In the game: The interface between Watson and Jeopardy!

B. L. Lewis

To play as a contestant in Jeopardy!™, IBM Watson™ needed an interface program to handle the communications between the Jeopardy! computers that operate the game and its own components: question answering, game strategy, speech, buzzer, etc. Because Watson cannot hear or see, when the categories and clues were displayed on the game board, they were also sent electronically to Watson. The program also monitored signals generated when the buzzer system was activated and when a contestant successfully rang in. If Watson was confident of its answer, it triggered a solenoid to depress its buzzer button and used a text-to-speech system to speak its response. Since it did not hear the host's judgment, it relied on changes to the scores and the game flow to infer whether its answer was correct. The interface program had to use what were sometimes conflicting events to determine the state of the game without any human intervention.

Introduction

The on-stage presence of IBM Watson* consisted of a monitor displaying an image that reflected the state of the game, a solenoid that mechanically pressed its signaling device (commonly referred to as a buzzer), and a speaker that delivered its voice. Watson could not see or hear (see the section "Alternative communication modes"), so the information it required to play the game was delivered electronically. Consequently, any categories that contained audio or visual clues were not included in its games, nor were categories that required additional information from the host. For our practice sessions, the Jeopardy!** producers constructed games satisfying these restrictions from games that they had already recorded but had not yet broadcast. Some of the games written for that season were randomly withheld and not recorded, and a random selection from these was used to construct the exhibition games.

This paper describes the Watson Controller, the program that orchestrated the communications between the components that made it possible for Watson to perform as a Jeopardy! contestant. The Controller interfaced with the Jeopardy! game system to acquire the clues and other game information to send to the Watson engine. It fed the resulting response confidences and game state to Watson's Strategy component, which decided whether to ring in and how much to wager on Daily Double and Final Jeopardy! clues. The Controller managed several devices and computers to ensure that Watson spoke at the right times, activated the signaling device, reacted visually to the state of the game, and displayed its responses and confidences.

The next section describes the steps a human contestant follows when playing the game and how Watson accomplished them. **Figure 1** illustrates the connections to the components and to the Jeopardy! system.

Playing the game

Starting a round

Each Jeopardy! game consists of three rounds. The first two have six categories with five clues in each, while the final round has only one clue. At the start of each round, the categories are revealed one by one as the host reads them. Because Watson could not hear or see, the six categories were sent as an electronic message from a Jeopardy! computer to the controller program as soon as the last category was revealed. This communication occurred over a wired private local area network (LAN) using the TCP/IP protocol. The messages were designed and timed to match the information displayed visually to the contestants so that Watson received an electrical signal at the same time the

Digital Object Identifier: 10.1147/JRD.2012.2188932

0018-8646/12/$5.00 © 2012 IBM

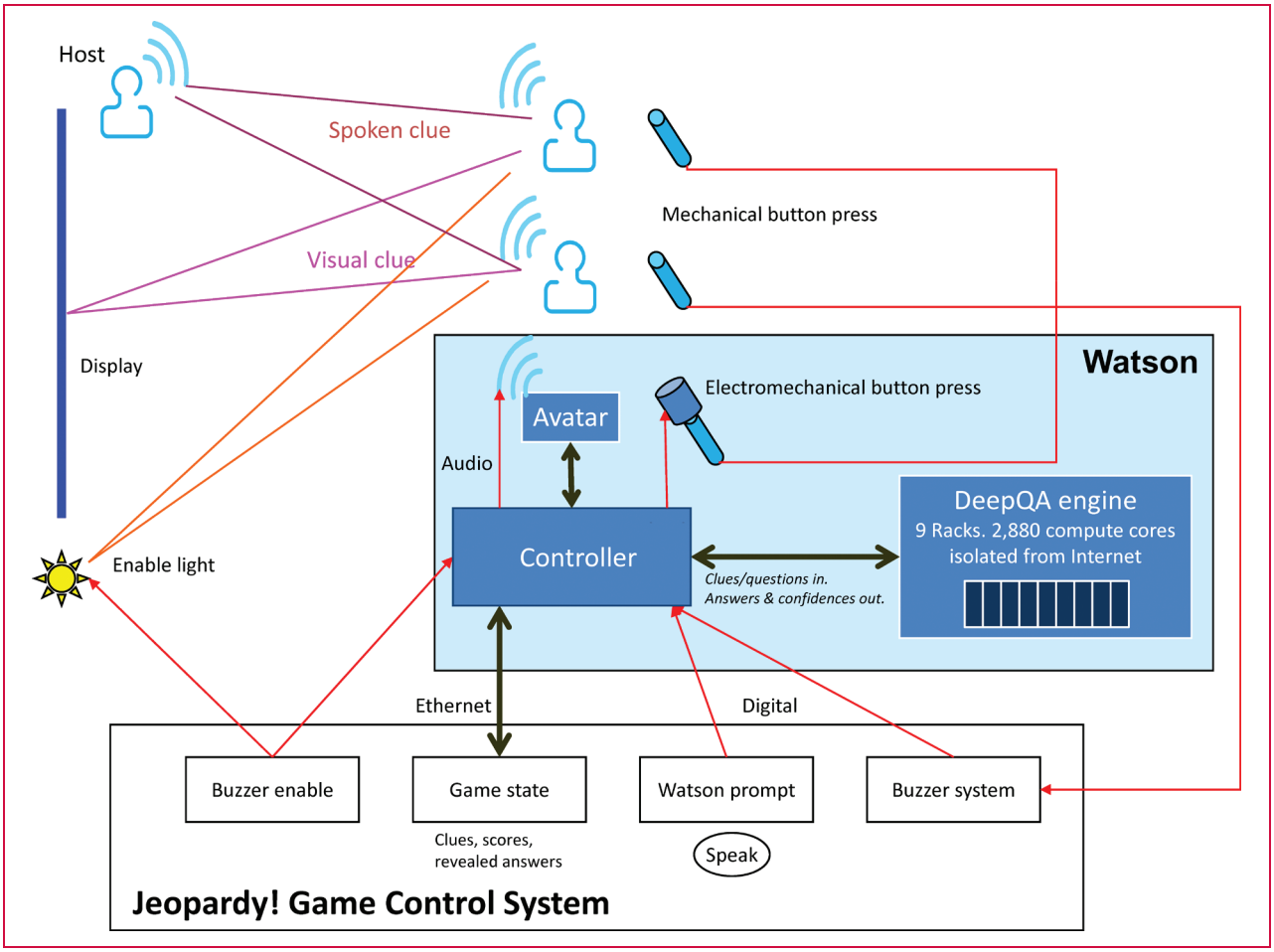

Figure 1

Communication paths between the Watson Controller and the components that allowed Watson to perform as a Jeopardy! contestant.

contestants received a visual one. At the start of each clue, Watson was sent a representation of the game-board display in the form of a list of all of the clue values available on the board, ranging from $200 to $1,000 in the first round and $400 to $2,000 in the second. This list was always sent when the game board was displayed to the contestants so that all parties would know what clues remained available for selection.

Selecting a clue

When it was Watson's turn to select a clue, the choice was made by the Strategy component [1] on the basis of its understanding of the game state, which included the list of available clues, the players' scores, the number of Daily Double clues already played, and its success on other clues in each category. Watson selected clues with a voice generated by the IBM Text-to-Speech (TTS) system [2], built from many hours of recordings of a professional actor. The categories were provided as uppercase text; therefore, a true-casing service [3] was used to restore their case. This was found to improve the quality of the pronunciations produced by the TTS system, which was designed for mixed-case text.

Filters were added to handle the unusual punctuation used in categories. For example, a word or part of a word might be quoted, purely to indicate a sequence of characters that must appear in the response, e.g., "L"egalese and "Ph"un words. Categories often contain fabricated words (e.g., "Dude-er-onomy" and "Scentsational"), but because the TTS system does not rely on a dictionary of known words, it had no trouble with words that followed the usual English pronunciation rules.

To speed up clue selection, all the categories and clue values were synthesized and cached as soon as they were revealed at the start of each round, so the appropriate pair of phrases could be produced with no latency. To

improve the flow of the game, Watson emulated the behavior of some contestants by omitting the category or value if possible ("800 please", "let's finish European History") and used the phrase "same category" if the previous clue had been selected from the same category less than 30 seconds earlier.

Unlike some contestants, Watson did not abbreviate categories and, of course, could not be confused by such a tactic because it was always told the location of a clue on the game board.

Reading a clue

As with the categories, the clue text and its location on the board were sent to Watson as an electronic message. The message was sent as soon as the clue was displayed for the other contestants to see so Watson's silicon cells received the information at essentially the same time it reached the humans' retinas. The Controller then sent the clue to the most important part of the system, the cluster of powerful computers housed in an adjacent room that made up the DeepQA engine [4]. The cluster was isolated from the Internet and was connected to the Controller via a private wired Ethernet LAN, independent of the one connected to the Jeopardy! computers. The engine was implemented as an Unstructured Information Management Architecture (UIMA) [5] Asynchronous Service, and the Controller created and sent it a question Common Analysis Structure (CAS) populated with the clue text, the category, and information about already played clues in the same category (a CAS is an object-based UIMA data structure holding the data to be analyzed as well as the results of any analytics). The engine responded with two answer CASs, each containing the ten best answers and corresponding confidences [6]. The DeepQA architecture exploited the scale-out capabilities of UIMA to produce the first CAS in about 3 seconds, whereas the second CAS with the complete analysis took another 1 to 2 seconds [7]. The first allowed Watson to make a fairly quick decision about ringing in, often before the host had finished reading the clue, whereas the second more accurate set of answers was usually available by the time Watson had to utter its top choice.

Ringing in

When the host finishes reading the clue, an off-stage operator presses a button that enables the buzzer system and turns on "enable" lights next to the game board indicating that it is legal to ring in. The buzzers are monitored by the "lock-out" computer, so called because the first player to ring in (or lock in) locks out the other players. Players often try to anticipate the operator's action and press their buzzer just as the host completes the clue, i.e., before they see the enable lights. If a player attempts to ring in too soon, a penalty of one-quarter second is applied. Ring-in attempts made after the buzzer system has been enabled but before the penalty has expired are ignored. The first player to ring in successfully has 5 seconds to deliver a response. Human players sometimes ring in before they have remembered the complete answer and then use this time to formulate their response.

Watson could predict if it might know the answer from the category text alone, and it did so for Final Jeopardy!, but it always waited for the clue to be processed before deciding whether to ring in. Watson did not listen to the host but instead used a Universal Serial Bus (USB)-connected digital input/output (I/O) device to monitor digital signals generated by the lock-out computer. When the Controller received a set of answers from the DeepQA engine and the buzzer system was enabled, it would ask the Strategy component if and when it should ring in. The strategy based its decision on the confidence of the top answer and its understanding of the game state. If the confidence score were above 50%, Watson would usually ring in, but if it were losing badly, this threshold might be lowered. On the other hand, it might be raised if a wrong response could jeopardize a good outcome [1]. Some end-game situations required detailed computations, so this request for a ring-in decision was made only when needed, when an answer was available, and when it was legal to ring in.

Watson's mechanical thumb consisted of a solenoid mounted on top of the signaling button at its podium. An external power supply and Solid State Relay (SSR) were stored inside the podium, and the SSR was activated by a signal generated by the USB-connected digital I/O device. The solenoid and buzzer were encased in a transparent plastic cylinder to reduce the noise produced, and although the click was audible from a few feet away, the contestants often did not notice it in the heat of a game. Watson's reaction time was limited by the C++ and Java** code paths monitoring the enable signal, the Strategy component deciding to buzz, the code activating the solenoid, and the mechanical operation of the solenoid. Watson's buzzer speed was more consistent than that of the human players; therefore, when it was able to quickly compute a high-confidence answer, Watson was difficult to beat. However, the human players were able to do so when they anticipated the actions of the enable operator and rang in just as the buzzer system was enabled.

Delivering the response

The Controller also used the digital I/O device to monitor digital signals generated by the buzzer system indicating which player rang in first. If Watson won the buzz, a special Jeopardy! "prompt" operator would press a button when the host asked for Watson's answer. This sent a signal to the Controller via the Jeopardy! lock-out computer, to affirm the fact that only Jeopardy! employees were operating the game, and all game events were sent from (and recorded by) Jeopardy! computers. This operator was the only human

affecting Watson's behavior during the games, telling Watson when, not what, to speak, and therefore having no impact on Watson's accuracy.[1] Upon receiving the signal, the Controller delivered the answer text to the TTS system for conversion to audio.

Daily Doubles

When a player selects a Daily Double clue, a wager must be submitted before the clue is revealed. If Watson had selected the clue, the Controller would ask the Strategy component to compute a wager and would then deliver it when prompted. After the host had read the clue, the prompt operator would signal that it was time for Watson to respond. If by this time the DeepQA engine had not delivered its second set of answers, the Controller would wait for it. If it did not arrive within 5 seconds, the Controller would use the first set or say "I'm stumped" if none had arrived in time. If the confidence was low, the Controller would add some words to indicate its uncertainty, e.g., "I'm not sure, but what is . . ."

Displaying the answers

During the early practice games, it was found to be both useful and entertaining to display Watson's top three responses for each clue. Even if the top response was incorrect, the audience would be able to see how often the correct response was in the second or third position. The confidence score for each response was displayed numerically as a percentage and graphically as the length and color of a horizontal bar. A vertical line through the bars indicated the ring-in threshold so that viewers could easily recognize when Watson was confident enough of its top response to ring in. The responses were visible only to the audience (and sometimes the host) but were not displayed until a player had locked in, or at the end of a triple stumper when no player rang in. During Final Jeopardy!, the top three responses were displayed as soon as Watson's response was revealed on the front of its podium. Displaying the answers any earlier caused significant distractions because the audience sometimes reacted with groans or laughter. They were displayed on a separate monitor at the edge of the stage and were also inserted at the bottom of the broadcast image.

Rebounding

If a player responds incorrectly, the buzzer system is reenabled, and the other players have the opportunity to ring in. If the Controller saw that the buzzer system had been enabled again for the same clue, it would ask the Strategy component if it should ring in on the rebound. Of course, if Watson had responded incorrectly earlier, it would not try again—a mistake that human players have been known to make. Since Watson did not hear the previous response, there was a risk that it would repeat the same wrong answer, a careless mistake that human players rarely make. However, Watson did make such an error during one of the exhibition games, as did a couple of humans during our practice games.

Ending the clue

At the end of a clue, players would know what the correct response was since the host would either accept a contestant's response or give the correct response. In order to provide Watson with the same information available to the others, it was sent the correct response when the game-board operator cleared the clue and displayed the dollar values of the remaining clues. Some clues might have multiple acceptable responses, but Watson was always sent the first one stored in the game data by the clue writing team. The Controller compared this to Watson's top answer and passed an accuracy measure to the Strategy component to help track its success on that category.

It also sent the clues and correct responses for all the revealed clues in the same category to a Learning Between Questions component [8] that was deployed as a UIMA Asynchronous Service on the same cluster of computers that hosted the DeepQA engine. This processing usually completed before the next clue was displayed, and the results of this learning operation were saved and added to the question CAS when processing other clues in the same category, enabling a better understanding of the type of answer expected.

To help track Watson's performance in each category, the Strategy component had to be told who won the last clue, and it needed this information before selecting another clue. At the end of a clue, the scoring operator would change the scores, the game-board operator would clear the clue, and the prompt operator might immediately prompt Watson to speak, without waiting for the host to request a selection. Watson would often receive the prompt signal first but not be able to select a clue until the game-board computer sent the updated list of available clues. The score changes were often late and sometimes confusing if the operator made a mistake and quickly corrected it. Thus, the Controller attempted to determine the expected score changes during a clue from changes to the enable, lock-in, and prompt signals. These signals were more timely than the score changes but not always reliable since the buzzer operator might make an error or the host might signal that time is up just as a player rings in. Operator errors can be edited out of a recorded television broadcast, but it was important to make Watson robust to such errors and to handle the out-of-order overlapping messages and events reliably.

[1] However, it so happened that during the exhibition match, the prompt operator did temporarily affect Watson's performance. When a system error caused a clue to appear on the wrong part of the game board, the operator did not react when the host asked Watson to answer, and Watson was judged wrong. Because of the source of the error, the clue was replaced and replayed.

Final Jeopardy!

When the Final Jeopardy! category was revealed, the Controller sent it to a service on the nearby cluster, which returned a confidence score to be used by the Strategy component when computing the wager. Wagers are entered during a commercial break; therefore, the strategy was able to devote considerable effort to this computation. The Controller sent the wager to the Jeopardy! game system as an electronic message, and it was saved as if it had been typed in at Watson's podium. Once all the wagers are entered, the host tells the players whether to start their response with "Who" or "What," so Watson was sent this information as a message.

Since players have 30 seconds to write their responses, the DeepQA engine was able to spend more resources on these more difficult clues [9]. The first answer was usually available within 5 seconds, and the second answer soon afterwards. Each answer was immediately sent to the Jeopardy! system, which kept only the last received. When the 30 seconds had expired, the contestants' pens were disabled, and any further answers from Watson were ignored. The Controller was notified when Watson's response was revealed to the audience so that it could also display its top three answers and confidences. When all the scores had been updated and the winner determined, Watson's avatar was told to react appropriately—looking happy by spinning many colorful threads or moving the threads to the side to "stare" at the winner.

Platform

The majority of the Controller code was written in Java, using Java Native Interface to access the C++ libraries for the TTS system and the digital I/O device. It was installed on a 2.6-GHz six-core Lenovo ThinkStation** with 8 GB of RAM running Microsoft Windows 7**. The two CPU-intensive components, strategy and TTS, did not overlap their execution; therefore, they had essentially full use of the machine.

Watson's avatar

Watson's avatar was a flat-screen monitor placed behind the center podium that displayed an animation based on the IBM Smarter Planet* logo. It was developed by the digital artist Joshua Davis [10] to react to the 30 different states Watson could be in during a game. Its spinning threads would speed up when Watson was processing a clue, and its colors depended on Watson's confidence in an answer. The threads would also move to one side of the planet when one of its competitors won a clue or a game. The Controller sent the state transitions via TCP/IP to a laptop that was generating the animation. The laptop was placed behind the podium and had an analog audio connection to Watson's voice so that its animation could respond to Watson's speech energy.

Alternative communication modes

Early in the development of the system, other interfaces between Watson and Jeopardy! were considered. Instead of receiving the categories and clues electronically, a video camera could have been focused on the game board, and Optical Character Recognition (OCR) technology could have been used to read the screen. The time taken to perform OCR could probably have been overcome by a slight improvement in the DeepQA engine speed, but it was decided that an effort to investigate and compensate for the impact of OCR delays and errors was outside this project's focus on core QA capability. Likewise, a photo-sensitive diode could have monitored the enable lights and generated an equivalent to the buzzer-enabled signal. The Final Jeopardy! wager and response are usually written with a stylus on a touch screen, although blind players may use a keyboard. Watson could have used a USB keyboard emulator to send the appropriate keystrokes.

Watson could have been given some hearing ability by applying Automatic Speech Recognition (ASR) technology to listen for the host's prompts and for the other contestants' responses. The first would merely have avoided the need for a prompt operator, but the second had the potential of preventing Watson from repeating a wrong answer on a rebound. Analysis of practice games showed that, on rebounds, Watson was more than ten times more likely to be correct than to repeat a wrong answer; therefore, to avoid the cost of mistakenly rejecting a correct answer, the ASR precision would have to be significantly greater than 90%, resulting in low recall, i.e., requiring a high accuracy reduces the chance of identifying any identical answers. Since these wrong repeats were rare, averaging only 1 in every 16 games, it was decided that resources would be better deployed in improving Watson's accuracy, and therefore make them even rarer. In fact, one did occur during the exhibition games, costing Watson $1,000, but it would have been a rather difficult ambiguous case to detect as the speech was "What are the 20's", whereas Watson's answer was "What is 1920s".

Since the goal of the project was to demonstrate the power of IBM's QA technology and not to make a robot that could perform more like a human, the simple "deaf and blind" model was followed.

Conclusion

In this paper, we have described the communication paths and connections that enabled Watson to understand the state of the game and to perform as a contestant on Jeopardy!, and the program that orchestrated the messages between the various components. We have described the difficulties of handling the multiple overlapping events produced by the actions of the operators and players, and how this control program had to handle the glitches and errors that can occur

in playing a sophisticated television game show, with essentially no human intervention.

Acknowledgments

The IBM Watson Team would like to acknowledge the critical support and cooperation received from the technical team at Jeopardy! in developing the connections to their computers and in providing a complete installation that allowed us to play many practice games.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Jeopardy Productions, Inc., Sun Microsystems, Lenovo Group Limited, or Microsoft Corporation in the United States, other countries, or both.

References

1. G. Tesauro, D. C. Gondek, J. Lenchner, J. Fan, and J. M. Prager, "Simulation, learning, and optimization techniques in Watson's game strategies," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 16, pp. 16:1–16:11, May/Jul. 2012.
2. J. Pitrelli, R. Bakis, E. M. Eide, R. Fernandez, W. Hamza, and M. A. Picheny, "The IBM expressive text-to-speech synthesis system for American English," *IEEE Trans. Audio Speech Lang. Process.*, vol. 14, no. 4, pp. 1099–1108, Jul. 2006.
3. L. Vita, A. Ittycheriah, S. Roukos, and N. Kambhatla, "tRuEcasing," in *Proc. ACL*, Sapporo, Japan, 2003, pp. 152–159.
4. D. A. Ferrucci, "Introduction to 'This is Watson'," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 1, pp. 1:1–1:15, May/Jul. 2012.
5. Apache UIMA. [Online]. Available: http://uima.apache.org/
6. D. C. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. Duboue, L. Zhang, Y. Pan, Z. M. Qiu, and C. Welty, "A framework for merging and ranking of answers in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 14, pp. 14:1–14:12, May/Jul. 2012.
7. E. A. Epstein, M. I. Schor, B. S. Iyer, A. Lally, E. W. Brown, and J. Cwiklik, "Making Watson fast," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 15, pp. 15:1–15:12, May/Jul. 2012.
8. A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll, "Question analysis: How Watson reads a clue," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 2, pp. 2:1–2:14, May/Jul. 2012.
9. J. Chu-Carroll, E. W. Brown, A. Lally, and J. W. Murdock, "Identifying implicit relationships," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 12, pp. 12:1–12:10, May/Jul. 2012.
10. Joshua Davis Studio. [Online]. Available: http://www.joshuadavis.com

Received July 29, 2011; accepted for publication December 21, 2011

Burn L. Lewis *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (burn@us.ibm.com).* Dr. Lewis is a member of the Unstructured Information Department at the IBM T. J. Watson Research Center. He received B.E. and M.E. degrees in electrical engineering from the University of Auckland in 1967 and 1968, respectively, and a Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1974. He subsequently joined IBM at the T. J. Watson Research Center, where he has worked on speech recognition and unstructured information management. He is author or coauthor of 8 patents and 12 technical papers.