barker, K.; Brown, E.; Chu-Carroll, J.; Ciccolo, A.; Duboue, P.; Fan, J.; Gondek, D.; Hovy, E.; Katz, B.; Lally, A.; McCord, M.; Morarescu, P.; Murdock, W.; Porter, B.; Prager, J.; Strzalkowski, T.; Welty, W.; and Zadrozny, W. 2009. Towards the Open Advancement of Question Answer Systems. IBM Technical Report RC24789, Yorktown Heights, NY.

Herbrich, R.; Graepel, T.; and Obermayer, K. 2000. Large Margin Rank Boundaries for Ordinal Regression. In *Advances in Large Margin Classifiers,* 115–132. Linköping, Sweden: Liu E-Press.

Hermjakob, U.; Hovy, E. H.; and Lin, C. 2000. Knowledge-Based Question Answering. In *Proceedings of the Sixth World Multiconference on Systems, Cybernetics, and Informatics (SCI-2002)*. Winter Garden, FL: International Institute of Informatics and Systemics.

Hsu, F.-H. 2002. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Princeton, NJ: Princeton University Press.

Jacobs, R.; Jordan, M. I.; Nowlan. S. J.; and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3(1): 79-–87.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the*

Thirteenth ACM Conference on Knowledge Discovery and Data Mining (KDD). New York: Association for Computing Machinery.

Ko, J.; Nyberg, E.; and Luo Si, L. 2007. A Probabilistic Graphical Model for Joint Answer Ranking in Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference,* 343–350. New York: Association for Computing Machinery.

Lenat, D. B. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38(11): 33–38.

Maybury, Mark, ed. 2004. *New Directions in Question-Answering*. Menlo Park, CA: AAAI Press.

McCord, M. C. 1990. Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In *Natural Language and Logic: International Scientific Symposium*. Lecture Notes in Computer Science 459. Berlin: Springer Verlag.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11): 39–41.

Moldovan, D.; Clark, C.; Harabagiu, S.; and Maiorano, S. 2003. COGEX: A Logic Prover for Question Answering. Paper presented at the Human Language Technology Conference, Edmonton, Canada, 27 May–1 June..

Paritosh, P., and Forbus, K. 2005. Analysis of Strategic Knowledge in Back of the Envelope Reasoning. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence* (AAAI-05). Menlo Park, CA: AAAI Press.

Prager, J. M.; Chu-Carroll, J.; and Czuba, K. 2004. A Multi-Strategy, Multi-Question Approach to Question Answering. In *New Directions in Question-Answering,* ed. M. Maybury. Menlo Park, CA: AAAI Press.

Simmons, R. F. 1970. Natural Language Question-Answering Systems: 1969. *Communications of the ACM* 13(1): 15–30

Smith T. F., and Waterman M. S. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147(1): 195–197.

Strzalkowski, T., and Harabagiu, S., eds. 2006. *Advances in Open-Domain Question-Answering*. Berlin: Springer.

Voorhees, E. M., and Dang, H. T. 2005. Overview of the TREC 2005 Question Answering Track. In *Proceedings of the Fourteenth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology.

Wolpert, D. H. 1992. Stacked Generalization. *Neural Networks* 5(2): 241–259.

David Ferrucci is a research staff member and leads the Semantic Analysis and Integration department at the IBM T. J. Watson Research Center, Hawthorne, New York. Ferrucci is the principal investigator for the DeepQA/Watson project and the chief architect for UIMA, now an

OASIS standard and Apache open-source project. Ferrucci's background is in artificial intelligence and software engineering.

Eric Brown is a research staff member at the IBM T. J. Watson Research Center. His background is in information retrieval. Brown's current research interests include question answering, unstructured information management architectures, and applications of advanced text analysis and question answering to information retrieval systems..

Jennifer Chu-Carroll is a research staff member at the IBM T. J. Watson Research Center. Chu-Carroll is on the editorial board of the *Journal of Dialogue Systems,* and previously served on the executive board of the North American Chapter of the Association for Computational Linguistics and as program cochair of HLT-NAACL 2006. Her research interests include question answering, semantic search, and natural language discourse and dialogue..

James Fan is a research staff member at IBM T. J. Watson Research Center. His research interests include natural language processing, question answering, and knowledge representation and reasoning. He has served as a program committee member for several top ranked AI conferences and journals, such as IJCAI and AAAI. He received his Ph.D. from the University of Texas at Austin in 2006.

David Gondek is a research staff member at the IBM T. J. Watson Research Center. His research interests include applications of machine learning, statistical modeling, and game theory to question answering and natural language processing. Gondek has contributed to journals and conferences in machine learning and data mining. He earned his Ph.D. in computer science from Brown University.

Aditya A. Kalyanpur is a research staff member at the IBM T. J. Watson Research Center. His primary research interests include knowledge representation and reasoning, natural languague programming, and question answering. He has served on W3 working groups, as program cochair of an international semantic web workshop, and as a reviewer and program committee member for several AI journals and conferences. Kalyanpur completed his doctorate in AI and semantic web related research from the University of Maryland, College Park.

Adam Lally is a senior software engineer at IBM's T. J. Watson Research Center. He develops natural language processing and reasoning algorithms for a variety of applications and is focused on developing scalable frameworks of NLP and reasoning systems. He is a lead developer and designer for the UIMA framework and architecture specification.

J. William Murdock is a research staff member at the IBM T. J. Watson Research Center. Before joining IBM, he worked at the United States Naval Research Laboratory. His research interests include natural-language semantics, analogical reasoning, knowledge-based planning, machine learning, and computational reflection. In 2001, he earned his Ph.D. in computer science from the Georgia Institute of Technology..

Eric Nyberg is a professor at the Language Technologies Institute, School of Computer Science, Carnegie Mellon University. Nyberg's research spans a broad range of text analysis and information retrieval areas, including question answering, search, reasoning, and natural language processing architectures, systems, and software engineering principles

John Prager is a research staff member at the IBM T. J. Watson Research Center in Yorktown Heights, New York. His background includes natural-language based interfaces and semantic search, and his current interest is on incorporating user and domain models to inform question-answering. He is a member of the TREC program committee.

Nico Schlaefer is a Ph.D. student at the Language Technologies Institute in the School of Computer Science, Carnegie Mellon University and an IBM Ph.D. Fellow. His research focus is the application of machine learning techniques to natural language processing tasks. Schlaefer is the primary author of the OpenEphyra question answering system.

Chris Welty is a research staff member at the IBM Thomas J. Watson Research Center. His background is primarily in knowledge representation and reasoning. Welty's current research focus is on hybridization of machine learning, natural language processing, and knowledge representation and reasoning in building AI systems.