Statistical Approaches to Question Answering in Watson

J. William Murdock and Gerald Tesauro, IBM TJ Watson Research Center

Introduction

The ability to understand and communicate in natural languages, such as English or Chinese, is a hallmark of human intelligence, and one of the core challenges in the field of Artificial Intelligence. Despite decades of AI research, the goal of building machines with human-level conversational fluency, as depicted by *HAL* and *C3PO* in science fiction, remains elusive and daunting. Nevertheless, a burst of recent progress in the field of Natural Language Processing (NLP) has been enabled by the explosive growth in machine-readable language data (primarily in text or hypertext form) available from sources such as the World Wide Web. Furthermore, increasingly powerful computer hardware makes it feasible to apply complex analysis to these large volumes of text. Consequently, researchers have made exciting progress in automated Question Answering (QA), which aims to find a specific answer to a user's natural language question.

The development of IBM's *Watson* QA system, which successfully competed against human grand champions on the TV game show *Jeopardy!*, provides the most striking demonstration of surprising advances in QA research. Other notable systems include *Wolfram Alpha*, *Siri* and *Evi*. Wolfram Alpha answers fact-based natural language queries by computing the answer from structured knowledge sources. Examples of questions it can answer include "How many internet users are in Europe?" and "What is the fifty-second smallest country by per-capita GDP?" Siri is a speech-based natural language interface for the iPhone. It can answer questions requiring synthesis of on-board data (Contacts, Calendar, etc.) as well as external data (e.g., movie and restaurant reviews). For example, to answer the query "Where can I find some good currywurst around here?", Siri combines GPS, business locations and restaurant reviews to return top-rated German restaurants near your current location. Evi is also a voice-enabled QA engine, available as both iPhone and Android apps, that answers questions using knowledge bases and semantic search technology from True Knowledge Ltd. It can answer questions requiring fairly deep reasoning, such as "Who was President when Queen Elizabeth II was a teenager?"

The development of Watson involved innovations in a wide variety of research areas, including rule-based natural-language processing, knowledge-based representation & reasoning, information retrieval, etc. For more details, we refer the reader to an upcoming special issue of the IBM Journal of Research and Development (available at www.research.ibm.com/journal) which contains a collection of 17 articles covering all aspects of the Watson system. One important theme running through many (but certainly not all) aspects of Watson is the use of statistics. This paper provides some prominent examples of the ways that statistical methods applied to large volumes of data are used in the Watson research project. Watson uses statistics in some of the algorithms it uses to answer questions, and researchers use statistics to evaluate how well Watson answers questions.

DeepQA Overview

A high-level depiction of Watson's DeepQA architecture can be seen in Figure 1. DeepQA is a processing pipeline wherein at each stage, statistical analysis of many alternatives is performed as part of a massively parallel computation. After basic NLP analysis of the question (parsing, etc.), the first stage sifts through a large quantity of unstructured text in a search for documents and passages that are sufficiently “similar” to the text of the question (and also searches for candidate answers in structured sources, when the basic NLP has understood the question well enough to derive a structured query). The results of these searches are used as sources of terms that may be considered as candidate answers. The next stage employs hundreds of algorithms that analyze evidence along different dimensions (“features”) such as type classification, time, geography, and semantic relatedness; as part of that stage, additional supporting evidence is retrieved for each candidate answer. The final stage sums over the different evidence features, using relative weightings that were obtained by training on a corpus of 25K previously aired *Jeopardy!* clues, with known right and wrong answers for each clue. The result is a single numerical score for each candidate, indicating aggregate supporting evidence that the candidate is correct. Watson selects the candidate with highest score, and applies a logistic function to obtain a “confidence” value, i.e., a probability estimate that the answer is correct. Watson will attempt to answer, i.e., “buzz in,” if the confidence value exceeds a threshold value that is computed by Watson's game-strategy component.

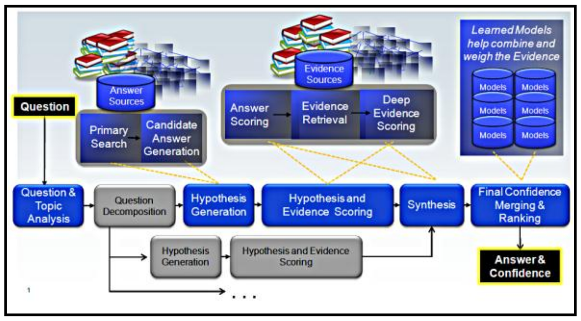

Figure 1: DeepQA Architecture

Information Retrieval

One key step in answering questions involves finding relevant sources of information. Much of the information used in Watson appears in the form of text. Watson has many algorithms for identifying candidate answers in text and evaluating whether the text provides compelling evidence that those candidates being correct. Before these algorithms can begin to operate, Watson needs to find a manageable quantity of text that is relevant to the *Jeopardy!* clue it is trying to answer.

The field of Information Retrieval (IR) focuses on taking a large corpus and finding a small subset that is relevant to a particular query. Watson uses existing, off-the-shelf IR components to perform the retrieval step, but uses a variety of novel mechanisms that take a *Jeopardy!* clue and formulate a query to pose to these components [1]. One very well-established mathematical technique in IR is TF*IDF (Term Frequency * Inverse Document Frequency). This technique is built on the observation that the relevance of a document to a query tends to increase with the number of times that query terms appear in that text but decrease with the number of times that those query terms appear in the whole corpus of text. If some term in the query appears often in a single document and rarely in the whole corpus, then there is considerable evidence that this document is the one that is most useful for the query. In contrast, if a term appears in many documents, then that term is not as useful for deciding whether any one of those documents is relevant to the query as a whole. There are a variety of mathematical formulations of TF*IDF, typically using logarithms and some sort of normalization. The off-the-shelf IR components use TF*IDF along with a wide variety of other IR techniques to identify text for Watson to consider.

Large-Scale Text Mining

One resource Watson uses for answering questions is PRISMATIC [2]: a large knowledge-base of information automatically extracted from text. One of the kinds of relationships stored in PRISMATIC is a relationship between an instance and a type. For example, given a sentence like "Aachen was the first major German city to be occupied by the Allies," a relation extractor [3] would detect an instance-type (or "isA") relationship between "Aachen" and "city." This information is then aggregated using a variety of statistics such as frequency and mutual information. This provides components of Watson with insights such as the fact that Aachen is often characterized as a city or a town and less often as a junction. Watson is able to use this information to find candidate answers and also to evaluate candidate answers that have been found in other sources.

Statistical Answer Merging and Ranking

As we mentioned earlier, Watson uses classification learning techniques to train a vector of weights $\boldsymbol{\beta}$ that apply to a candidate's evidence feature vector **x** to obtain an aggregate evidence score $E = \boldsymbol{\beta} * \mathbf{x} = (\Sigma\, \beta_i\, x_i + \beta_0)$, where β_0 is a learned offset parameter. During the project we experimented with many different learning architectures (e.g., decision trees, neural networks and Support Vector Machines), however we consistently found that a relatively simple logistic regression model yielded the best performance. The output of logistic regression is $f(E) = 1/(1+\exp(-E))$, and may be interpreted as a "confidence" (probability of correctness) estimate. While the Watson architecture allows separate models for ranking candidates and estimating confidence, we observed no loss in performance by simply using a single logistic regression model where ranking is based on confidence scores.

An interesting aspect of scoring and ranking candidates is that several textually different candidates may be equivalent in the context of a given question, and hence their evidence scores can be combined. For example, the terms "Lincoln," "Abraham Lincoln" and "President Lincoln" might be equivalent for a question asking about the 16th US President, but not if the question is asking about Ford automobiles. When combining candidates, choosing the best string to represent the combination may be far from trivial. The learned weights obtained by training on unmerged candidates can provide a great benefit, as the highest-ranked candidate can be selected to represent the merged set, unless there is significant evidence to override this. Watson's methodology for merging candidates also includes methods for merging the respective feature vectors. The architecture allows different merging methods to be used on different features, so that certain features may be merged according to max value, while others may be merged via a sum or decaying sum of individual values [4].

Evaluation Metrics

An enormous part of the success of the Watson project has been the team's commitment to frequent and detailed evaluation of our progress. Some of that evaluation is anecdotal: the team looks at specific cases of questions that Watson answers incorrectly (or cases where Watson chooses a correct answer but lacks confidence that the answer is correct). Anecdotal error analysis is useful for developing new ideas for how to improve the system, but judging the potential impact of some idea (or the actual impact, once it is implemented) needs to be conducted in terms of the impact on a sufficiently large sample of questions.

Some of the most interesting metrics that are used to evaluate Watson's effectiveness at answering questions are:

- Accuracy: Percent of questions with correct answer in first place.
- Precision@70%: Percent of questions with correct answer in first place on the 70% of questions with the highest confidence scores.
- Average precision from 30% to 80%: Average value across the range from 30% to 80% of the percent of questions with correct answer in first place when the set is restricted the with the highest confidence scores.
- Ideal uniform earnings fraction: The maximum across all possible threshold values of the "uniform earnings fraction" at that threshold. Uniform earnings fraction at a threshold is defined as the number of questions for which the top ranked answer is correct and has a confidence greater than the threshold divided by the total number of questions for which the top ranked answer has a confidence greater than the threshold.
- Binary Recall: Percent of questions with the correct answer somewhere in the answer list.
- Binary Recall @ K (for K=5, 10, 25, 50, and 100): Percent of questions with the correct answer in the top K answers for that question.

Accuracy is a simple and intuitive metric that directly measures how well a system identifies the correct answer. It is very useful for component developers working on development data because changes in accuracy can be clearly and unambiguously associated with specific questions that were gained or lost.

However, accuracy does not measure all of the aspects of the system's capabilities that we would want to measure. One key issue that accuracy does not measure is how effective the system's confidence scores are at gauging whether a particular answer is right. Precision@70, average precision from 30% to 80%, and ideal uniform earnings fraction are metrics that do account for confidence. Of these, ideal uniform earnings fraction most directly measures what is needed to win at *Jeopardy!*, since *Jeopardy!* provides the same amount of credit for answering a question correctly as the penalty it provides for answering a question incorrectly. In other domains in which the ability of the system to determine a level of confidence is important, other metrics may better reflect how useful the system is.

Figure 2 (which also appears in [5]) shows a confidence curve for two configurations of the Watson 1.0 system: one that includes the set of components referred to as "TyCor" [5] and one that does not. Each point on this graph shows the percentage of questions that the system answers correctly (the vertical value) when it attempts to answer the specified percentage of questions (the horizontal value) for which it has the highest confidence in the correct answer. The blue line shows how effective the full Watson 1.0 system is at answering questions and judging confidence in its answers; the red line shows how well the ablated Watson 1.0 system without the TyCor components performs. The far right end-points of the graph show the accuracy: the percent answered correctly when Watson attempts to answer 100% of all questions. The vertical value at the 70 point on the horizontal axis is the precision@70 for the two configurations. The average vertical value from the 30 to the 80 points on the horizontal axis is the average precision from 30% to 80%. As you can see on the graph, the blue line is higher throughout the range: no matter what percentage of question Watson tries to answer, it gets more of those questions right with the TyCor components than it does without them. Furthermore, the difference between the lines is bigger at the 70% value than it is at the 100% value, suggesting that the TyCor components are useful not only for selecting answers for a particular question but also for assessing the confidence in the selected answer: these components provide a bigger boost to how precisely Watson can answer questions if Watson is able to avoid answering the 30% for which it is least confident.

Another aspect of the system's effectiveness is the ability of the system to provide correct answers that are not the top answer. This ability is not particularly important for playing *Jeopardy!* because a *Jeopardy!* player generally only provides the answer that they like best, not any other answer. However, in other applications, a question answering system may provide a list of answers with supporting evidence for each. For those applications, the ability to find the right answer and rank it near the top of the list can be extremely valuable. Even in *Jeopardy!*, binary recall is useful for tracking the progress of Watson's ability to identify the correct answer as a candidate answer, independently of Watson's ability to rank that answer correctly. Since Watson has distinct components for finding answers, it is useful to have metrics that track the progress of those components.

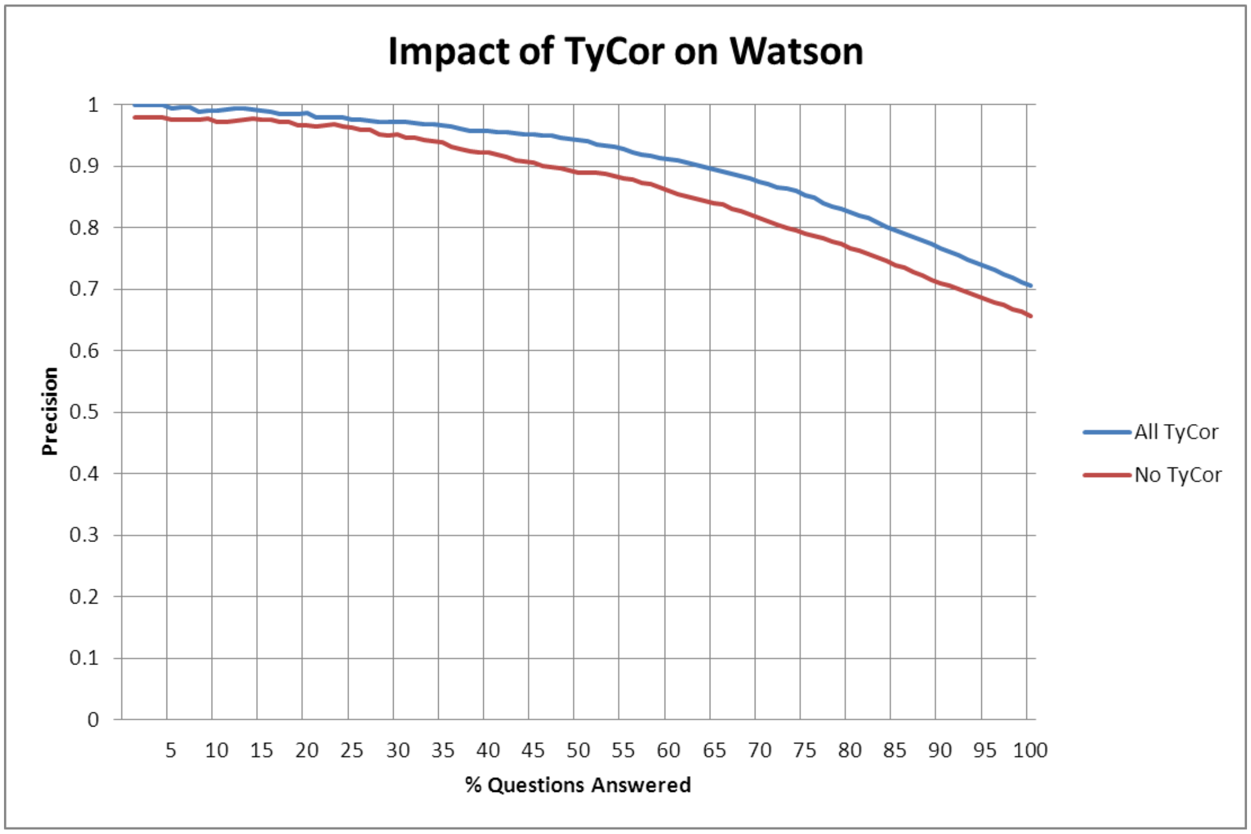

Figure 2: Precision vs. percentage of questions answered

Jeopardy! Game Strategy

Watson also contains a significant body of code for making *Jeopardy!* strategy decisions, such as wagering on a Daily Double (DD) and deciding whether to attempt to answer a clue. While this does not help in computing correct answers, it nonetheless makes Watson a much more formidable *Jeopardy!* contestant. As described in [6], we employed a variety of advanced statistical techniques to develop Watson's strategy algorithms. Our approach is founded upon estimating Watson's winning chances in a given game state, using a stochastic simulation model that provides a reasonably faithful emulation of likely events when Watson plays against two humans. We can perform live Monte-Carlo simulations to calculate whether Watson should buzz near the end of the game, and its best wager in Final Jeopardy. We also used a nonlinear regression model, trained over millions of offline simulation trials, to compute DD wagers in cases where the Monte-Carlo analysis is too slow to use in live play. Finally, Watson can seek out and find the DDs faster than humans by using Bayesian inference: this combines the historic frequencies of DD placement with evidence from revealed clues to compute where the DDs are most likely to be located. These methods give Watson a distinct edge in the strategic aspects of *Jeopardy!*, since human contestants cannot match the speed and precision of Watson's strategy calculations during live play.

Future Work

The Watson research team is no longer studying *Jeopardy!*, but many of the mathematical issues described here still apply to other challenge problems. Watson's mechanisms for retrieving information, mining data from large volumes of text, and selecting answers using a statistical classifier are all relevant to a wide variety of practical applications.

In many cases, different evaluation metrics are more important: for example, in *Jeopardy!* no credit is awarded when Watson has the right answer as its second choice. However, for many real-world information gathering tasks, having the right answer along with relevant evidence among a list of five or ten possible answers provides value to end users. Thus some important metrics for real-world applications should focus not only on the accuracy and confidence of Watson's top answer, but also on the quality of answers that it ranks near the top of its list. Providing convincing evidence for an answer is an important task for a real-world Watson system. Effectiveness on this task is challenging to measure because it is very subjective; addressing this challenge will be key to the future success of Watson, because a Watson-based tool for helping real-world decision makers will not be particularly useful if it finds a lot of correct answers but cannot convince the users that those answers are correct.

Looking beyond near-term extensions of Watson, we see the field of NLP research as being poised for tremendous progress over the next decade, building upon recent advances in Watson and other impressive QA systems. With ever-increasing language data, algorithm sophistication and compute power, it now seems plausible that computers could be built in the next decade that outperform humans on virtually any fact-based QA task. Such QA systems will also make use of vast and deep structured knowledge bases, enabling much deeper reasoning and comprehension of the material contained in evidence sources. Computers may play an important role in automatically constructing such knowledge bases, as evidenced by today's systems such as PRISMATIC and CMU's Never-Ending Language Learner [7]. The progress that we envision is likely to result from the same types of quantitative, massive data-driven approaches that went into building Watson; the rate of progress in NLP due to such approaches may in fact be accelerating.

References

[1] J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty, Finding needles in the haystack: Search and candidate generation. IBM J. Res. & Dev., vol. 56, no. 3/4, 2012.

[2] J. Fan, A. Kalyanpur, D. C. Gondek, and D. Ferrucci, Automatic knowledge extraction from documents, IBM J. Res. & Dev., vol. 56, no. 3/4, 2012.

[3] C. Wang, A. Kalyanpur, J. Fan, B. K. Boguraev, and D. C. Gondek, Relation extraction and scoring in DeepQA. IBM J. Res. & Dev., vol. 56, no. 3/4, 2012.

[4] 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, 2012.

[5] J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. Ferrucci, D. Gondek, L. Zhang, and H. Kanayama. Typing candidate answers using type coercion. IBM J. Res. & Dev., vol. 56, no. 3/4, 2012.

[6] G. Tesauro, D. Gondek, J. Lenchner, J. Fan and J. Prager. Simulation, Learning and Optimization Techniques in Watson's Game Strategies. IBM J. Res. & Dev., vol. 56, no. 3/4, 2012.

[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr. and T.M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In: Proceedings of AAAI, 2010.