RC25489 (WAT1409-048) September 17, 2014
Computer Science

IBM Research Report

WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information

Adam Lally, Sugato Bachi, Michael A. Barborak, David W. Buchanan, Jennifer Chu-Carroll, David A. Ferrucci*, Michael R. Glass, Aditya Kalyanpur, Erik T. Mueller, J. William Murdock, Siddharth Patwardhan, John M. Prager, Christopher A. Welty
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

* This work was done while at the IBM Thomas J. Watson Research Center

Research Division
Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich

WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information

Adam Lally[1], **Sugato Bagchi**[1], **Michael A. Barborak**[1], **David W. Buchanan**[1], **Jennifer Chu-Carroll**[1], **David A. Ferrucci**[2], **Michael R. Glass**[1], **Aditya Kalyanpur**[1], **Erik T. Mueller**[1], **J. William Murdock**[1], **Siddharth Patwardhan**[1], **John M. Prager**[1] **Christopher A. Welty**[1]

[1]IBM Research and IBM Watson Group
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

[2] This work was done while at the IBM Thomas J. Watson Research Center

Abstract

We present WatsonPaths™, a novel system that can answer scenario-based questions, for example medical questions that present a patient summary and ask for the most likely diagnosis or most appropriate treatment. WatsonPaths builds on the IBM Watson™ question answering system that takes natural language questions as input and produces precise answers along with accurate confidences as output. WatsonPaths breaks down the input scenario into individual pieces of information, asks relevant subquestions of Watson to conclude new information, and represents these results in a graphical model. Probabilistic inference is performed over the graph to conclude the answer. On a set of medical test preparation questions, WatsonPaths shows a significant improvement in accuracy over the base Watson QA system. We also describe how WatsonPaths can be used in a collaborative application to help users reason about complex scenarios.

1 Introduction

IBM Watson™ is a question answering system that takes natural language questions as input and produces precise answers along with accurate confidences as output (Ferrucci et al., 2010). Watson defeated two of the best human players on the quiz show Jeopardy! in 2011.

Watson has been described (Kelly and Hamm, 2013) as an opening to the era of *cognitive computing*: computers that interact in a natural way with humans, assist human cognition, and learn and improve from interaction. To fulfill this vision, further advances are required. One such advance is the ability to answer more complex questions. Another is to allow the user to understand and participate in the question answering process.

Consider the following questions, one from medicine and one from taxation:

> A 32-year-old woman with type 1 diabetes mellitus has had progressive renal failure. Her hemoglobin concentration is 9 g/dL. A blood smear shows normochromic, normocytic cells. What is the problem?

> I inherited real-estate from a relative who died 5 years ago via a trust that was created before his death. The property was sold this year after dissolution of the trust, and the money was put in a Roth-IRA. Which tax form(s) do I need to file?

We asked domain experts to describe their approach to solving such questions. An example from medical experts is shown in Figure 1. Many drew a graph of initial signs and symptoms leading to their most likely possible causes and connecting them to a final conclusion. We noticed that their reasoning process often resembled probabilistic inference.

At the core of Watson's question answering is a suite of algorithms that match passages containing candidate answers to the original question. These algorithms have been described in a series of articles (Chu-Carroll et al., 2012; Ferrucci, 2012; Gondek et al., 2012; Lally et al., 2012; McCord et al., 2012; Murdock et al., 2012a; Murdock et al., 2012b). But, when questions involve complex

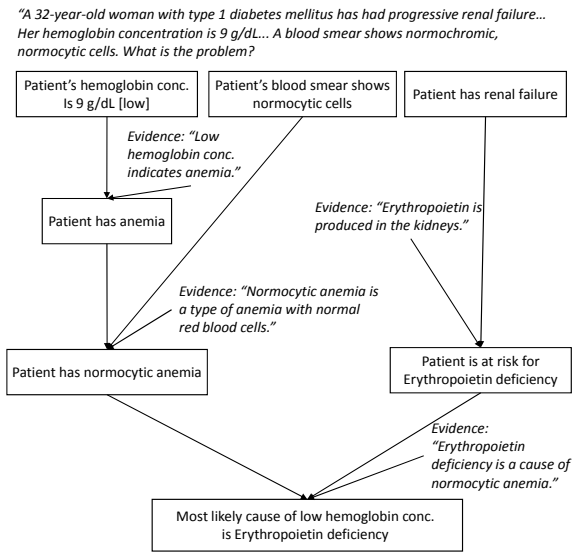

Figure 1: Simple Diagnosis Graph for a Patient with Erythropoietin Deficiency

scenarios, as in the above examples, passage matching by itself is often insufficient to locate the answer. This is because scenario-based question answering requires integrating and reasoning over information from multiple sources. Furthermore, we must often apply general knowledge to a specific case, as in a medical scenario about a patient.

In this paper, we present a new approach that builds on Watson's strengths and is in line with the human reasoning process we observed. We break down the input scenario into individual pieces of information, ask relevant *subquestions* to conclude new information, and combine these results into an *assertion graph*. We then perform probabilistic inference over the graph to conclude the answer to the overall question. This process is repeated to extend the graph until a stopping condition is met. Because we use Watson to answer the subquestions, and because we attempt to construct paths of inference to a final answer, we call our system *WatsonPaths*™.

In the WatsonPaths graph, the evidence is drawn from a variety of sources including general knowledge encyclopedias, domain-specific books, structured knowledge bases, and semi-structured knowledge bases. We were motivated by the desire to design a solution that could harness Watson, and we observed that each edge in this graph could correspond to a question asked of Watson.

An added dimension to WatsonPaths is the ability to interact with the user. The original Watson system that won Jeopardy! was largely non-interactive. For many applications, it is important to engage the user in the problem solving process. WatsonPaths has the ability to elicit user input at multiple points of question answering and decision making to clarify questions, to judge evidence, and to ask new questions. A key advantage of this approach is that user feedback can be used as training data to improve both Watson and WatsonPaths.

2 WatsonPaths Medical Use Case

Although WatsonPaths enables general-purpose scenario-based question answering, we decided to start by focusing our attention on the medical domain. We focused on the problem of patient scenario analysis, where the goal is typically a diagnosis or a treatment recommendation.

To explore this kind of problem solving, we obtained a set of medical test preparation questions. These are multiple choice medical questions based on an unstructured or semi-structured natural language description of a patient. Although WatsonPaths is not restricted to multiple choice questions, we saw multiple choice questions as a good starting point for development. Many of these questions involve diagnosis, either as the entire question, as in the previous medical example, or as an intermediate step, as in the following example:

> A 63-year old patient is sent to the neurologist with a clinical picture of resting tremor that began 2 years ago. At first it was only on the left hand, but now it compromises the whole arm. At physical exam, the patient has an unexpressive face and difficulty in walking, and a continuous movement of the tip of the first digit over the tip of the second digit of the left hand is seen at rest. What part of his nervous system is most likely affected?

For this question, it is useful to diagnose that the patient has Parkinson's disease before determining which part of his nervous system is most likely affected. These multi-step inferences are a natural fit for the graphs that WatsonPaths constructs. In this example, the diagnosis is the *missing link* on the way to the final answer.

3 Scenario-based Question Answering

In scenario-based question answering, the system receives a scenario description that ends with a *punchline question*. For instance, the punchline question in the Parkinson's example is "What part of his nervous system is most likely affected?" Instead of treating the entire scenario as one monolithic question as would Watson, WatsonPaths explores multiple facts in the scenario in parallel and reasons with the results of its exploration as a whole to arrive at the most likely conclusion regarding the punchline question. The architecture of WatsonPaths is shown in Figure 2.

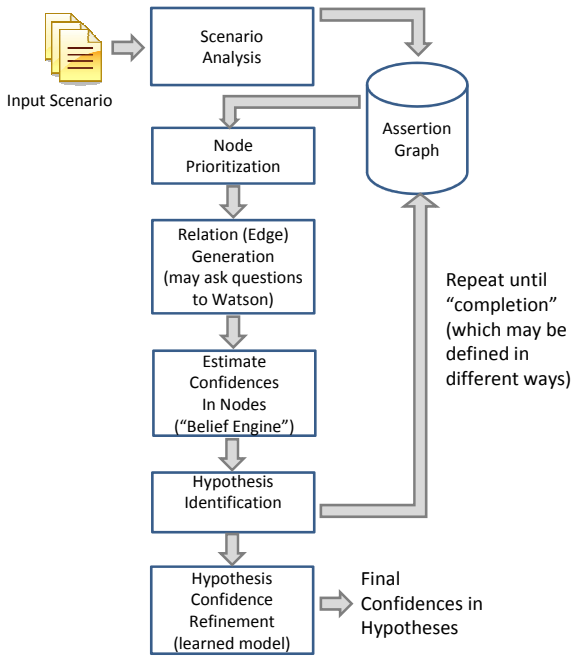

Figure 2: Scenario-based Question Answering Architecture

3.1 Scenario Analysis

The first step in the pipeline is *scenario analysis*, where we identify *factors* in the input scenario that may be of importance. In the medical domain, the factors may include demographics ("32-year old woman"), pre-existing conditions ("type 1 diabetes mellitus"), signs and symptoms ("progressive renal failure"), and test results ("hemoglobin concentration is 9 g/dL," "normochromic cells," "normocytic cells"). The extracted factors become nodes in a graph structure called the *assertion graph*. The assertion graph structure is defined in Section 4, while more details of the scenario analysis process are given in Section 5.

3.2 Node Prioritization

The next step is *node prioritization*, where we decide which nodes in the graph are most important for solving the problem. In a small scenario like this example, we may be able to explore everything, but in general this will not be the case. Factors that affect the priority of a node may include the system's confidence in the node assertion or the system's estimation of how fruitful it would be to expand a node. For example, normal test results and demographic information are generally less useful for starting a diagnosis than symptoms and abnormal test results.

3.3 Relation Generation

The *relation generation* step, which is described in more detail in Section 6, builds the assertion graph. We do this primarily by asking Watson questions about the factors. In medicine we want to know the causes of the findings and abnormal test results that are consistent with the patient's demographic information and normal test results. Given the scenario in the Introduction, we could ask, "What does type 1 diabetes mellitus cause?" We use a medical ontology to guide the process of formulating subquestions to ask Watson. Relevant factors may also be combined to form a single, more targeted question. Because in this step we want to emphasize recall, we take several of Watson's highly-ranked answers. The exact number of answers taken, or the confidence threshold, are parameters that must be tuned. Given a set of answers, we add them to the graph as nodes, with edges from nodes that were used in questions to nodes that were answers. The edge is labeled with the relation used to formulate the question (like *causes* or *indicates*), and the strength of the edge is initially set to Watson's confidence in the answer. Although Watson is the primary way we add edges to the graph, WatsonPaths allows for any number of *relation generator* components to post edges to the graph.

3.4 Belief Computation

Once the assertion graph has been expanded in this way, we recompute the confidences of nodes in the graph based on new information. We do this using probabilistic inference systems that are described in Section 7. The inference systems take a holistic view of the assertion graph and try to reconcile the results of multiple paths of exploration.

3.5 Hypothesis Identification

As Figure 2 shows, this process can go through multiple iterations, during which the nodes that were the answers to the previous round of questions can be used to ask the next round of questions, producing more nodes and edges in the graph. After each iteration we may do *hypothesis identification*, where some nodes in the graph are identified as potential final answers to the punchline question (for example, the most likely diagnoses of a patient's problem). In some situations hypotheses may be provided up front—a physician may have a list of competing diagnoses and want to explore the evidence for each. But in general the system needs to identify these. Hypothesis nodes may be treated differently in later iterations. For instance, we may attempt to do backward chaining from the hypotheses, asking Watson what things, if they were true of the patient, would support or refute a hypothesis. The process may terminate after a fixed number of iterations or based on some other criterion like confidence in

the hypotheses.

While hypothesis identification is part of WatsonPaths, it is not described in detail in this paper. In the system that generates the results we present in Section 10, no hypothesis identification is necessary because the multiple choice answers are provided. That system always does one iteration of expansion, both forward from the identified factors and backward from the hypotheses, before stopping.

3.6 Hypothesis Confidence Refinement

As described so far, WatsonPath's confidence in each hypothesis depends on the strengths of the edges leading to it, and since our primary relation (edge) generator is Watson, the hypothesis confidence depends heavily on the confidence of Watson's answers. Having good answer confidence depends on having a representative set of question/answer pairs with which to train Watson. The following question arises: What can we do if we do not have a representative set of question/answer pairs, but we do have training examples for entire scenarios (e.g., correct diagnoses associated with patient scenarios)? To leverage the available scenario-level ground truth, we have built machine learning techniques to learn a refinement of Watson's confidence estimation that produces better results when applied to the entire scenario. This learning process is discussed in Section 8.

3.7 Collaborating with the User

WatsonPaths can run in a completely automated way, as the Watson question answering system did when playing Jeopardy! (This is the case for the results presented in Section 10.) But there are also many interesting possibilities for user interaction at each step in the process. In this way, WatsonPaths exemplifies cognitive computing. Our vision for cognitive computing is that the user and the computer work together to explore a scenario and reach conclusions faster and more accurately than either could do alone. We discuss the collaborative learning aspects of WatsonPaths in Section 9.

4 Assertion Graphs

The core data structure used by WatsonPaths is the *assertion graph*. Figure 3 explains this data structure, along with the visualization that we commonly use for it. Assertion graphs are defined as follows.

A *statement* is something that can be true or false (though its state may not be known). Often we deal with *unstructured statements*, which are natural language expressions like "A 63-year-old patient is sent to the neurologist with a clinical picture of resting tremor that began 2 years ago." WatsonPaths also allows for statements that are structured expressions, namely, a predicate and arguments. Not all natural language expressions can have a

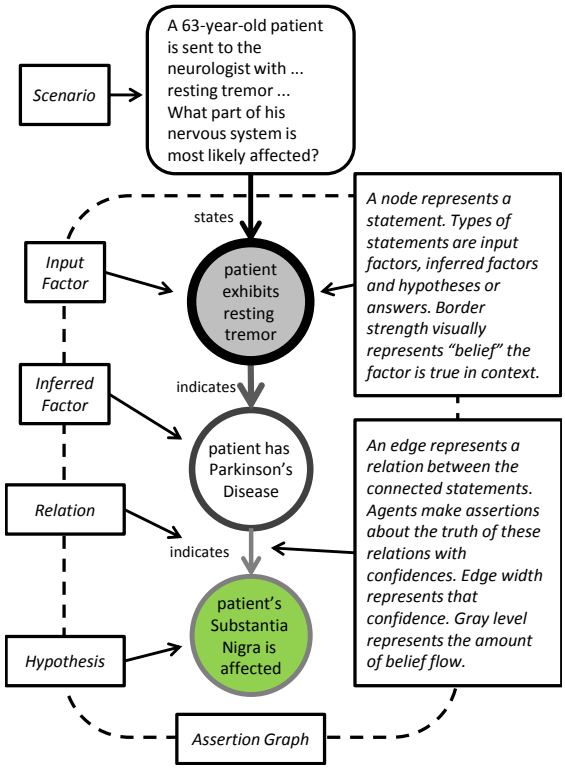

Figure 3: Visualization of an Assertion Graph. By convention, input factors are placed at the top and hypotheses at the bottom with levels of inference factors in between.

truth value. For instance, the string "patient" cannot be true or false; thus it does not fit into the semantics of an assertion graph. WatsonPaths is charitable in interpreting strings as if they had a truth value. For instance, the default semantics of the string "low hemoglobin" is the same as "patient has low hemoglobin."

A *relation* is a named association between statements. Technically, relations are themselves statements, and have a truth value. Each relation has a predicate; for instance in medicine we may say that "Parkinsons *causes* resting tremor" or "Parkinson's *matches* Parkinsonism." Typically we are concerned with relations that may provide evidence for the truth of one statement given another. Although some relations may have special meanings in the probabilistic inference systems, a common semantics for a relation is *indicative* in the following way: "A indicates B" means that the truth of A provides an independent reason to believe that B is true. Section 7 provides more detail on the inference systems.

An *assertion* is a claim that some agent makes about the truth of a statement (including a relation). The assertion records the name of the agent and a confidence value. Assertions may also record provenance information that explains how the agent came to its conclusion.

For the Watson question answering agent, this includes natural language passages that provide evidence for the answer. When the system is collaborating with a user, it is crucial to be able to display evidence to the user.

In the assertion graph, each node represents exactly one statement, and each edge represents exactly one relation. Nodes and edges may have multiple assertions attached to them, one for each agent that has asserted that node or edge to be true.

We often visualize assertion graphs by using a node's border width to represent the confidence of the node, an edge's width to represent the confidence of the edge, and an edge's gray level as the amount of "belief flow" along that edge. Belief flow is described later, but essentially it is how much the value of the head influences the value of the tail. This depends mostly on the confidences of the assertions on the edge.

5 Scenario Analysis

The goal of scenario analysis is to identify information in the natural language narrative of the problem scenario that is potentially relevant to solving the problem. When human experts read the problem narrative, they are trained to extract concepts that match a set of semantic types relevant for solving the problem. In the medical domain, doctors and nurses identify semantic types like chief complaints, past medical history, demographics, family and social history, physical examination findings, labs, and current medications (Bowen, 2006). Experts also generalize from specific observations in a particular problem instance to more general terms used in the domain corpus. An important aspect of this information extraction is to identify the *semantic qualifiers* associated with the clinical observations (Chang et al., 1998). These qualifiers could be temporal (e.g.,"pain started two days ago"), spatial ("pain in the epigastric region"), or other associations ("pain after eating fatty foods"). Implicit in this task is the human's ability to extract concepts and their associated qualifiers from the natural language narrative. For example, the above qualifiers might have to be extracted from the sentence "The patient reports pain, which started two days ago, in the epigastric region especially after eating fatty foods."

The computer system needs to perform a similar analysis of the narrative. We use the term *factor* to denote the potentially relevant observations along with their associated semantic qualifiers. Reliably identifying and typing these factors, however, is a difficult task, because medical terms are far more complex than the kind of named entities typically studied in natural language processing. Our scenario analytics pipeline attempts to address this problem with the following major processing steps:

1. The analysis starts with syntactic parsing of the natural language. This creates a dependency tree of syntactically linked terms in a sentence and helps to associate terms that are distant from each other in the sentence.

2. The terms are mapped to a dictionary to identify concepts and their semantic types. For the medical domain, our dictionary is derived from the UMLS Metathesaurus (National Library of Medicine, 2009), Wikipedia redirects, and medical abbreviation resources. The concepts identified by the dictionary are then typed using the UMLS Semantic Network, which consists of a taxonomy of biological and clinical semantic types like *Anatomy*, *SignOrSymptom*, *DiseaseOrSyndrome*, and *TherapeuticOrPreventativeProcedure*. In addition to mapping the sequence of tokens in a sentence to the dictionary, the dependency parse is also used to map syntactically linked terms. For example "...stiffness and swelling in the arm and leg" can be mapped to the four separate concepts contained in that phrase.

3. The syntactic and semantic information identified above are used by a set of predefined rules to identify important relations. Negation is commonly used in clinical narratives and needs to be accurately identified. Rules based on parse features identify the negation trigger term and its scope in a sentence. Factors found within the negated scope can then be associated with a negated qualifier. Another example of rule-based annotation is lab value analysis. This associates a quantitative measurement to the substance measured and then looks up reference lab value ranges to make a clinical assessment. For example "hemoglobin concentration is 9 g/dL" is processed by rules to extract the value, unit, and substance and then assessed to be "low hemoglobin" by looking up a reference. Next, the clinical assessment is mapped by the dictionary to the corresponding clinical concept.

At this point, we should have all the information to identify factors and their semantic qualifiers. We have to contend, however, with language ambiguities, errors in parsing, a noisy and non-comprehensive dictionary, and a limited set of rules. If we were to rely solely on a rule-based system, then the resulting factor identification would suffer from a compounding of errors in these components. To address this issue, we employ machine learning methods to learn clinical factors and their semantic qualifiers in the problem narrative. We obtained the ground truth by asking medical students to annotate clinical factor spans and their semantic types. They also annotated semantic qualifier spans and linked them to

factors as attributive relations.

The machine learning system is comprised of two sequential steps:

1. A conditional random field (CRF) model (Lafferty et al., 2001) learns the spans of text that should be marked as one of the following factor types: *finding*, *disease*, *test*, *treatment*, *demographics*, negation, or a semantic qualifier. Features used for training the CRF model are lexical (lemmas, morphological information, part-of-speech tags), semantic (UMLS semantic types and groups, demographic and lab value annotations), and parse-based (features associated with dependency links from a given token). A token window size of 5 (2 tokens before and after) is used to associate features for a given token. A BIO tagging scheme is used by the CRF to identify entities in terms of their token spans and types.

2. A maximum entropy model then learns the relations between the entities identified by the CRF model. For each pair of entities in a sentence, this model uses lexical features (within and between entities), entity type, and other semantic features associated with both entities, and parse features in the dependency path linking them. The relations learned by this model are *negation* and *attributeOf* relations linking negation triggers and semantic qualifiers (respectively) to factors.

The combined entity and relation identification models have a precision of 71% and recall of 65% on a blind evaluation set of patient scenarios found in medical test preparation questions. We are currently exploring joint inference models and identification of relations that span multiple sentences using coreference resolution.

6 Relation Generation

The scenario analysis component described in the previous section extracts pertinent factors related to the patient from the scenario description. At this stage, the assertion graph consists of the full scenario, individual scenario sentences, and the extracted factors. An *indicates* relation is posted from a source node (e.g., a scenario sentence node) to a target node whose assertion was derived from the assertion in the source node (e.g., a factor extracted from that sentence). In addition, a set of hypotheses, if given, are posted as goal nodes in the assertion graph.

The task of the relation generation component is to (1) expand the graph by inferring new facts from known facts in the graph and (2) identify relationships between nodes in the graph (like *matches* and *contraindicates*) to help with reasoning and confidence estimation. We begin by discussing how we infer new facts for graph expansion.

6.1 Expanding the Graph with Watson

In medical problem solving, experts reason with chief complaints, findings, medical history, demographic information, and so on, to identify the underlying causes for the patient's problems. Depending on the situation, they may then proceed to propose a test whose results will allow them to distinguish between multiple possible problem causes, or identify the best treatment for the identified cause, and so on.

Motivated by the medical problem solving paradigm, WatsonPaths first attempts to make a diagnosis based on factors extracted from the scenario. The graph is expanded to include new assertions about the patient by asking questions of a version of the Watson question answering system adapted for the medical domain (Ferrucci et al., 2013). WatsonPaths takes a two-pronged approach to medical problem solving by expanding the graph forward from the scenario in an attempt to make a diagnosis, and then linking high confidence diagnoses with the hypotheses. The latter step is typically done by identifying an important relation expressed in the punchline question (e.g., "What is the most appropriate treatment for this patient" or "What body part is most likely affected?"). This approach is a logical extension of the open-domain work of Prager et al. (2004), where in order to build a profile of an entity, questions were asked of properties of the entity and constraints between the answers were enforced to establish internal consistency.

The graph expansion process of WatsonPaths begins with automatically formulating questions related to high confidence assertions, which in our graphs represent statements WatsonPaths believes to be true to a certain degree of confidence about the patient. These statements may be factors, as extracted and typed by the algorithm described in Section 5, or combinations of those factors.

To determine what kinds of questions to ask, WatsonPaths can use a domain model that tells us what relations form paths between the semantic type of a high confidence node and the semantic type of a hypothesis like a diagnosis or treatment. For the medical domain, we created a model that we called the *Emerald*, which is shown in Figure 4. (Notice the resemblence to an emerald.) The Emerald is a small model of entity types and relations that are crucial for diagnosis and for formulating next steps.

We select from the Emerald all relations that link the semantic type of a high-confidence source node to a semantic type of interest. The relations and the high-confidence nodes then form the basis of instantiating the target nodes, thereby expanding the assertion graph. To instantiate the target nodes, we issue WatsonPaths subquestions to Watson. All answers returned by Watson that score above a pre-determined threshold are posted as target nodes in the inference graph. A relation edge

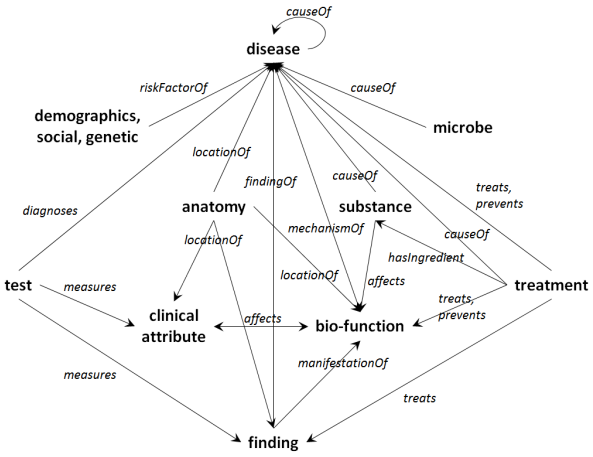

Figure 4: The Emerald

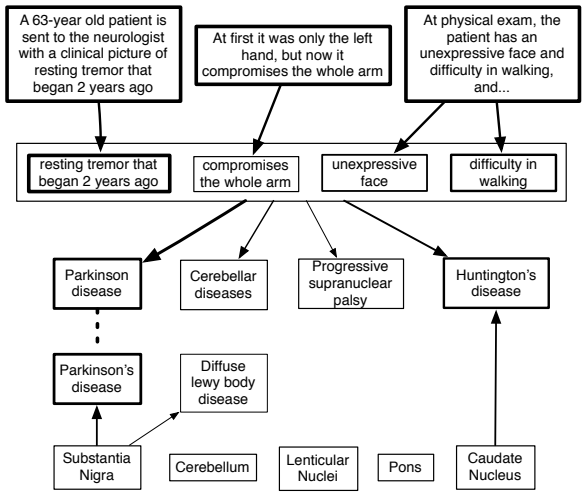

Figure 5: WatsonPaths Graph Expansion Process

is posted from the source node to each new target node where the confidence of the relation is Watson's confidence in the answer in the target node.

In addition to asking questions from scenario factors, WatsonPaths may also expand backwards from hypotheses. The premise for this approach is to explore how a hypothesis fits in with the rest of the inference graph. If one hypothesis is found to have a strong relationship with an existing node in the assertion graph, then the probabilistic inference mechanisms described in 7 allow belief to flow from known factors to that hypothesis, thus increasing the system's confidence in that hypothesis.

Figure 5 illustrates the WatsonPaths graph expansion process. The top two rows of nodes and the edges between them show a subset of the WatsonPaths assertion graph after scenario analysis, with the second row of nodes representing some clinical factors extracted from the scenario sentences.

The graph expansion process identifies the most confident assertions in the graph, which include the four clinical factor nodes extracted from the scenario. These four nodes are all typed as *finding*s, so they are aggregated into a single *finding* node for the purpose of graph expansion. For a *finding* node, the Emerald proposes a single *findingOf* relation that links it to a *disease*. This results in the formulation of the subquestion "What disease causes resting tremor that began 2 years ago, compromises the whole arm, unexpressive face, and difficulty in walking?" whose answers include *Parkinson disease*, *Huntington's disease*, *cerebellar disease*, and so on. These answer nodes are added to the graph and some of them are shown in the third row of nodes in Figure 5.

In the reverse direction, WatsonPaths explores relationships between hypotheses to nodes in the existing graph based on the punchline question in the scenario, which in this case is "What part of his nervous system is mostly likely affected?" Assuming each hypothesis to be true, the system formulates subquestions to link it to the assertion graph. Consider *Substantia nigra*. WatsonPaths can ask "In what disease is substantia nigra most likely affected?" A subset of the answers to this question, including *Parkinson's disease* and *Diffuse Lewy body disease* are shown in the fourth row of nodes in Figure 5.

6.2 Matching Graph Nodes

When a new node is added to the WatsonPaths assertion graph, we compare the assertion in the new node to those in existing nodes to ensure that equivalence relations between nodes are properly identified. This is done by comparing the statements in those assertions: for unstructured statements, whether the statements are lexically equivalent, and for structured statements, whether the predicates and their arguments are the same. A more complex operation is to identify when nodes contain assertions that *may* be equivalent to the new assertion.

We employ an aggregate of term matchers (Murdock et al., 2012a) to match pairs of assertions. Each term matcher posts a confidence value on the degree of match between two assertions based on its own resource for determining equivalence. For example, a WordNet-based term matcher considers terms in the same synset to be equivalent, and a Wikipedia-redirect-based term matcher considers terms with a redirect link between them in Wikipedia to be a match. The dotted line between *Parkinson disease* and *Parkinson's disease* in Figure 5 is posted by the UMLS-based term matcher, which considers variants for the same concept to be equivalent.

7 Confidence and Belief

Once the assertion graph is constructed, and some questions and answers are posted, there remains the problem

of confidence estimation. We develop multiple models of inference to address this step.

7.1 Belief Engine

One approach to the problem of inferring the correct hypothesis from the assertion graph is probabilistic inference over a graphical model (Pearl, 1988). We refer to the component that does this as the *belief engine*.

Although the primary goal of the belief engine is to infer confidences in hypotheses, it also has two secondary goals. One is to infer belief in unknown nodes that are not hypotheses. These intermediate nodes may be important intermediate steps toward an answer; by assigning high confidences to them in the main loop, we know to assign them high priority for subquestion asking. Another secondary goal is to support the user interface (see Section 9). Among inference algorithms that perform well in terms of accuracy and other metrics, we try to make choices that will make the flow of belief intuitive for users. This facilitates the gathering of better opportunistic annotations, which improves future performance.

To execute the belief engine, we first make a working copy of the assertion graph that we call the *inference graph*. A separate graph is used so that we can make changes without losing information that might be useful in later steps of inference. For instance, we might choose to merge nodes or reorient edges. Once the inference graph has been built, we run a probabilistic inference engine over the graph to generate new confidences. Each node represents an assertion, so it can be in one of two states: true or false ("on" or "off"). Thus a graph with k nodes can be in 2^k possible states. The inference graph specifies the likelihoods of each of these states. The belief engine uses these likelihoods to calculate the marginal probability, for each node, of it being in the true state. This marginal probability is treated as a confidence. Finally, we read confidences and other data from the inference graph back into the assertion graph.

There are some challenges in applying probabilistic inference to an assertion graph. Most tools in the inference literature were designed to solve a different problem, which we will call the *classical inference problem*. In this problem, we are given a training set and a test set that can be seen as samples from a common joint distribution. The task is to construct a model that captures the training set (for instance, by maximizing the likelihood of the training set), and then apply the model to predict unknown values in the test set. Arguably the greatest problem in the classical inference task is that the structure of the graphical model is underdetermined; a large space of possible structures needs to be explored. Once a structure is found, adjusting the *strengths* is relatively easier, because we know that samples from the training set are sampled from a consistent joint distribution.

In WatsonPaths, we face a different set of problems. The challenge is not to construct a model from training data, but to use a very noisy, already constructed model to do inference. Training data in the classical sense is absent or very sparse; all we have are correct answers to some scenario-level questions. An advantage is that a graph structure is given. A disadvantage is that the graph is noisy. Furthermore, it is not known that the confidences on the edges necessarily correspond to the optimal edge strengths. (In the next section, we address the problem of learning edge strengths.) Thus we have the problem of selecting a *semantics*—a way to convert the assertion graph into a graph over which we can do optimal probabilistic inference to meet our goals.

After much experimentation, the primary semantics used by the belief engine is the *indicative* semantics: If there is a directed relation from node A to node B with strength x, then A provides an independent reason to believe that B is true with probability x. Some edges are classified as *contraindicative*; for these edges, A provides an independent reason to believe that B is false with probability x. The independence means that multiple parents R can easily be combined using a noisy-OR:

$$(1 - \prod_{r \in R} (1 - r)) = \bigoplus_{r \in R} r$$

The graph, so interpreted, forms a noisy-logical Bayesian network (Yuille and Lu, 2007). The strength of each edge can be interpreted as an *indicative power*, a concept related to *causal power* (Cheng, 1997), with the difference that we are semantically agnostic as to the true direction of the causal relation. Formally, the probability of a node being "on" (true) is given by

$$P(x|R_x, Q_x) = \left[\bigoplus_{r \in R_x} (s_r p_r)\right] \left[1 - \bigoplus_{q \in Q_x} (s_q p_q)\right]$$

where $P(x)$ is the probability of node x being on, R_x is the set of indicative parents of x, and Q_x is the set of contraindicative parents. The parent's state is represented by p_r: 1 if the parent is on, and 0 otherwise. The value s_r represents the strength of the edge from the parent to x. In other words, the probability that a node x is on is the noisy-OR of its active indicative parent edge strengths combined via a noisy-AND-NOT with the noisy-OR of its active contraindicative parent edge strengths.

For instance, if the node *resting tremor* indicates *Parkinson disease* with strength 0.8, and the node *difficulty in walking* indicates *Parkinson disease* with power 0.4, then the probability of *Parkinson disease* will be $(1 - (1 - 0.8)(1 - 0.4)) = 0.88$. If so, then the edge with strength 0.9 to *Parkinson's disease* will fire with

probability $0.88 \cdot 0.9 = 0.792$. In this way, probabilities can often multiply down simple chains. Inference must be more sophisticated to handle the graphs we see in practice, but the intuition is the same.

An example that adds sophistication to the inference is an "exactly one" constraint that can be optionally added to multiple-choice questions. This constraint assigns a higher likelihood to assignments in which exactly one multiple choice answer is true. Because of these kinds of constraints, and because of the fact that the graphs contain directed and undirected cycles, we cannot simply calculate the probabilities in a feed-forward manner. To perform inference we use Metropolis-Hastings sampling over a factor graph representation of the inference graph. This has the advantage of being a very general approach—the inference engine can easily be adapted to a new semantics—and also allows an arbitrary level of precision given enough processing time.

Users and annotators report that they find the indicative semantics intuitive, and it performs at least as well as other semantics in experiments. One of the first semantics we tried was undirected pairwise Markov random fields. These performed poorly in practice. We hypothesize that this is because important information is contained in the direction of the edges that Watson returns: Asking about A and getting B as an answer is different from asking about B and getting A as an answer. An undirected model loses this information.

The indicative semantics is a default, basic semantics. The ability to reason over arbitrary relations makes the indicative semantics robust, but it is easy to construct examples in which the indicative semantics is not strictly correct. For instance, "fever is a finding of Lyme disease" may be correctly true with high confidence, but this does not mean that fever provides an independent reason to believe that Lyme disease is present, with high probability. Fever is caused by many things, each of which could explain it. We are currently working on adding a causal semantics in which we use a noisy-logical Bayesian network, but belief flows from causes to effects, rather than from factors to hypotheses. Edges are oriented according to the types of the nodes: Diseases cause findings but not vice-versa. Currently this does not lead to detectable improvement in accuracy and we expect that we need to improve the precision of the rest of the system before it will show impact.

7.2 Closed-Form Inference

The inference method in Section 7.1 obtains the strengths of edges directly from the confidence values on Watson's answers to subquestions, which depend on having a representative training set of subquestion/answer pairs. We have also developed inference methods where each edge has a feature vector (produced by Watson or any other relation generator) and we express the confidence in each hypothesis as a closed-form, parameterized expression over the feature values. We can then optimize the parameters on a training set of scenarios and correct diagnoses (see Section 8).

To illustate the idea we describe in detail one such model, the *Noisy-OR Model*, which is based on the same intuition is the indicative semantics just described.

We first convert the assertion graph to a directed acyclic graph (DAG). The assertion graph is not, in general, free of cycles. Additionally, the assertion graph contains matching relations, which are undirected. To form a DAG, the nodes in the assertion graph are first clustered by these matching relations, and then cycles are broken by applying heuristics to re-orient edges to point from factors to hypotheses.

The confidence in factors extracted by Scenario Analysis is 1.0. For all other nodes the confidence is defined recursively in terms of the confidences of the parents and the confidence of the edges produced by the QA system. Let the set of parents in the DAG for a node n be given by $a(n)$. The feature vector the QA system gives for one node, m, indicating another, n, is given by $\lambda(m, n)$. Then the confidence for a non-factor node is given below. The learned weight vector for the QA features is $\vec{q}$.

$$P(n) = \bigoplus_{a_i \in a(n)} \sigma(\vec{q} \cdot \lambda(a_i, n)) \cdot P(a_i)$$

where $\sigma(x)$ denotes the sigmoid function.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The noisy-OR combination is most sensible when the sources of evidence are independent. When there are two edges leading from the same inference node (which may be the result of merging two or more assertion graph nodes) to the node under consideration, these edges are combined by max rather than noisy-OR.

In addition to the Noisy-OR model, we have also developed the following:

- The *Edge Type* variant of the Noisy-OR model considers the type of the edge when propagating confidence from parents to children. The strength of the edge according to the QA model is multiplied by a per-edge-type learned weight, then a sigmoid function is applied. In this model, different types of sub-questions may have different influence on confidences, even when the QA model produces similar features for them.

- The *Matching Model* estimates the confidence in a hypothesis according to how well each factor in

the scenario, plus the answers to forward questions asked about it, match against either the hypothesis or the answers to the backward questions asked from it. We estimate this degree of match using the term matchers described Section 6.2.

- The *Feature Addition Model* uses the same DAG as the Noisy-OR model, but confidence in the intermediate nodes is computed by adding the feature values for the questions that lead to it and then applying the logistic model to the resulting vector. An effect is that the confidence for a node does not increase monotonically with the number of parents. Instead, if features that are negatively associated with correctness are present in one edge, it can lower the confidence of the node below the confidence given by another edge.

- The *Causal Model* attempts to capture causal semantics by expressing the confidence for each candidate as the product over every clinical factor of the probability that either the diagnosis could explain the factor (as estimated from Watson/QA features), or the factor "leaked" - it is an unexplained observation or is not actually relevant.

In the closed-form inference systems described, there is no constraint that the answer confidences sum to one. We implement a final stage where features based on the raw confidence from the inference model are transformed into a proper probability distribution over the candidate answers.

8 Learning over Assertion Graphs

Inference methods like those described in the previous section depend on the strengths of edges created from the answers to subquestions. WatsonPaths uses supervised machine learning to learn these edge strengths from training data. There are two different kinds of training data that we can employ:

- *Scenario question training data* includes complete scenarios, questions about those scenarios, and answers to those questions (e.g., a detailed description of a patient, a question asking what is wrong with the patient, and the correct diagnosis).

- *Subquestion training data* includes simpler, atomic questions and answers to those questions (e.g., "What diseases cause joint pain?" and some answers to that question).

The WatsonPaths process that we have described up to this point assumes that we first train a subquestion answering model using subquestion training data and use the outputs of that model as confidences for inference. But we have found that this approach has limitations, due in part to issues with our existing subquestion training data (described in Section 10). This approach also suffers from the limitation that it only uses information internal to the subquestion answering system. Some inference methods have parameters that are not based on subquestions. For example, some approaches develop a model for the degree that two nodes match or the importance of a given node. A simple baseline for node importance is to give all nodes either equal weight or a weight based on a single simple statistic like IDF (inverse document frequency). A simple model for matching can take the confidence from a single term matcher thought to be generally effective. Graph-based features like these can be useful in combination with subquestion answering features for inference model learning.

Therefore, we have added a final step to the process that makes use of the scenario question training data. Using the assertion graphs that WatsonPaths has built for each scenario question, our goal is to learn a model that produces a probability distribution over answers with as much of the mass as possible concentrated on the correct answer. This learning is challenging because each assertion graph contains very different nodes and edges from the others, even different numbers of nodes and edges. Fortunately, the edges in these graphs do share a common set of features, such as question answering features, matching features, and node type features.

A complication is that Watson has a large quantity of question answering features, and many of them have similar purposes; e.g., many features independently assess whether the answer has the desired type (Murdock et al., 2012b). There is a subtle and indirect connection between the behavior of the subquestion answering system and the final answers to scenario question training data; this makes it very hard for a learning system using only the scenario question training data to find an effective model over so many features. Consequently, we employ a hybrid approach. We partition features into a small number of groups with similar purposes and we use subquestion training data to build a separate model for each group. The outputs of these models represent a consolidated set of question answering features (with one score for each group). We then use this consolidated set of question answering features as features for learning inference models (along with the additional graph-based features).

8.1 Direct Learning

We have explored several methods for transforming an assertion graph a into a function mapping the values of the weights to confidence in the correct hypothesis $\Phi_a : \mathbb{R}^n \rightarrow \mathbb{R}$. The methods in Section 7.2 provide fast, exact inference. These approaches permit express-

ing the confidence in the correct answer as a closed-form expression. Summing the log of the confidence in the correct hypothesis across the training set T, we construct a learning problem with log-likelihood in the correct final answer as our objective function. The result is a function that is non-convex, and in some cases (due to max) not differentiable in the parameters.

To limit overfitting and encourage a sparse, interpretable parameter weighting we use L1-regularization. The absolute value of all learned weights is subtracted from the objective function.

$$\vec{w^*} = \underset{\vec{w} \in \mathbb{R}^n}{\operatorname{argmax}} -\|\vec{w}\|_1 + \sum_{t \in T} \log(\Phi_t(\vec{w}))$$

To learn the parameters for the inference models we apply a "black-box" optimization method: greedy-stochastic local search. This is a method of direct search (Kolda et al., 2003) that considers a current point in $p \in \mathbb{R}^n$ and a neighborhood function mapping points to subsets of $\mathbb{R}^n$, $\mathcal{N} : \mathbb{R}^n \rightarrow \mathscr{P}(\mathbb{R}^n)$. Additionally the optimization procedure maintains p^*, the best known point. From the current point a new point p' is randomly selected from $\mathcal{N}(p)$. If the change improves the objective function, then it is kept; if the change worsens the objective function, then it is accepted with some probability ϵ. In this way, the learning explores the parameter space, tending to search in regions of high value while never becoming stuck in a local maximum.

We use a neighborhood function $\mathcal{N}$ related to compass search. A single parameter or a pair of parameters is selected to change by some δ. Additionally, due to the L1 regularization, the neighborhood permits setting any single parameter to zero, encouraging sparse solutions.

There is no straightforward stopping criterion for this search, so we limit by time. Empirically we found that optimization longer than two hours rarely improved the objective function substantially.

Not every Φ_t depends on every element of $\vec{w}$. Even in cases where a Φ_t depends on $\vec{w_i}$, many pieces of the function may not. To enable efficient recomputation, a preprocessor constructs for each weight $\vec{w_i}$ a DAG indicating which parts of functions will need to be recomputed, and in what order, if that weight is changed. Unchanged function parts return their cached value if used in the computation of a part that does change.

We also experimented with the Nelder-Mead simplex method (Nelder and Mead, 1965) and the multidirectional search method of Torczon (1989) but found weaker performance from these methods.

8.2 Ensemble Learning

We have multiple inference methods, each approaching the problem of combining the subquestion confidences from a different intuition and formalizing it in a different way. To combine all these different approaches we train an ensemble. This is a final, convex, confidence estimation over the multiple choice answers using the predictions of the inference models as features. The ensemble learning uses the same training set that the individual closed-form inference models use. To avoid giving excess weight to inference models that have overfit the training set, we use a common technique from stacking ensembles (Breiman, 1996). The training set is split into five folds, each leaving out 20% of the training data, as though for cross validation. Each inference model from Section 7.2 is trained on each fold. When the ensemble gathers an inference model's confidence as a feature for an instance, the inference model uses the learned parameters from the fold that excludes that instance. In this way, each inference model's performance is test-like, and the ensemble model does not overly trust overfit models.

The ensemble is a binary logistic regression per answer hypothesis using three features from each inference model. The features used are: the probability of the hypothesis, the logit of the probability, and the rank of the answer among the multiple choice answers. Using the logit of the probability ensures that selecting a single inference model is in the ensemble's hypothesis space, achieved by simply setting the weight for that model's logit feature to one and all other weights to zero.

Each closed-form inference model is also trained on the full training set. These versions are applied at test time to generate the features for the ensemble.

9 Collaborative Learning Application

Applying a question-answering tool like Watson to complex, scenario-driven problems was a challenge that we didn't solve until we observed how humans do this. As explained in the Introduction, an inspiration for our approach came from seeing medical students explain their reasoning about medical test preparation questions. Their process was one of identifying significant details, drawing inferences, and evaluating hypotheses. This way of attacking a problem is recognizable in the Watson-Paths execution flow. So when it came time to create an interactive application, it was the obvious conclusion that it should facilitate a certain way of reasoning about complex scenarios. We call this the Collaborative Learning Application.

9.1 A Learning Application

The Collaborative Learning Application creates a workflow in which the user and Watson work together to analyze a problem. We believe that this approach will result in better solutions than if the user or Watson were working alone. It is also thought this will create opportunities for the user and Watson to learn.

For the user, "to learn" is meant in the traditional sense. First, we propose that the application aids in teaching critical thinking through its advocacy of a certain reasoning process. Second, we propose that exploring Watson's analyses provides educational value by giving a unique and relevant index into the huge body of unstructured knowledge that was examined to produce these results.

For Watson, learning is primarily in the sense of machine learning. That is, learning involves deriving labeled data from usage data and using that data to improve our statistical models. What we find interesting about the WatsonPaths application are the many opportunities to gather such data through both implicit and explicit means. We call these "opportunistic annotations," because they are gathered in the course of using the system. There is a synergy between mimicking the way a human thinks about a problem and the way the machine analyzes a problem in developing ways to gather useful data.

If the Collaborative Learning Application can successfully produce educational value for the user or Watson, then we submit that the system (inclusive of the application and the user) learns and improves. That is, we can expect the system to provide better results over time. Of course, we know that humans are capable of this property and so along that dimension we wish to show that the application is associated with a faster rate of learning than, for instance, using a search engine over similar corpora. And along the dimension of the machine's improvement, we wish to show statistically significant results. As the ability to achieve this is a function of the amount and quality of usage data gathered, it will be interesting to explore what requirements this implies.

9.2 A Collaborative Application

The interactions that the application supports are best explained at the logical level. At this level we find the assertion graph and the WatsonPaths processes used to populate it.

Our initial state is an empty assertion graph, and our first operation is to describe the problem scenario through statements of fact and assertions of truth about those statements. WatsonPaths does this through the scenario analysis process in which the input is natural language text and the output is a number of asserted statements. Through the Collaborative Learning Application, the user may choose to accept the result of this process, alter it through judgment, or bypass it altogether and create their own asserted statements. Each of these interactions produces data that we hope is valuable for improving the system.

By accepting the result of the process, there is an implicit annotation that the result has positive value in the user's judgment.

Altering the process through judgment produces explicit annotations by the user. Note that it is a more general interaction than might be inferred from this context since it is done at the assertion graph level and so is applicable to all of the WatsonPaths processes that operate on this data structure. A judgment is in fact the user expressing an opinion about a statement for which the system has also expressed an opinion. Expressing a similar opinion is an example of positive feedback. Expressing a dissimilar opinion is an example of negative feedback. If we can associate the system's opinion with a particular WatsonPaths process, then we can use these judgments as feedback regarding that process. (Understanding the bias of this feedback mechanism is a concern for us.)

By creating their own asserted statements, the user is generating their opinion of the ground truth for the WatsonPaths process. Knowing the input the user was operating on to produce this ground truth allows us to derive labeled data for the process.

In this step one might infer that there is a gated process in which the user vets and augments the machine's results prior to moving on with the analysis. Our current focus on medical test preparation questions is amenable to this approach but for very complex inputs (such as hundreds of pages of a patient's medical record) practicalities may necessitate a different, automatic mode of operation. Such a mode might be to allow WatsonPaths to work through the entire scenario alone and then invite the human to judge or augment the results as they see fit. In fact we are exploring both approaches as each has desirable characteristics (primarily understandability in the case of the gated process and decision facilitation in the automatic case).

The next operation is to prioritize statements in the assertion graph for further inquiry. That is, which statements about the scenario have the most promise of producing relevant inferences? Again, there is a WatsonPaths process to do this, and the user may choose to accept the result of this process, alter it, or bypass it altogether. Similarly to the previous case, annotations and labeled data may be derived from these interactions.

Next in the WatsonPaths process is to apply a semantic template to the prioritized statements to generate inferences. This semantic template may include information like "diseases cause findings" from which we may derive a query to infer a disease from a finding or a finding from a disease. When using Watson's question answering function to do this, the form of this query is a natural language question. For example, imagine that the statement *resting tremor* (read as "the patient has resting tremor") has been prioritized. Applying the described semantic template then would produce the question "What disease causes resting tremor?" The answers to this ques-

tion are new inferred factors.

There are many opportunities for user feedback in this process, but perhaps the most interesting one is to induce a semantic template from user interactions. We envision doing this by allowing users to ask questions of statements themselves and then extracting information from them. For example, if the user asks of "resting tremor," "What causes this?" then we might extract a semantic template instance of "things cause resting tremor." The value of this template might be improved by knowing the type of instances, and so we may ask the user, "What type of things cause resting tremor?" to which we receive an answer of "disease" or "neurological disorder," among other things. And through typing (a functionality of WatsonPaths), we may suppose that "resting tremor" is a finding. This may lead us to ask of the user, "Do diseases cause findings?" or "Do neurological disorders cause findings?" The user's response may further refine the semantic template. How far to refine the semantic template and whether it should be done with respect to a prior ontology are questions open to experimentation. An added personalization aspect is that the system could learn user-specific semantic templates, allowing each user to employ their own methodology for problem solving.

Executing the queries produced in the previous operation results in new asserted statements. As mentioned before, Watson's question answering function can be used to answer these intermediate queries. Here too, user input can help improve Watson's results. The user can accept Watson's results, alter them through judgment, or produce their own. For example, the user can suggest alternative names and phrasings for entities and relations in the question for use in query expansion. And as before, these interactions produce data that can be used to improve the system. In this case, these results fit nicely into AdaptWatson (Ferrucci and Brown, 2012), a methodology for improving question answering performance based on example question-answer pairs, data science, and machine learning.

An advantage of using Watson's question answering ability for inference is that results are supported by evidential passages. These passages are typically a few sentences extracted from a document that Watson used during feature generation for a candidate answer to a question and so often entail the high confidence answers. As such, exposing this evidence to the user can be very beneficial as an explanatory tool and so is something we emphasize in the Collaborative Learning Application. Allowing the user to judge this evidence can provide data that can be used to improve the search components of the question answering process as well as to train a *justifying passage model*—a model that classifies passages as justifying an answer to a question or not.

Evaluating the expanded assertion graph with respect to determining belief in the statements is the role of the belief engine. Here, judgments by the user about the significance or irrelevance of statements to the overall case can aid in how that evaluation is done.

A final step in the overall WatsonPaths process is hypothesis identification. For medical test preparation questions, hypotheses are provided, but for the Collaborative Learning Application they are not. For a particular scenario, the list of hypotheses will change as the analysis progresses. What might have begun with general disorders may end with specific diseases. And what might have begun with diagnosis might transition into treatment. By observing the patterns of usage, we hope to automate this higher level of reasoning imposed on the system.

The final assertion graph as confirmed by the user can be stored in Watson's internal knowledge base. The knowledge base can store assertions along with confidence and provenance information (with appropriate merging and conflict resolution strategies), and grows as more users interact with the system. This growing pool of background knowledge further improves Watson's question answering capability.

The annotations we expect to derive from the application are quite specific to the WatsonPaths processes. This is a benefit of choosing processes that mimic human reasoning and so have some degree of familiarity and intuitive performance for the user. This also leads to opportunistic annotations or specific questions posed to the user that might generate useful data. For example, if the user were to dismiss a evidential passage that Watson had scored highly, an opportunity arises to ask why. Perhaps a particular question answering component had generated a high score for the passage. Determining if it should not have can be helpful to improving that component. Other questions may provide axiomatic information (such as existence of a paraphrase) that may be useful in the specific context.

9.3 A Cognitive Computing Application

The values we follow in the development of the Collaborative Learning Application are described in shorthand as cognitive computing. That notion encompasses three primary characteristics: facilitate human reasoning, communicate in a natural way, and learn and improve (Kelly and Hamm, 2013). How we are addressing the first and last of these characteristics should be clear at this point, but the second deserves further mention.

Being able to express the assertion graph to the user in an intuitive way is a challenge we are working on, but which has generated positive feedback. Drawing on the value of concept maps (Daley and Torre, 2010), our graph-based visualization provides an approachable pre-

sentation that users understand quickly.

For example, Figure 6 shows the application during analysis of a scenario in which a patient's symptoms lead to a diagnosis of Parkinson's disease which in turn leads to the answer *Substantia nigra*. (The ultimate question being answered is, "What part of the patient's nervous system is most likely affected?") The graph representation allows the user to visually navigate the result from asserted true statements (nodes arranged at the top of the screen), through inferences (white nodes in the middle of the screen), and to hypotheses (nodes arranged at the bottom of the screen). Decorations on the graph like line width and opacity give the user a sense of how belief is flowing while significance indicators (dashed lines below factors) show which factors the belief engine favored in its choice of an answer. Something to note is that the complete result has 333 nodes and 444 edges and so editing of the graph is needed.

9.4 Current Status

The Collaborative Learning Application is a work in progress and we are refining and exploring in the context of our collaboration with the Cleveland Clinic Lerner College of Medicine. At that medical school, critical thinking is taught through a problem-based learning curriculum in which students work through medical scenarios as a group. The way in which the students do this has similarities to the WatsonPaths process, and so we hope that the application we are building on that functionality will be able to facilitate their thinking while providing educational value—something we hope to measure in an upcoming pilot.

10 Evaluation

As illustrated in the previous section, an interactive, collaborative clinical decision support tool can benefit from the same components and technologies needed for an automatic scenario-based medical question answering system. Thus developing and testing the automatic system in the standard way on sets of medical questions has the benefits of (1) driving the development of the core technology, (2) providing an evaluation of the automatic system, and (3) improving the components of the interactive system; such an evaluation is the subject of this section. Note that an evaluation of the interactive system itself is a separate exercise and will be reported in a future paper.

10.1 Data Sets

For the automatic evaluation of WatsonPaths, we used a set of medical test preparation questions from Exam Master and McGraw-Hill, which are analogous to the examples we have used throughout this paper. These questions consist of a paragraph-sized natural language scenario description of a patient case, optionally accompanied by a semi-structured tabular structure. The paragraph description typically ends with a punchline question and a set of multiple choice answers (average 5.2 answer choices per question). We excluded from consideration questions that require image analysis or whose answers are not text segments.

The punchline questions may simply be seeking the most likely disease that caused the patient's symptoms (e.g., "What is the most likely diagnosis in this patient?"), in which case the question is classified as a diagnosis question. The diagnosis question set reported in this evaluation was identified by independent annotators. Non-diagnosis punchline questions may include appropriate treatments, the organism causing the disease, and so on (e.g., "What is the most appropriate treatment?" and "Which organism is the most likely cause of his meningitis?" respectively).

We split our data set of 2190 questions into a training set of 1000 questions, a development set of 690 questions, and a blind test set of 500 questions. The development set was used to iteratively drive the development of the scenario analysis, relation generation, and belief engine components, and for parameter tuning. The training set was used to build models by the learning component described in Section 8.

As noted earlier, our learning process requires subquestion training data to consolidate groups of question answering features into smaller, more manageable sets of features. We do not have robust and comprehensive ground truth for a sufficiently large set of our automatically generated subquestions. Instead, we use a preexisting set of simple factoid medical questions as subquestion training data: the Doctor's Dilemma (DD) question set (American College of Physicians, 2014). DD is an established benchmark used to assess performance in factoid medical question answering. We use 1039 DD questions (with a known answer key) as our subquestion training data. Although the Doctor's Dilemma questions do have some basic similarity to the subquestions we ask in assertion graphs, there are some important differences:

- In an assertion graph subquestion, there is usually one known entity and one relation that is being asked about. For DD, the question may constrain the answer by multiple entities and relations.
- An assertion graph subquestion like "What causes hypertension?" has many correct answers, whereas DD questions have a single best answer.
- There may be a mismatch between how confidence for DD is trained and how subquestion confidence is used in an inference method. The DD confidence model is trained to maximize log-likelihood on a

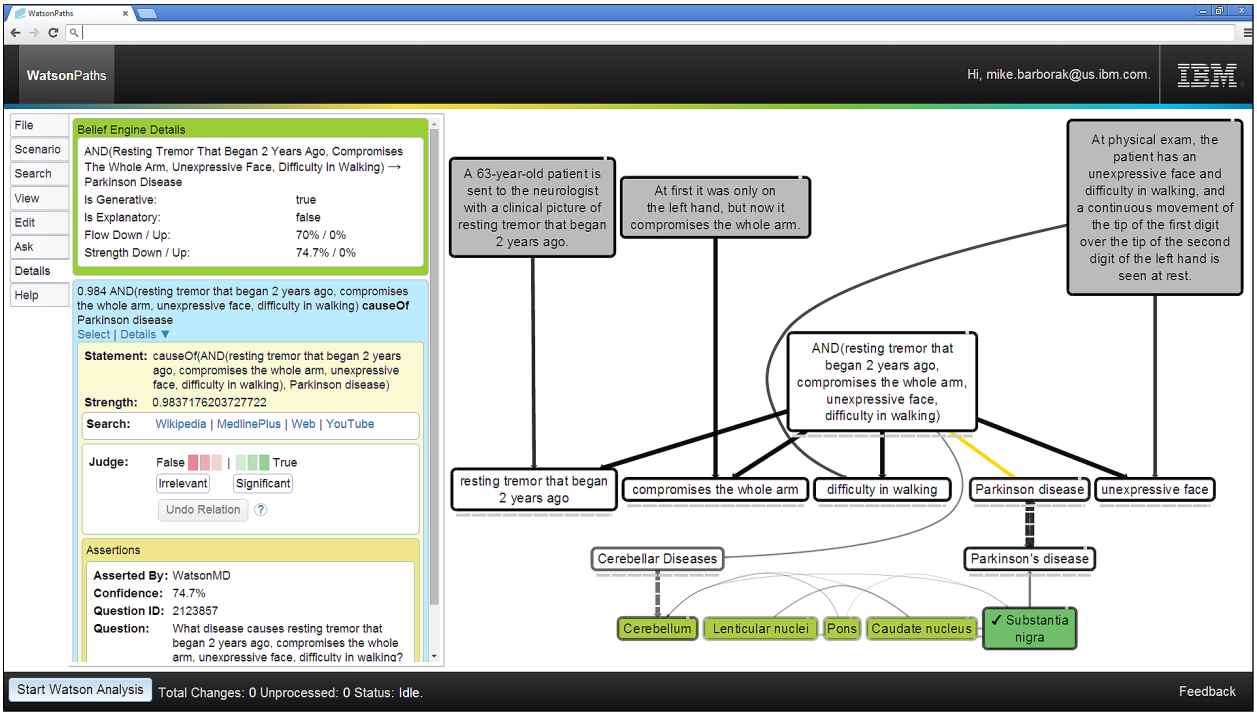

Figure 6: WatsonPaths User Interface

correct/incorrect binary classification task. In contrast, many probabilistic inference methods use confidence as something like strength of indication or relevance.

For all these reasons, DD data is poorly suited to training a complete model for judging edge-strength for subquestion edges in WatsonPaths. But we have found that DD data is useful as subquestion training data[1] in the hybrid learning approach described in Section 8; we use 1039 DD questions for consolidating question answering features and then use the smaller, consolidated set of features as inputs to the inference models that are trained on the 1000 medical test preparation questions.

10.2 Experimental Setup

For comparison purposes, we used our Watson question answering system adapted for the medical domain (Ferrucci et al., 2013) as a baseline system. This system takes the entire scenario as input and evaluates each multiple choice answer based on its likelihood of being the correct answer to the punchline question. This one-shot approach to answering medical scenario questions contrasts with the WatsonPaths approach of decomposing the scenario, asking questions of atomic factors, and performing probabilistic inference over the resulting graphical model.

We tuned various parameters in the WatsonPaths system on the development set to balance speed and performance. The system performs one iteration each of forward and backward relation generation. The minimum confidence threshold for expanding a node is 0.25, and the maximum number of nodes expanded per iteration is 40. In the relation generation component, the Watson medical question answering system returns all answers with a confidence of above 0.01.

We evaluate system performance both on the full test set as well as on the diagnosis subset only. The reason for evaluating the diagnosis subset separately is because, in the vast majority of these questions, either the punchline question seeks the diagnosis or depends on a correct diagnosis along the way. We use the full 1000 questions in the training set to learn the models for both the baseline system and the WatsonPaths system. As noted earlier, Doctor's Dilemma training data is used to consolidate question answering features in the WatsonPaths system. We did not use Doctor's Dilemma training data for any purpose in the baseline system.

[1]We are also investigating the use of actual subquestions generated by WatsonPaths as training data. Building a comprehensive answer key for such questions is very time consuming, and an incomplete answer key can be less effective. Although this approach has not yet succeeded, it may still succeed if we invest much more in building a bigger, better answer key for actual WatsonPaths subquestions.

		Full	Diagnosis
Accuracy	Baseline	42.0%	53.8%
	WatsonPaths	**48.0%**	**64.1%**
Confidence Weighted Score	Baseline	59.8%	75.3%
	WatsonPaths	**67.5%**	81.8%

Table 1: WatsonPaths Performance Results

10.3 Results and Discussion

Table 1 shows the results of our evaluation on a set of 500 blind questions of which a subset of 156 questions were identified as diagnosis questions by annotators.

We report results using two metrics. **Accuracy** simply measures the percentage of questions for which a system ranks the correct answer in top position. **Confidence Weighted Score** is a metric that takes into account both the accuracy of the system and its confidence in producing the top answer (Voorhees, 2003). We sort all <question, top answer> pairs in an evaluation set in decreasing order of the system's confidence in the top answer and compute the confidence weighted score as

$$\text{CWS} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{number correct in first } i \text{ ranks}}{i}$$

where n is the number of questions in the evaluation set. This metric rewards systems for more accurately assigning high confidences to correct answers, an important consideration for real-world question answering and medical diagnosis systems.

Our results show statistically significant improvements at $p<0.05$ (results in bold in Table 1) on the full blind set of 500 questions for both metrics. For the diagnosis subset, the accuracy improvement is statistically significant but the confidence weighted score improvement is not, even with a 6+% score increase. This is likely due to the small diagnosis subset, which contains only 156 questions.

11 Related Work

Clinical decision support systems (CDSS) have had a long history of development starting from the early days of artificial intelligence. These systems use a variety of knowledge representations, reasoning processes, system architectures, scope of medical domain, and types of decision (Musen et al., 2014). Although several studies have reported on the success of CDSS implementations in improving clinical outcomes (Kawamoto et al., 2005; Roshanov et al., 2013), widespread adoption and routine use is still lacking (Osheroff et al., 2007).

The pioneering Leeds abdominal pain system (De Dombal et al., 1972) used structured knowledge in the form of conditional probabilities for diseases and their symptoms. Its success at using Bayesian reasoning was comparable to experienced clinicians at the Leeds hospital where it was developed. But it did not adapt successfully to other hospitals or regions, indicating the brittleness of some systems when they are separated from their original developers. A recent systemic review of 162 CDSS implementations shows that success at clinical trials is significantly associated with systems that were evaluated by their own developers (Roshanov et al., 2013). MYCIN (Shortliffe, 1976) was another early system which used structured representation in the form of production rules. Its scope was limited to the treatment of infectious diseases and, as with other systems with structured knowledge bases, required expert humans to develop and maintain these production rules. This manual process can prove to be infeasible in many medical specialties where active research produces new diagnosis and treatment guidelines and phases out older ones. Many CDSS implementations mitigate this limitation by focusing their manual decision logic development effort on clinical guidelines for specific diseases or treatments, e.g., hypertension management (Goldstein et al., 2001). But such systems lack the ability to handle patient comorbidities and concurrent treatment plans (Sittig et al., 2008). Another notable system that used structured knowledge was Internist-1. The knowledge base contained disease-to-finding mappings represented as conditional probabilities (of disease given finding and of finding given disease) mapped to a 1–5 scale. Despite initial success as a diagnostic tool, its design as an expert consultant was not considered to meet the information needs of most physicians. Eventually, its underlying knowledge base helped its evolution into an electronic reference that can provide physicians with customized information (Miller et al., 1986). A similar system, DXplain (Barnett et al., 1987) continues to be commercially successful and extensively used. Rather than focus on a definitive diagnosis, it provides the physician with a list of differential diagnoses along with descriptive information and bibliographic references.

Other systems in commercial use have adopted the unstructured medical text reference approach directly, using search technology to provide decision support. Isabel provides diagnostic support using natural language processing of medical textbooks and journals. Other commercial systems like UpToDate and ClinicalKey forgo the diagnostic support and provide a search capability to their medical textbooks and other unstructured references. Although search over unstructured content makes it easier to incorporate new knowledge, it shifts the reasoning load from the system back to the physician.

In comparison to the above systems, WatsonPaths uses a hybrid approach. It uses question-answering technology over unstructured medical content to obtain answers to specific subquestions generated by WatsonPaths. For

this task, it builds on the search functionality by extracting answer entities from the search results and seeking supporting evidence for them in order to estimate answer confidences. These answers are then treated as inferences by WatsonPaths over which it can perform probabilistic reasoning without requiring a probabilistic knowledge base.

Another major area of difference between CDSS implementations is the extent of their integration to the health information system and workflow used by the physicians. Studies have shown that CDSS are most effective when they are integrated within the workflow (Kawamoto et al., 2005; Roshanov et al., 2013). Many of the guideline-based CDSS implementations are integrated with the health information system and workflow, having access to the data being entered and providing timely decision support in the form of alerts. But this integration is limited to the structured data contained in a patient's electronic medical record. When a CDSS requires information like findings, assessments, or plans in clinical notes written by a healthcare provider, existing systems are unable to extract them. As a result, search-based CDSS remain a separate consultative tool. The scenario analysis capability of WatsonPaths provides the means to analyze these unstructured clinical notes and serves as a means for integration into the health information system.

A major point of differentiation between the CDSS implementations described above and the design of WatsonPaths is its ability to serve as a collaborative problem solving tool as described in Section 9. When teamed with a student, the role of WatsonPaths approaches that of intelligent tutoring systems (Woolf, 2009). Key differences exist, however, in the representation of domain knowledge and student knowledge. Most tutoring systems have a structured representation of the domain knowledge, carrying with it the same knowledge update and maintenance issues faced by CDSS implementations. WatsonPaths lacks a student model (or in general a model of the collaborator), which is a key capability of intelligent tutoring systems. As a result, it cannot guide or customize the tutoring according to student needs, relying instead on an instructor's choice of the problem scenario to be used.

12 Conclusions and Further Work

WatsonPaths is a system for scenario-based question answering that has a graphical model at its core. It includes a collaborative decision support tool that allows users to understand and contribute to the reasoning process. We have developed WatsonPaths on a set of multiple choice questions from the medical domain. On this test set, WatsonPaths shows a significant improvement over Watson. Although the test preparation question set has been important for the early development of the system, we have designed WatsonPaths to function well beyond it. In future work, we plan to extend WatsonPaths in several ways.

The present set of questions are all multiple choice questions. This means that hypotheses have already been identified, and it is also known that exactly one of the hypotheses is the correct answer. Although they have made the early development of scenario-based question answering more straightforward, the overall WatsonPaths architecture does not rely on these constraints. For instance, we can easily remove the confidence re-estimation phase for the closed-form inference systems and the "exactly one" constraint from the belief engine. Also, it will be straightforward to add a simple hypothesis identification step to the main loop. One way to do this is to find nodes whose type corresponds to the type being asked about in the punchline question. We already find such correspondances in the base Watson system (Chu-Carroll et al., 2012). In the collaborative application, we are exploring ways of having the user help identify hypotheses.

We also plan to extend WatsonPaths beyond the medical domain. For medical applications, it might have been easier to design Watson with certain medical aspects hardcoded into the flow of execution. Instead we designed the overall flow as well as each component to be general across domains. Note that the Emerald could be replaced by a structure from a different domain, and the basic semantics we have explored: matching, indicative and causal, have no requirement that the graph structure come from medicine. Even the causal aspect of the belief engine could apply to any domain that involves diagnostic inference (e.g., automotive repair). Most importantly, the way that subquestions are answered is completely general. By asking the right subquestions and using the right corpus, we can apply WatsonPaths to any scenario-based question answering problem. We hope to develop a toolbox of expansion strategies, relation generators, and inference mechanisms that can be reused as we apply WatsonPaths to new domains.

The most important area for further work is on the collaborative user application. In the early development of the system, it was necessary to focus on automatic performance (as presented in Section 10) to create a viable scenario-based question answering system. As this performance improves, we are focusing more on how WatsonPaths can interact better with users. We plan to develop and more rigorously evaluate how WatsonPaths learns from users and how users learn from WatsonPaths.

In a fully automatic system, the user receives an answer using little or no time or cognitive effort. In a collaborative system, the user spends some time and effort, and potentially gets a better answer. We suspect that, in

many applications of scenario-based question answering, this will be an attractive tradeoff for the user, because of the complexity of the scenario and the importance of the answer. Our objective is to minimize the time and effort required of users and maximize the benefit they receive. The combination of the user and WatsonPaths should be able to handle more difficult problems more quickly than either could alone.

References

American College of Physicians. 2014. Doctor's Dilemma competition. `http://www.acponline.org/residents_fellows/competitions/doctors_dilemma/`.

G. Octo Barnett, James J. Cimino, Jon A. Hupp, and Edward P. Hoffer. 1987. DXplain: An evolving diagnostic decision-support system. *JAMA*, 258(1):67–74.

Judith L. Bowen. 2006. Educational strategies to promote clinical diagnostic reasoning. *New England Journal of Medicine*, 355(21):2217–2225.

E. Boyd, Kenneth W. Kennedy, Richard A. Tapia, and Virginia Joanne Torczon. 1989. Multi-directional search: A direct search algorithm for parallel machines. Technical report, Rice University.

Leo Breiman. 1996. Stacked regressions. *Machine Learning*, 24(1):49–64.

Rowland W. Chang, Georges Bordage, and Karen J. Connell. 1998. Cognition, confidence, and clinical skills: The importance of early problem representation during case presentations. *Academic Medicine*, 73(10):S109–111.

Patricia W. Cheng. 1997. From covariation to causation: A causal power theory. *Psychological Review*, 104(2):367.

Jennifer Chu-Carroll, James Fan, Branimir K. Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. 2012. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, 56(3/4):6:1–6:12.

Barbara J. Daley and Dario M. Torre. 2010. Concept maps in medical education: An analytical literature review. *Medical Education*, 44(5):440–448.

F. T. De Dombal, D. J. Leaper, John R. Staniland, A. P. McCann, and Jane C. Horrocks. 1972. Computer-aided diagnosis of acute abdominal pain. *British Medical Journal*, 2(5804):9.

David Ferrucci and Eric Brown. 2012. AdaptWatson: A methodology for developing and adapting Watson technology. Technical Report RC25244, IBM Research Division.

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. 2010. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31:59–79.

David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. 2013. Watson: Beyond Jeopardy! *Artificial Intelligence*, 199–200:93–105.

David Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3/4):1:1–1:15.

M. K. Goldstein, B. B. Hoffman, R. W. Coleman, S. W. Tu, R. D. Shankar, M. O'Connor, S. Martins, A. Advani, and M. A. Musen. 2001. Patient safety in guideline-based decision support for hypertension management: ATHENA DSS. In *Proceedings of the AMIA Symposium*, page 214. American Medical Informatics Association.

David C. Gondek, Adam Lally, Aditya Kalyanpur, J. William Murdock, Pablo A. Duboue, Lei Zhang, Yue Pan, Zhao Ming Qiu, and Chris Welty. 2012. A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research and Development*, 56(3/4):14:1–14:12.

Kensaku Kawamoto, Caitlin A. Houlihan, E. Andrew Balas, and David F. Lobach. 2005. Improving clinical practice using clinical decision support systems: A systematic review of trials to identify features critical to success. *British Medical Journal*, 330:765–72.

John E. Kelly and Steve Hamm. 2013. *Smart machines: IBM's Watson and the era of cognitive computing*. Columbia University Press, New York.

Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. 2003. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482.

John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*, pages 282–289.

Adam Lally, John M. Prager, Michael C. McCord, Branimir K. Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How Watson reads a clue. *IBM Journal of Research and Development*, 56(3/4):2:1–2:14.

Michael C. McCord, J. William Murdock, and Branimir K. Boguraev. 2012. Deep parsing in Watson. *IBM Journal of Research and Development*, 56(3/4):3:1–3:15.

Randolph A. Miller, Melissa A. McNeil, Sue M. Challinor, Fred E. Masarie Jr, and Jack D. Myers. 1986. The Internist-1/Quick Medical Reference project – status report. *Western Journal of Medicine*, 145(6):816.

J. William Murdock, James Fan, Adam Lally, Hideki Shima, and Branimir K. Boguraev. 2012a. Textual evidence gathering and analysis. *IBM Journal of Research and Development*, 56(3/4):8:1–8:14.

J. William Murdock, Aditya Kalyanpur, Chris Welty, James Fan, David Ferrucci, David C. Gondek, Lei Zhang, and Hiroshi Kanayama. 2012b. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56(3/4):7:1–7:13.

Mark A. Musen, Blackford Middleton, and Robert A. Greenes. 2014. Clinical decision-support systems. In *Biomedical Informatics*, pages 643–674. Springer.

National Library of Medicine. 2009. UMLS reference manual. Bethesda, MD: National Library of Medicine (US). `http://www.ncbi.nlm.nih.gov/books/NBK9676/`.

J. A. Nelder and R. Mead. 1965. A simplex method for function minimization. *Computer Journal*, 7:308–313.

Jerome A. Osheroff, Jonathan M. Teich, Blackford Middleton, Elaine B. Steen, Adam Wright, and Don E. Detmer. 2007. A roadmap for national action on clinical decision support. *Journal of the American Medical Informatics Association*, 14(2):141–145.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, San Francisco.

J. M. Prager, J. Chu-Carroll, and K. Czuba. 2004. Question answering using constraint satisfaction: QA-by-dossier-with-constraints. In *Proceedings of the 42nd Association for Computational Linguistics*, pages 575–582, Barcelona.

Pavel S. Roshanov, Natasha Fernandes, Jeff M. Wilczynski, Brian J. Hemens, John J. You, Steven M. Handler, Robby Nieuwlaat, Nathan M. Souza, Joseph Beyene, Harriette G. C. Van Spall, Amit X. Garg, and R. Brian Haynes. 2013. Features of effective computerised clinical decision support systems: Meta-regression of 162 randomised trials. *British Medical Journal*, 346(f657).

Edward H. Shortliffe. 1976. *MYCIN: Computer-based medical consultations*. Elsevier, New York.

Dean F. Sittig, Adam Wright, Jerome A. Osheroff, Blackford Middleton, Jonathan M. Teich, Joan S. Ash, Emily Campbell, and David W. Bates. 2008. Grand challenges in clinical decision support. *Journal of Biomedical Informatics*, 41(2):387–392.

Ellen M. Voorhees. 2003. Overview of TREC 2002. In *Proceedings of the Text REtrieval Conference*.

Beverly Park Woolf. 2009. *Building intelligent interactive tutors*. Morgan Kaufmann, Burlington, MA.

Alan L. Yuille and Hongjing Lu. 2007. The noisy-logical distribution and its application to causal inference. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc.