Elementary? Question Answering, IBM's Watson, and the Jeopardy! Challenge

Raman Chandrasekar

In this article, we start with John McCarthy's definition of Artificial Intelligence and describe select approaches to its sub-areas: natural language processing and question answering. We then outline the challenge that IBM Research undertook to build Watson, a question-answering computer system that would compete at the level of a human champion in real time on the American TV quiz show Jeopardy!. We describe the rules of Jeopardy! and list issues that any contestant faces in competing on Jeopardy!. We describe how IBM's Watson system addresses these issues with its metrics and DeepQA architecture. We briefly describe how Watson performed in the Jeopardy! Challenge shows, and conclude with some thoughts about Watson and its potential applications.

Raman Chandrasekar received his PhD in machine translation from TIFR, Bombay, and worked at the National Centre for Software Technology, Bombay. From 1995, he worked at IRCS, University of Pennsylvania on automatic sentence simplification, and using NLP to improve search. From 1998 to 2010, he worked on search and text mining at Microsoft Research. He then worked on news-dissemination at startup Evri.com. Since 2012, he has been at ProQuest, Seattle.

1. John McCarthy and AI

Not everyone gets to make a significant impact on a field. Even fewer make a *series* of influential advances in a field. John McCarthy was one of those special people. As you would have read in the article on him by Rajaraman in this issue of *Resonance* [1], McCarthy defined and named the field of *Artificial Intelligence* (AI), developed the LISP programming language widely used in work in artificial intelligence, and in addition came up with a number of profound ideas including garbage collection, non-monotonic reasoning, and situation calculus.

McCarthy in his personal website[1] defined AI as "the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." But then, how do we define intelligence in a machine? Is

[1] http://www-formal.stanford.edu/jmc/whatisai/node1.html

Keywords
John McCarthy, artificial intelligence, natural language processing, question answering system, IBM, DeepQA, Watson, Jeopardy!, quiz show.

doing simple mathematics intelligence? If so, does this mean a calculator is intelligent? Is a machine that plays chess intelligent?

To answer this question, Alan Turing [2] proposed a test for machines to be considered intelligent. Turing defined an "imitation game" by which the machine would interact with the observer using natural language conversations via an old-style text-only screen and keyboard, and the machine would try and persuade the observer into thinking it was human. Turing suggested that a machine should be considered intelligent if it could successfully convince a human observer that it was human. This test has been named the Turing Test in his honor.

Question Answering (QA) is seen as a sub-area of NLP that refers specifically to answering natural language questions, and thus demonstrating understanding and use of language.

Note that the test involves understanding natural language. Many researchers in AI prefer to label this area, which covers a number of aspects of language comprehension such as natural language understanding and generation, word sense disambiguation, automatic summarization, machine translation, speech recognition and generation, etc., with the umbrella term Natural Language Processing, often abbreviated to NLP. In particular, Question Answering (QA) is seen as a sub-area of NLP that refers specifically to answering natural language questions, and thus demonstrating understanding and use of language.

It is a pleasure to pay homage to McCarthy by looking briefly at select NLP systems and more deeply at a significant recent development in the area of question answering – IBM Research's work on the DeepQA architecture, the Watson project and the Jeopardy! Challenge.

In this article, we will first outline and compare select approaches to NLP/QA. In Section 3, we describe the challenge that IBM Research took on, to build the Watson computer system competing in real time on the Jeopardy! show at the level of a human champion. In Sections 4 and 5, we describe the rules of Jeopardy! and give a flavour of the issues facing any contestant when competing on Jeopardy!. Section 6 covers the definition of metrics of performance. In Section 7, we give a brief description

of the DeepQA architecture, with some examples of Watson's approach in Section 8. We briefly describe how Watson performed in the live Jeopardy! shows in Section 9, and conclude with some thoughts about Watson and its applications in Section 10.

2. NLP and Question Answering Systems

In this section, we will mention a few interesting NLP/QA systems. The goal here is to illustrate some of the issues in understanding, and a few approaches to handling these issues. This is nowhere near a complete survey. You will find descriptions of such systems in any book on AI or NLP, such as [3] or [4]. One of the most readable accounts of early AI systems, including NLP systems, may be found in [5].

2.1 *ELIZA*: *Template-Driven Responses*

Joseph Weizenbaum from MIT wrote ELIZA [6], one of the earliest examples of NLP. This program, running the DOCTOR script, imitated a psychotherapist, producing human-like interactions in response to users' statements and questions. The system (named after George Bernard Shaw's character Eliza Doolittle in Pygmalion) worked by using simple parsing of the user's input, and combining simple substitutions, canned phrases and templates to respond to users. For instance, if the user said, "*I am bored with you*", ELIZA might respond with "*Why are you bored with me?*" Here the system may be using a template to change the sentence to a question and substituting pronouns. If the system sees a keyword such as mother, father, brother, sister, etc., it could respond with "Tell me more about your family." If the system did not have a suitable substitution, it could use one of a number of generic responses to keep the user talking, such as "Tell me more." Try out a version of the Eliza Chatbot at *http://www.simonebaldassarri.com/eliza/eliza.html* – you will soon realize the limitations of the system. *Figure* 1 shows a small sample of interaction with this system; ELIZA's comments start with a '>' and the user's comments start with an asterisk ('*').

ELIZA imitated a psychotherapist, producing human-like interactions in response to users' statements and questions.

Weizenbaum was surprised by how seriously people took the program, even after he explained to them how ELIZA worked. For example, users asked him to leave the room when they interacted with ELIZA, or accused him of invading their privacy when they learnt that their conversations were being recorded. His book has a good discussion of issues in this area.

> Hello, I am Eliza.
* Hello, I feel troubled.
> Do you often feel troubled?
* Once in a while.
> Come, come, elucidate your thoughts.
* My mother thinks I worry too much.
> That is quite interesting.
* She says I should laugh more.
> Can you elaborate on that?

Figure 1. Sample interaction with the ELIZA Chatbot.

2.2 *SHRDLU and Understanding in a Simplified World*

Terry Winograd (also from MIT) wrote SHRDLU [7], another early language understanding program, in 1968–1970. SHRDLU operated in a simplified virtual world of blocks – rectangular blocks, cones, spheres, etc. of different colours. Users could have a conversation with SHRDLU, naming blocks, moving them, asking about the state of the blocks, etc.

SHRDLU seemed to understand the user because:

- The domain was a simple, small blocks world, with vocabulary limited to a few nouns (e.g., block, cone), a few verbs (e.g., place on, move to) and some adjectives. There were only a few variations of commands one could generate.
- The system could remember labels (names) given to objects or simple collections or arrangements of objects (e.g., arch), and answer questions about these names, or recreate these arrangements.
- The system had some memory of what actions it had taken, which it could use to answer questions or determine what item a command referred to. For example, assume the world included two different blue blocks and a red one and that SHRDLU had moved one of the blue blocks in a previous step. Assume the next command is, "Put the blue block on the red block". SHRDLU would infer that the blue block to be moved is the one it had moved in the previous step. SHRDLU had some rules and could also learn from examples to identify what was physically possible, and what was not. Thus, for example, it would avoid stacking a cone on top of a cone.

Person: Pick up a big red block.
Computer: OK. (does it)
Person: Grasp the pyramid.
Computer: I don't understand which pyramid you mean.
Person: Find a block which is taller than the one you are holding and put it into the box.
Computer: By "it", I assume you mean the block which is taller than the one I am holding.
Computer: OK. (does it)
Person: What does the box contain?
Computer: The blue pyramid and the blue block.
Person: What is the pyramid supported by?
Computer: The box.
Person: How many blocks are not in the box?
Computer: Four of them.
Person: Is at least one of them narrower than the one which i told you to pick up?
Computer: Yes, the red cube.
Person: Is it supported?
Computer: Yes, by the table.
Person: Can the table pick up blocks?
Computer: No.
Person: Can pyramid be supported by a block?
Computer: Yes.

Figure 2. Sample SHRDLU demo interaction.

SHRDLU was a great demonstration of planning and understanding in a very simple virtual world. Incidentally, it was written in LISP (and Micro-Planner). Winograd's page on SHRDLU (*http://hci.stanford.edu/~winograd/shrdlu/*) has a pointer to Windows-versions of the SHRDLU program, with source code, which you can try out. A sample interaction with SHRDLU from that page is given in *Figure* 2.

2.3 *The START Natural Language System*

Fast-forward many years. Boris Katz and his group (again, from MIT – you see a pattern here!) developed the START [8] Natural Language System (operating since December 1993), designed to answer natural language questions about its knowledge base.

START (SynTactic Analysis using Reversible Transformations) uses 'annotations' – sentences and phrases to describe content associated with information segments. An understanding module analyzes text given to the system to produce a knowledge base of information contained in the text, while a generating module produces English sentences. Given a specific knowledge base of annotated text, typically covering information about people, places, movies, etc., input questions are answered by matching annotations of these questions to annotations of information segments in the knowledge base, extracting relevant information and using the generating module to produce answers. You can try START at *http://start.csail.mit.edu*/, for example, with the query "Who is John McCarthy?"

An understanding module analyzes text given to the system to produce a knowledge base of information contained in the text, while a generating module produces English sentences.

2.4 *AskMSR*

AskMSR [9] is a system developed at Microsoft Research in 2002, which uses the Web as a large data resource. Using some substitution rules, AskMSR transforms user queries into search queries. For example, a user query like "Who is Bill Gates married to?" would be transformed to a web search query like "Bill Gates is married to". This query is sent to a search engine, and the results for the query are analyzed. From the result summaries, AskMSR extracts a number of substrings, filters out these substrings based on query types (i.e., is it a *Who* question or *What* question, etc.) and assembles answers to the user query from frequent substrings. The system depends on information redundancy on the Web, and does not have any means to check the validity of the answers it returns. So, for instance, if there are a large number of instances of the metaphorical use of "married" as in "Bill Gates is married to his work", AskMSR may return "his work" as an answer to the user query above.

AskMSR depends on information redundancy on the Web, and does not have any means to check the validity of the answers it returns.

2.5 *Comparison of Systems*

There has been extensive work in NLP and QA and the interested reader is urged to search the web for other relevant resources. The systems mentioned here were chosen to cover a spectrum of

There are a number of issues in conversational language that need to be solved for true NLP.

approaches. ELIZA is a template-based system, which is really limited in its conversational abilities, as well as in the knowledge of its domain. SHRDLU is slightly better – it has better knowledge of its small domain, but the range of the discourse between SHRDLU and its users is still limited. START has a large knowledge base built over time, but again it is focused on answering questions in a few specific areas – questions about people, places, movies, some conversions, etc. AskMSR relies on redundancy to answer questions, and fails when there is no information or when there is considerable metaphorical or word-sense ambiguity.

There are a number of issues in conversational language that need to be solved for true NLP. These include disambiguating between multiple senses of words (is *Bush* a person or a plant?), dealing with pronoun references (Who does the *she* refer to in the sentence "When Gita met Rita, she was very happy"), dealing with world knowledge (the word *Delhi* in "Delhi took a strong stance" could refer to the Government of India, the State Government, the people in Delhi or a sports team) and pragmatic use of language (the expected response to "Do you know what the time is?" is not a "yes" or a "no", but the time). None of the systems above attack these in any significant fashion. True discourse-understanding systems should deal with all these and many other issues.

Clearly, language understanding is very difficult. Even understanding and using language at the level of a child is not easy, as work in child language acquisition shows (see [10] for a quick introduction to this area). However, these difficulties should not scare us away from the problem of language understanding. Instead, we should view these issues as a treasure trove of problems to be solved, perhaps by tackling them in small chunks, in restricted contexts, one by one. As we shall see in the rest of this article, the DeepQA project and Watson take exactly this approach, to show expert level language understanding in the relatively restricted context of the Jeopardy! show.

Language understanding is very difficult. Even understanding and using language at the level of a child is not easy, as work in child language acquisition shows.

3. Watson and the Jeopardy! Challenge

Any advance in question-answering has applications in a variety of business domains.

For a long time, expert performance in games such as *Go* or chess were considered to be hallmarks of intelligent behavior. There has been significant attention from AI researchers in developing chess-playing programs. IBM tackled that challenge by developing the Deep Blue chess-playing system [11] that won a six-game match against world chess champion Garry Kasparov in May 1997.

In much the same spirit, IBM Research sought a major challenge to tackle the problem of precisely answering users' natural language questions. Clearly any advance in question- answering has applications in a variety of business domains.

Question-answering in relatively unconstrained domains requires collaboration and coordination amongst a number of areas of computer science and AI.

Question-answering in relatively unconstrained domains requires collaboration and coordination amongst a number of areas of computer science and AI including: information retrieval, natural language understanding, machine learning, distributed computing systems, and performance.

With its capability in all these areas, and collaborating with various University partners including Carnegie Mellon University, MIT, University of Massachusetts, University of Southern California, University of Texas and the University of Trento to add to this expertise, IBM Research chose to build a computer system named Watson (named after IBM's founder Thomas J Watson) to compete at the human champion level *in real-time* on the American TV quiz show Jeopardy![2].

In its latest incarnation, the Jeopardy! show has been aired on prime-time American TV since 1984. In this show, three contestants compete with each other to answer questions from a very broad spread of topics. Because wrong answers attract penalties, competitors have to be reasonably sure they know the correct answer before they respond. A champion Jeopardy! player would have to get the right answer with high precision, with great speed, and with confidence in the correctness of the answer. Ferrucci *et al* [4] estimate a system would have to answer roughly 70% of the

[2] Much of the description of the Watson system in the rest of the article is based on the excellent *AI Magazine* article on Watson and DeepQA by Ferrucci *et al*, [12]. The interested reader is strongly urged to read the original article.

A champion Jeopardy! player would have to get the right answer with high precision, with great speed, and with confidence in the correctness of the answer.

questions with greater than 80% precision in 3 seconds or less to beat a human player. This is the challenge that Watson faced.

In the next two sections, we take a look at the rules of Jeopardy! and at issues in succeeding in Jeopardy!.

4. Rules of Jeopardy!

The Jeopardy! show features rich natural language questions drawn from an extremely wide range of human knowledge, and attracts extremely capable contestants (whom we will also refers to as players). In each show, three contestants are pitted against each other in a 3-round contest. The first two rounds each have a game board of 30 questions, organized in 6 topics or categories. Each *category* has 5 rows of cells, with each cell containing a *clue* and is represented by its dollar value. See *Figure* 3 for a sample game board. In a little twist on the usual quiz format, the clue corresponding to each question is an assertion with some missing information; the contestant has to answer with a response in the form of a question. Thus, if the clue is: "*This harvest festival is celebrated by the Tamil people of India in January*", a valid response would be "*What is Pongal*"? In the rest of the article, we will call the text that is given to the

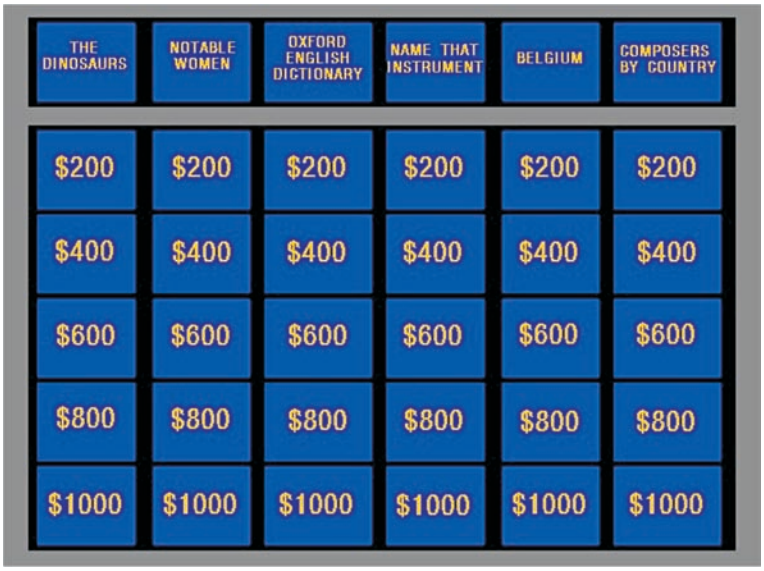

Figure 3. Sample Jeopardy! game board, showing the categories and clues.
Source: http://commons.wikimedia.org/wiki/File:Jeopardy!_game_board.png

contestants as the *clue* or the *question*, even though it is in the form of an assertion. And even though a contestant response is expected to be in the form of a question, we will refer to the response as an *answer*.

In a little twist on the usual quiz format, the clue corresponding to each question is an assertion with some missing information; the contestant has to answer with a response in the form of a question.

To play, the contestant whose turn it is selects a question by naming one of the categories, and choosing a dollar value – e.g., "*Notable Women* for $600". At that point, the host (Alex Trebek, since 1984) reads out the clue corresponding to that cell. Each player has a hand-held signaling device, and any of them may choose to 'buzz-in' using his/her device. Players have to wait till the host reads out the clue completely before buzzing-in; otherwise, they are penalized.

The first player to register a valid buzz then has to provide an answer in the form of an interrogative sentence. If the answer is correct, the dollar value of the clue is added to the player's prize winnings, and the player selects the next clue. If the answer is wrong, or if the player fails to answer within 5 seconds, the value of the clue is subtracted from that player's earnings. The system then allows the other players to buzz-in.

The first two rounds contain one and two 'Daily Double' cells respectively, whose location is random. When the player selects this cell, new rules apply: the player has to first make a wager within set limits and then provide an answer. If the answer is correct, the player's score goes up by the amount of the wager. Otherwise, the player loses that amount. The other players do not get a chance to answer this clue. Each of the first two rounds last only for a specific duration. Clues that are not chosen remain hidden after the round ends.

The first player to register a valid buzz then has to provide an answer in the form of an interrogative sentence. If the answer is correct, the dollar value of the clue is added to the player's prize winnings, and the player selects the next clue.

The third round (the 'Final Jeopardy!' round) consists of a single question. The category is announced and each player privately writes down a wager less than or equal to their earnings up to that point. The clue is then revealed, and the players have 30 seconds (with the famous Jeopardy! theme music playing) to write down their response to the clue. At the end, all the players' responses

and their wagers are revealed. Each player gains their wagered amount if they got the answer right, or loses that amount otherwise. The player with the maximum earnings at this point wins the game, and is invited to play on the next day's game. Winning players get to keep playing each day as long as they keep winning.

How much can a player win? Ken Jennings[3] holds the record for the longest winning streak – he won 74 Jeopardy! games in a row, winning a total of $2,520,700 from that series.

[3] http://en.wikipedia.org/wiki/Ken_Jennings

5. Issues in Succeeding at Jeopardy!

Now that we know the rules, we can see why it could be difficult to win a Jeopardy! game. Here are some of the issues that the players face, besides having to know a lot about a lot of topics, and being able to answer really fast.

***Confidence in their answer*:** Players have to be fast in buzzing-in, so that they are in 'command'. But the penalty for wrong answers requires players to have confidence in their answers before they buzz-in. So players have to know what they know, and also when not to attempt an answer.

***Choosing the category and the clue*:** Players' choices of categories are probably based on their own strengths. For instance, in one recent game, there was a contestant who knew a lot about TV cartoons, so he was able to answer all 5 questions in that category very quickly. When time starts running out, it is probably wise to first pick the higher priced clues in a category you know a lot about. While some categories are straightforward (e.g., *History, Astronomy, Parts of Speech*), others may be based on puns (e.g., *A Round of Gulf (not Golf)*, about bodies of water, not the sport). Some categories have constraints built into their names: *Landlocked Country Fun* (with clues such as *What is the largest landlocked country in Africa?*). Some categories have a condition on the answer expected; for example, all answers must begin with "flo" or contain the letters "end".

When time starts running out, it is probably wise to first pick the higher priced clues in a category you know a lot about.

The actual questions are relatively random, with the implicit understanding that higher value questions are of higher difficulty. However, this is very subjective, as difficulty depends on one's knowledge of an area.

***Wagering strategy*:** There is a lot of strategy in choosing amounts to wager at different stages of the game. Since the wagering strategy is not directly related to question-answering performance, we will not get into any detail about it.

But what does doing well in Jeopardy! mean? In the next section, we discuss this issue.

6. Metrics and Goal Setting

In any competition, to define winning performance, you need one or more measurements or *metrics* to compare your performance against competitors. These metrics may also be used to set goals, measure, and improve one's own performance. For example, in a sport like high jump, a competitor's level is measured by the height of the bar he or she jumps over.

When we talk of computer systems doing well, we need metrics to measure these systems as well, to see how they perform and to keep improving them. What would be meaningful metrics for Watson and the Jeopardy! Challenge task? Overall performance would be based on speed of answering, how well the system estimated its confidence in its answers, how it selected categories and questions to answer, how well it answered these, and its betting strategies, where appropriate. The score at the end – the dollar value of the winnings – is not a good indicator of QA ability because of various chance-based aspects of the game. Speed of answering is also not important if you are only interested in measuring QA performance.

In any competition, to define winning performance, you need one or more measurements or *metrics* to compare your performance against competitors.

Watson uses two metrics to determine core question-answering performance, bypassing speed and aspects of chance:

1. Percent-answered: percentage of questions the system chooses to answer.

Statistics were computed over the 74 games that Ken Jennings won in a row; he managed to answer 62% of the questions with 92% precision. These numbers thus defined the bar for Watson.

2. Precision: the percentage of questions the system gets right, out of those it decides to answer.

A further metric, accuracy, is defined as the precision if all questions are answered.

Watson has a threshold level of confidence to decide when to answer a question. If this level is high, Percent-answered goes down, and Precision goes up. If the threshold is low, the system takes more chances, and Percent-answered goes up while Precision usually goes down.

An analysis of about 2000 historical Jeopardy! games showed that Jeopardy! champions performed with a precision of between 85% and 95%, and were able to buzz-in and answer between 40% and 50% of questions. As a special case, statistics were computed over the 74 games that Ken Jennings won in a row; he managed to answer 62% of the questions with 92% precision. These numbers thus defined the bar for Watson.

But were there already any systems performing at this level? To obtain some baseline performance metrics, a few available QA or QA-like systems were evaluated. These included an IBM system named PIQUANT [13] and a system developed at Carnegie Mellon University named OpenEphyra[4]. Tests were also conducted on a text search engine (using question terms as queries) and a search engine based on structured data. Details about these baselines are available in [12]. However, none of these systems had the high precision and high percent-answered numbers along with the requisite confidence estimation that the Jeopardy! task required.

[4] http://sourceforge.net/projects/openephyra/

What sort of system architecture would attain this level of performance? That is what we consider in the next section, as we take a look at DeepQA.

7. DeepQA Architecture

DeepQA is the software architecture for the massively parallel

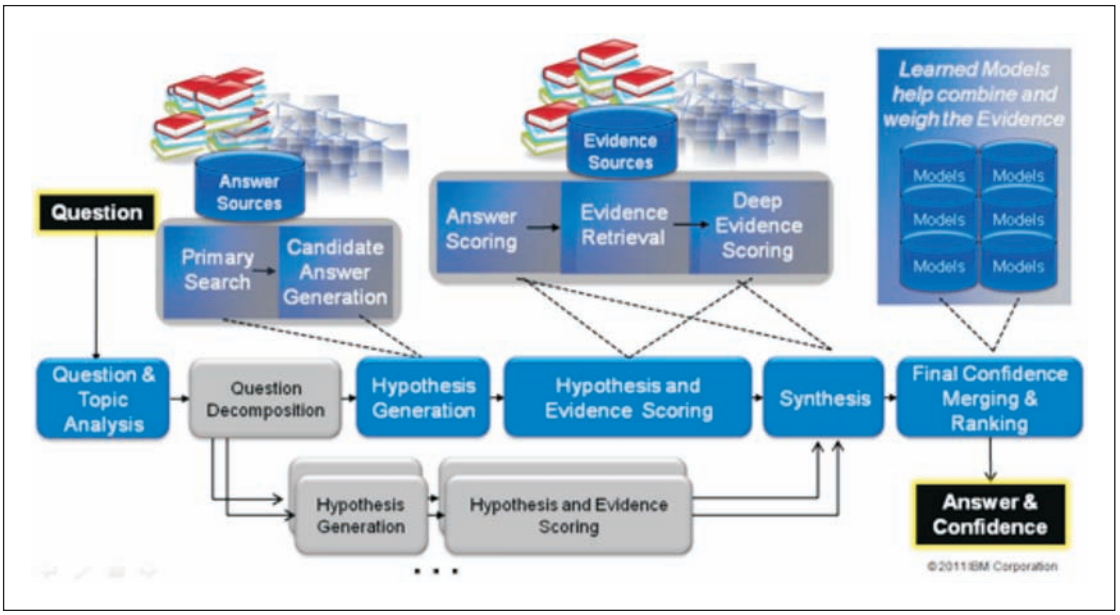

Figure 4. DeepQA High-level Architecture.
Source: http://researcher.ibm.com/researcherview_project_subpage.php?id=2159]
Courtesy: IBM Research.

deep content analysis and evidence-based reasoning that is behind the Watson program which appeared as a Jeopardy! contestant.

Figure 4 provides an overview of DeepQA's architecture. To start with, a content acquisition step gathers content using information for Watson from encyclopedias, dictionaries, news articles, etc., as well as databases and ontologies.

A question analysis step then determines how the system will process each question. Here a number of components extract information at various levels, ranging from

- Category classification: is the question a puzzle, or a pun, etc.?
- Answer type detection: what is the answer expected to be? A person, place, sport, etc.?
- Relation detection: are there relations we can identify between the entities in a question?
- Question decomposition: should questions be broken into subquestions and if so, how?

Hypothesis generation searches the answer sources to generate a set of possible responses. The approach is to generate enough of

these responses initially so that the correct answer is not excluded at this stage by being too restrictive. The assumption is that later steps will filter out the bad responses, and move the right answer to the top. Hundreds of such candidate answers are generated in this stage.

In the next stage, a 'soft' filtering step, using information such as answer type, question classification, etc., the number of candidate answers is reduced to about 100 which are then passed on to the next step.

The next step is hypothesis and evidence scoring. The system first gathers all the evidence related to each candidate answer, from a number of sources of information. This supporting evidence for each candidate is sent to a scoring stage. Here scoring components act on the evidence, evaluating a number of dimensions of evidence, and produce a combined score for each candidate answer.

The next stage merges different equivalent candidate answers (e.g., the candidate answers Mahatma Gandhi, Gandhi, Mohandas Gandhi are equivalent). Finally, the ranking stage uses machine learning to rank the different candidates and their scores to come up with the best answer and its associated confidence value.

To summarize, the system figures out the kind of information required, generates many possible responses, and uses other information from the question to filter out bad responses, while collecting evidence in support of the good responses. Each remaining response is scored on the evidence, and the best answer is bubbled to the top with its associated evidence and confidence score. Note that this approach is not limited to Question Answering – this can be used in a variety of problem-solving techniques.

The DeepQA architecture figures out the kind of information required, generates many possible responses, and uses other information from the question to filter out bad responses, while collecting evidence in support of the good responses.

At the time that Ferrucci *et al* wrote their paper [12], DeepQA was answering more than 85% of the questions within 5 seconds, and performing at about 85% precision at 70% questions-attempted.

8. Watson's Approach

The DeepQA architecture allows for task-specific components to be plugged in, to handle specific aspects. Here are some examples of components specific to the Jeopardy! task.

The DeepQA architecture allows for task-specific components to be plugged in, to handle specific aspects.

There are a number of issues specific to playing Jeopardy! that are not encountered in normal question-answering tasks, such as trying to decode category names. As we have seen above, category names range from the prosaic descriptive ones such as *Astronomy*, which are useful, to ones with embedded puns, which are harder to interpret in a meaningful way. Following the strategy of using many different sources of information, Watson had a component to learn from the category name if it could.

After the clue was revealed, the system made use of query classification. Some questions can be answered with factual information; for example, the question "*These are the two US states that are rectangular in shape.*" This requires Watson to determine what is required in the answer, and to develop a plan to get to the relevant part of this factual information.

Some questions have nested sub-questions, which require question decomposition, for example: "*Which is the southern-most landlocked country in Africa?*" This requires a list of landlocked countries in Africa to be known, and the southernmost of these identified.

It also helps to figure out the expected entity class, or topic, of the answer: whether it is a person's name, the name of a place, a date, etc., for that constrains the set of possible answers. So if the category is *16*th *Century People*, knowing that the answer is a person or persons can be very useful.

In the next section, we will see how this architecture and Watson performed in the Jeopardy! Challenge shows.

9. The Jeopardy! Challenge Shows

In 2011, Watson competed against Ken Jennings and Brad

Watson received its clues as electronic texts at the same time that the players got to see them. It also got an electronic signal to indicate when it could buzz-in. When it was sure of its answer, Watson used an electronic finger to press the buzzer. Watson used a synthesized (text-to-speech) voice to speak out its answer.

Rutter, two of the show's most successful contestants ever, in a 3-day Jeopardy! Challenge consisting of two matches. Watson's appearance on Jeopardy! was widely anticipated and watched.

Watson was represented as a globe avatar. The hardware behind Watson consisted of 90 IBM Power750 processors, with 16 terabytes of RAM holding all its data. The system was not connected to the Internet when playing the game.

There were some changes made to adapt to a computer contestant. Watson received its clues as electronic texts at the same time that the players got to see them. It also got an electronic signal to indicate when it could buzz-in. When it was sure of its answer, Watson used an electronic finger to press the buzzer. Watson used a synthesized (text-to-speech) voice to speak out its answer. Even when Watson did not get to answer, its responses were shown. To keep things fair, Jeopardy! used a random selection of questions that had been written earlier for all-human players, and had never been broadcast. In addition, two categories of questions were excluded because of the additional challenges they presented: audio-visual questions, and questions with categories where the nature of the answer had to be explained by the host.

The first round of the first match was broadcast on February 14th, 2011. Watson showed some weakness, doing better in some categories than in others. At the end of this round, Watson and Brad Rutter were tied at $5000 each, while Ken Jennings was at $2000.

Watson's performance was impressive on the second day (February 15th), during the second round of the first match. Watson's language when choosing topics was entertaining, and the quality of the text-to-speech was quite good. It was amusing to see Watson wagering a non-rounded amount ($6,435) at one point. And because Watson did so well on a number of questions, it was surprising that the system suggested, *"What is Toronto?????"* (instead of the correct answer "*Chicago"*) for the category US

Cities and the clue "*Its largest airport was named for a World War II hero; its second largest, for a World War II battle*". There was a suggestion by a researcher later that the semi-colon in the clue was confusing, and it was therefore not clear that the phrase *second largest* referred to its *second largest airport*. Watson did have Chicago as its second choice. In a strange way, Watson's mistake probably made it all the more human. Watson ended the day with a healthy score of $35,734 to Brad Rutter's $10,400, while Ken Jennings trailed with $4,800.

The second match was broadcast on February 16th. This time, Watson started well but seemed to grow hesitant, and was not quick enough to buzz-in. However, Watson did better later in the game. All three contestants answered the Final Jeopardy! question correctly. Acknowledging Watson's significant achievement, Ken Jennings added, "I for one welcome our new computer overlords", when he wrote down his response. At the end of the matches, Watson finished first with an impressive total score of $77,147; Ken Jennings was second with $24,000 and Brad Rutter third with $21,600.

Acknowledging Watson's significant achievement, Ken Jennings added, "I for one welcome our new computer overlords", when he wrote down his response.

The first prize was a million dollars, the second prize $300,000 and the third prize $200,000. IBM donated the $1 million that Watson won to 2 charities and Ken Jennings and Brad Rutter gave away 50% of their winnings — so it was a good day for both computer science and philanthropy!

10. What Has Watson Given Us?

Watson's win on Jeopardy! Challenge was a big win for the IBM team and its university collaborators, and for natural language researchers everywhere. It was a great advance for question answering, showing that a computer system could perform at championship level on a complex game like Jeopardy!, answering with speed, precision and confidence questions which used the full richness of human language and covered a very broad subject domain.

Watson's win was a great advance for question answering, showing that a computer system could perform at championship level on a complex game like Jeopardy!.

IBM showed that having an architecture, not specific to QA, but

While Watson is a great step in the right direction to understanding language and AI, we have to be clear not to consider QA as a solved problem.

adaptable to different domains, made sense. They showed that it made sense to have a variant of the 'generate and test' paradigm of problem-solving: analyzing the question, coming up with a lot of possible responses, and using intelligent pruning and evidence accumulation and scoring to come to the right answer. They showed how general computer science smarts helped them improve system performance to make answering extremely fast.

While Watson is a great step in the right direction to understanding language and AI, we have to be clear not to consider QA as a solved problem. The mistakes that Watson made show it has more to learn. But once we realize the limits of this work, as well as what makes it impressive, it is easy to see that there are many areas where Watson technology could be of immense use.

For example, IBM has announced an application to improve healthcare insurance decisions using Watson's analysis of data and evidence-based reasoning[5]. IBM also has a project with Memorial Sloan-Kettering Cancer Center, where Watson is being used to provide recommendations with confidence estimates to doctors, to help them in their decision-making[6]. This use of Watson as a decision-aide rather than as a decision-maker will be a safe step forward in life-critical applications. IBM is also looking at applications in finance. They are looking for entrepreneurs and developers to build Watson-powered applications, enlarging the reach of this technology.

These are exciting times both for the technology and for its applications. The vast improvements in QA and related AI areas that DeepQA and Watson brought about foresee the entry of AI systems into new application areas. And in a few years, it may happen that a Jeopardy! clue could read "This computer system was the first to appear as a contestant on Jeopardy!", and the correct answer would be, of course, "What is Watson?"!

Acknowledgements

Sincere thanks to Kody Janney for her careful edits and useful comments on this article.

[5] http://www-03.ibm.com/innovation/us/watson/pdf WellPoint_Case_Study_IMC14792.pdf

[6] http://www-03.ibm.com/innovation/us/watson/pdf/MSK_Case_Study_IMC14794.pdf

Suggested Reading

[1] V Rajaraman, John McCarthy – Father of Artificial Intelligence, *Resonance*, Vol.19, No.3, pp.198–207, 2014.

[2] Alan Turing, Computing Machinery and Intelligence, *Mind*, Vol.LIX, No.236, pp.433–460, October 1950. PDF available at http://mind.oxfordjournals.org/content/LIX/236/433.full.pdf retrieved 2014-01-18

[3] Christopher D Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, 1999.

[4] Daniel Jurafsky and James H Martin, *Speech and Language Processing*, 2nd Ed., Pearson Prentice Hall, 2008.

[5] Margaret A Boden, *Artificial Intelligence And Natural Man*, 2nd Ed., MIT Press, 1987.

[6] J Weizenbaum, *Computer Power and Human Reason*: *From Judgement to Calculation*, W H Freeman, San Francisco, 1976.

[7] Terry Winograd, *Understanding Natural Language*, Academic Press, 1972.

[8] Boris Katz, Using English for Indexing and Retrieving, *Proceedings of the 1st RIAO Conference on User-Oriented Content-Based Text and Image Handling* (RIAO'88), 1988.

[9] Michele Banko, Eric Brill, Susan Dumais and Jimmy Lin, AskMSR: Question Answering Using the Worldwide Web, *EMNLP*, 2002.

[10] Raman Chandrasekar, How children learn to use language, *Resonance*, Vol.13, No.5, pp.430–439, 2008.

[11] M Campbell, A J Hoane and F-H Hsu, Deep Blue, *Artificial Intelligence*, Vol.134, Nos.1–2, pp.57–83, 2002.

[12] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer and Chris Welty,Building Watson, An Overview of the DeepQA Project, *AI Magazine*, pp.59–79, Fall 2010.
Also available at: http://researcher.watson.ibm.com/researcher/files/us-mike.barborak/WatsonInAIMagReprint.pdf

[13] J M Prager, J Chu-Carroll and K Czuba, A Multi-Strategy, Multi-Question Approach to Question Answering, In *New Directions in Question-Answering*, Ed. M Maybury, AAAI Press, 2004.

Address for Correspondence
Raman Chandrasekar
ProQuest
501 North 34th Street
Suite 300, Seattle,
WA 98103, USA
Email:
Raman.Chandrasekar@outlook.com