Structured data and inference in DeepQA

A. Kalyanpur
B. K. Boguraev
S. Patwardhan
J. W. Murdock
A. Lally
C. Welty
J. M. Prager
B. Coppola
A. Fokoue-Nkoutche
L. Zhang
Y. Pan
Z. M. Qiu

Although the majority of evidence analysis in DeepQA is focused on unstructured information (e.g., natural-language documents), several components in the DeepQA system use structured data (e.g., databases, knowledge bases, and ontologies) to generate potential candidate answers or find additional evidence. Structured data analytics are a natural complement to unstructured methods in that they typically cover a narrower range of questions but are more precise within that range. Moreover, structured data that has formal semantics is amenable to logical reasoning techniques that can be used to provide implicit evidence. The DeepQA system does not contain a single monolithic structured data module; instead, it allows for different components to use and integrate structured and semistructured data, with varying degrees of expressivity and formal specificity. This paper is a survey of DeepQA components that use structured data. Areas in which evidence from structured sources has the most impact include typing of answers, application of geospatial and temporal constraints, and the use of formally encoded a priori knowledge of commonly appearing entity types such as countries and U.S. presidents. We present details of appropriate components and demonstrate their end-to-end impact on the IBM Watson™ system.

Introduction

A misconception about IBM Watson* and the DeepQA technology is that it takes an English question and "looks up the answer" in a structured knowledge resource or database. Although this is simply not the case, this misconception can be attributed to the fact that in the early days of automated question answering (QA), the systems were conceived as natural-language interfaces to databases [1, 2]. These early QA systems worked by translating the natural-language question into a formal structured query and issuing it against a precompiled database of knowledge in order to arrive at the answer.

There are two main downsides with this traditional approach to QA. First, it requires language to be precisely and completely translated into a formal representation. Second, it requires the underlying structured data and schema to be encoded and populated (either semi-automatically or manually) in a form suitable to answering questions, which can be a complex and cumbersome process. These two factors make the approach narrow in scope and extremely brittle. The systems are often unable to interpret the question in a way that matches their predefined vocabulary and schema. Moreover, such an approach does not work for open-domain QA and particularly for the Jeopardy!** problem, where questions are expressed using a wide variety of linguistic expressions and span a broad range of topics.

The design philosophy of DeepQA [3] breaks from this traditional view without completely discarding it; that is, although Watson can make use of formal structured knowledge, it does not completely rely on it, and Watson employs many retrieval and scoring methods based on unstructured and structured knowledge to find and evaluate candidate answers.

Many of these methods do not rely on any internal representation of the natural language at all but rather employ algorithms that use unstructured sources and compare natural-language statements to each other on the basis of properties of the statements themselves [4]. Such methods tend to provide broad coverage; they are capable of producing candidate answers and evidence scores for many different questions, but they are not always precise. Methods that use formal structured knowledge typically cover a

Digital Object Identifier: 10.1147/JRD.2012.2188737

©**Copyright** 2012 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/12/$5.00 © 2012 IBM

narrower range of questions but are more precise within that range. As a result, they are a natural complement to unstructured methods.

The critical factor in exploiting formally represented knowledge is the ability to translate from natural language into a formal representation. The natural-language processing (NLP) community has struggled with this problem since its inception, but, within the past 20 years, less exacting approaches that do part of this job have emerged, e.g., named entity recognition and relation extraction. These technologies extract pieces of a formal representation, such as a limited set of types of entities and a few types of relations expressed between them. In addition, and importantly, they do these tasks with some understood notion of error, such as precision and recall.

DeepQA's use of formally represented knowledge is similar in direction. For the most part, Watson does not get a complete meaning representation of a question and then try to answer it; rather, Watson finds pieces of knowledge from questions and tries to exploit what sources it has as evidence for answers. Importantly, it does this in the presence of precision and recall errors that normally confound representation and reasoning systems.

The components that rely on structured data work in parallel with DeepQA's other less structured scoring mechanisms [4]. DeepQA answer ranking [5] treats both types of results as machine learning features that are used to rank candidate answers and determine the confidence of ranked answers.

Note that there is no single monolithic structured data module in the DeepQA system. Instead, the current design of DeepQA allows for different components to use and integrate structured and semistructured data with varying degrees of expressivity and formal specificity. Components that need access to background resource description framework (RDF) stores such as DBpedia [6] and Freebase [7] use a shared query application programming interface (API). However, in many cases, components have their own internal representations for storing and reasoning over structured data. This paper is a survey of the main DeepQA components that benefit from using structured data and inference.

The remainder of this paper is organized as follows: In the next section, we motivate the need for structured data analytics in DeepQA. The subsequent section describes Watson's structured data resources and the manner in which they were aggregated or built. Then, we describe specific components in Watson that use structured data and inference and evaluate their impact on the Jeopardy! task. Finally, we discuss related work and ideas for future work.

Motivation

One of the main uses of structured data is to provide evidence in support of estimating the confidence in a candidate answer. To illustrate this point, consider the following question that expresses several constraints that the answer must satisfy:

> THE HOLE TRUTH (1200): Asian location where a notoriously horrible event took place on the night of June 20, 1756. (Answer: "Black Hole of Calcutta")

The correct answer must be a location, be in Asia, and be associated with an event that happened there on June 20, 1756. Evidence for any of these constraints *might* be found in text; however, it is very unlikely that the location would be explicitly expressed as being in Asia—more likely India or Calcutta. Even when a constraint is explicit in text, using formal knowledge to evaluate constraints can provide additional evidence for or against various candidate answers. Temporal and/or spatial constraints such as these are common in questions, and a significant proportion of those are covered by relationships such as containment or relative positioning among geospatial entities (near, east, west, etc.). Explicit geospatial databases are prevalent on the web, and temporal associations can be extracted with varying degrees of precision from background sources. It was clear that exploiting this wealth of structured data to help with these aspects of evaluating evidence would be beneficial. For example, we want to be able to use structured sources to tell us that the "Black Hole of Calcutta" is an Asian location.

Another important benefit of structured data evidence is that by virtue of its well-defined semantics, it is particularly useful for explanatory purposes as well. For example, semantic evidence related to time and space information is particularly helpful in understanding why particular answers are compatible or incompatible with certain aspects of the question.

Structured data stores used in Watson

In this section, we describe the sources of structured knowledge used in DeepQA, which broadly fall into four categories as follows:

1. Large online "off-the-shelf" databases of relations between known entities (such as movie databases) and type labels for known entities (such as Wikipedia** categories).
2. Large collections of automatically extracted data from unstructured sources that target specific kinds of knowledge (such as temporal associations).
3. A small amount of handcrafted additions to off-the-shelf sources to account for recognized differences between the task domain (e.g., Jeopardy!) and the source.
4. A small amount of handcrafted formal knowledge targeting the most common question and answer types.

The online off-the-shelf databases were available in RDF and were stored in a Sesame RDF repository [8], whereas for

the other three categories, the extracted data was stored in custom data structures that were designed and optimized to work with the corresponding components that used them.

Off-the-shelf knowledge resources

Contrary to the approach taken by, e.g., CYC** [9] and, more recently, WolframAlpha** [10], we committed early in the project *not* to curate large databases or common-sense axioms for answering questions but to exploit what was available on the Web. One reason for this decision was that we wanted to build reusable and adaptable QA technology, not a special-purpose Jeopardy! game-playing system. We also wanted some quantitative understanding of whether the wealth of data being brought online in the Semantic Web had value for QA applications.

Given our commitment to Wikipedia for candidate answer generation [11], it was clear that DBpedia [6] would be a natural source of structured information, because it is created from Wikipedia information boxes (i.e., infoboxes), and each Wikipedia article is an entity in DBpedia. We also extracted a small portion of the Freebase [7] knowledge base targeting spatial information, in particular, containment and border relations between geo-entities, since Freebase provides higher precision and coverage for these relations than DBpedia.

The DBpedia knowledge base contains relational information found in the infoboxes of Wikipedia pages. A one-to-one correspondence between all Wikipedia pages and DBpedia entries maps the two resource names (or URIs). The mapping between the two resources links unstructured and structured data (information from the structured domain provides a context for interpreting unstructured text, and vice versa), which we use for NLP tasks such as Entity Disambiguation and Matching (EDM) [12] and relation detection [13].

DBpedia also contains temporal and geospatial information, such as birth and death dates of people, durations of major events, and latitude/longitude for geo-entities, used by the temporal and geospatial scoring components in DeepQA. Because DBpedia is scraped from infoboxes, there are no controlled schema or data formats for the values listed there; hence, for example, dates are not in one standard format: The date "January 23, 1950" is variously represented as strings such as "23rd Jan 1950", "01/23/1950", and "23-01-1950". Consequently, we ran a temporal expression recognizer, broadly compliant with TimeML's TIMEX3 guidelines [14], converting temporal expression values to a standardized notation. Similarly, for spatial relations, we normalized variations in expressing latitude/longitude coordinate values by applying a regular-expression pattern recognizer.

Additionally, DBpedia has type assertions for many instance objects. The types are assigned from a collection of ontologies, including YAGO (Yet Another Great Ontology) [15], a large taxonomy of more than 100,000 types with mappings to WordNet** [16] synsets. Every YAGO type corresponding to a WordNet concept has the associated nine-digit WordNet sense identifier appended to its name/ID. Thus, the YAGO type "Plant100017222" links to the WordNet concept plant (living organism), whereas the type "Plant103956922" corresponds to the concept of an industrial plant or factory. YAGO types are arranged in a hierarchy, and DBpedia instances are often assigned several low-level types corresponding to Wikipedia categories (e.g., "CompaniesEstablishedIn1896"). For these, navigation up the YAGO type tree leads to more general and normalized (via sense encoding) YAGO WordNet concepts.

These design points of DBpedia and YAGO enable us to obtain precise type information for many candidate answers across many different possible types while including many linguistic variations (i.e., synonyms) for expressing them. This type information is exploited by the YAGO type coercion (TyCor) component, described later.

Automatically extracting structured data

Some DeepQA answer-scoring components use structured knowledge that is automatically mined from the Web. For example, one temporal answer scorer determines whether a candidate answer is "temporally compatible" with dates in the question. This scorer consults a knowledge base of (entity, date, and count) triples, where "count" is the number of entity–date co-occurrence pairs extracted from Wikipedia. The co-occurrence extractions are done in two ways: i) document-level, looking at Wikipedia articles where the title matches the entity and the article contains the date, and ii) sentence-level, looking for sentences across the entire Wikipedia corpus that contain the hyperlinked entity reference and the corresponding date.

Other examples of automatically extracted Web knowledge used in the DeepQA system include PRISMATIC [17] and some of the TyCor resources [12] (e.g., type information extracted from Wikipedia Lists).

Manually extending off-the-shelf sources

Certain DeepQA components rely on custom extensions of existing knowledge. For example, the YAGO ontology does not represent mutual exclusivity (disjointness) between types; e.g., there are no instances belonging to both *Country* and *Person*. Type disjointness is useful for QA, to rule out candidate answers whose types are incompatible with the question lexical answer type (LAT) (see [12] for anti-TyCor).

Given the size of the YAGO ontology (>100,000 types), manually asserting such relations between all applicable type pairs is infeasible. Instead, we only specify disjointness between prominent top-level types of the YAGO hierarchy and use a logical reasoning mechanism to propagate the disjointness to lower subclasses. For example, it follows

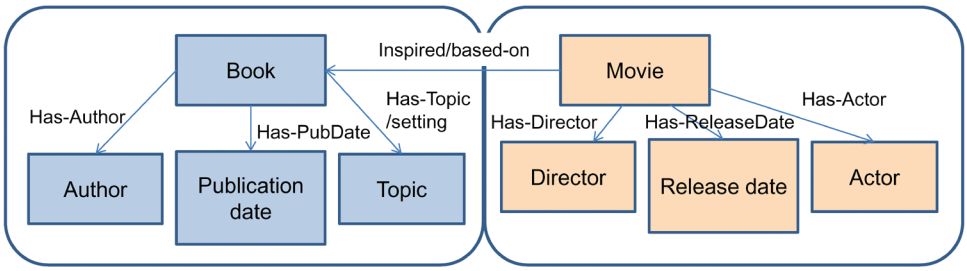

Figure 1

Book and Movie interrelated frames in DeepQA.

that if *Person* and *GeoPoliticalEntity* are disjoint, then every subclass of *Person* is disjoint with every subclass of *GeoPoliticalEntity* (e.g., *Musician* is disjoint with *Country*). Our additions to the YAGO type system comprise approximately 200 hand-curated explicit disjoint relations.

Handcrafting domain-specific knowledge

There are a few common subjects, such as *U.S. Presidents*, *works of Shakespeare*, *U.S. States*, and *Countries*, as well as a few categories of puzzle questions, for which it does make practical sense to build special-purpose templates, or *frames*, for the question types and to populate the frames with appropriate instance data.

Frames are coherent groups of related concepts (based on the classical knowledge representation [KR] notion [18]) and can be defined in isolation or with inter-frame relations. These are particularly helpful for questions that are better suited to an exhaustive generate-and-test solution using structured knowledge than to a search through unstructured sources. For example,

> BRAIN MAUL!: The 4 U.S. states that begin with the word "new." (Answer: "New York, New Jersey, New Hampshire, New Mexico")
> PRESIDENTIAL ELECTIONS: The only 2 Democratic Presidents defeated for reelection since the Civil War. (Answer: "Cleveland, Carter")

One could search for the keywords in these questions and hope to find passages that explicitly answer them (e.g., "New Mexico is a U.S. state that begins with the word 'new'."). This could potentially provide independent support for four different answers, and DeepQA could combine these to form a single final answer during answer merging and ranking [5]. However, some of these characteristics are unlikely to be asserted in text but are easy to verify given an answer. Lexical constraints (e.g., begins with "*new*") can be checked by inspecting a string, whereas semantic constraints (e.g., since the Civil War) can be checked using structured knowledge.

A special consumer of precise structured domain knowledge is the DeepQA semantic frame component. **Figure 1** highlights portions of the definitions for two sample and interrelated frames in DeepQA: Book and Movie. Whereas frames are just graphs, the nodes and arcs represent the *expectation* that instance-based graphs conforming to these frames will have a certain structure. In other words, we expect books to have an author, publication date, *and* topic. Often, clues will specify some parts of the frame explicitly, enough to select some instance of the frame, and the answer will be one of the parts not specified in the question, e.g., "Bram Stoker's famous book on vampires was published in this year". Frame definitions are hand-built from analysis of the domain and the most common types of things that appear in questions, and they are particularly helpful for questions combining complex constraints over interrelated frames that frequently occur in Final Jeopardy!, e.g., "A 1992 movie starring Anthony Hopkins was based on an 1897 book by this author".

On the basis of our analysis of many Jeopardy! clues, and particularly focusing on Final Jeopardy!, we built frames for U.S. presidents and vice presidents, U.S. states and their capitals, countries and their capitals, books, movies, and awards (specifically Nobel Prizes and some kinds of Oscars and Pulitzer Prizes). We extracted topic-relevant data from a variety of structured and semistructured sources—e.g., Wikipedia lists and tables—manually vetting the results. We also exploited the structured knowledge in DBpedia and Freebase by mapping to and from our hand-crafted frames.

Using structured data in DeepQA

In this section, we describe several components in the DeepQA system that use structured data. The first, and perhaps most error-prone, step in using structured data is matching strings that come from the question or candidate answers to represented entities in the source. There are many entities with the same name and many names for the same

entity. This problem, which we call Entity Disambiguation and Matching, pervades all the uses of structured data described below and is described in more detail in [12]. The remainder of this paper assumes that the string-to-entity matching has been done.

Temporal and geospatial reasoning

The DeepQA system deals with many questions that express temporal or geospatial relations involving the correct answer. The system attempts to gather evidence for these relations in order to support or refute potential answer candidates and does so by primarily using structured sources. The evidence is then expressed as a set of "answer scores" (features) weighed by the DeepQA system in ranking candidates.

Each answer scorer operates in two phases. During the question analysis phase [19], the system identifies information in the question that the answer scorers can use to gather evidence for the candidates and may even decide whether a particular scorer should apply for the given question. In the case of temporal scoring, question analysis would detect temporal expressions in the question and determine whether the correct answer is related to that temporal expression. On the basis of this information, during evidence gathering [4], an answer scorer may run in the pipeline to seek specific types of evidence for ranking the candidate answers. Our temporal and geospatial answer scorers, for instance, search structured resources to determine whether candidate answers are temporally compatible with the question or have the same geospatial relation as that expressed in the question. The remainder of this section presents details of these two types of answer scorers in the DeepQA system.

Detecting and scoring temporal relations

The main temporal relation we detect is a "TLink" [14], which specifies the end points of a generic temporal relationship—i.e., an entity and a time/date expression that are connected via a dependency path in the parse of the question. Unless specialized to a specific relation such as `birthDate` or `deathDate`, "TLinks" are untyped and, thus, are not directly usable for table-based validation. However, the presence of a TLink involving the focus in the question is indicative of a temporal relationship that needs to be considered on all candidates. DeepQA, therefore, includes answer scorers that measure the degree to which candidate answers are "temporally compatible" with the argument of the TLink.

One way to measure temporal compatibility of an answer candidate is to determine whether the time expression in the clue is chronologically before the entity came into existence. The hypothesis here is that an entity is unlikely to be associated with dates before it ever came into existence and, thus, could be down-weighted as an answer to a question discussing such dates. The date compatibility scorer uses the temporal knowledge base containing birthdates of entities and posts a Boolean "before birth" feature indicating whether the time expression in the question is chronologically before the entity came about. We did also look into an "after death" feature for time expressions in the question occurring chronologically after the "death" of the entity. However, we found, in many cases, that entities do have events associated with them even after their death (such as posthumous awards, celebrations of their birth, etc.), and thus, a feature based on dates after the entity's "death" does not have any impact on the Jeopardy! task.

Another option for scoring the temporal compatibility of answer candidates to the question is by matching the time expression in the question to a date of a significant or important event associated with the entity, using a set of "important" dates. For this purpose, our temporal knowledge base stores, for each entity, the frequency counts of time expressions appearing in the same sentence as the entity on the entity's Wikipedia page.

Our temporal match scorer posts two temporal match scores for the DeepQA model to use in ranking the candidates: an absolute temporal score and a temporal ratio score. The absolute temporal score is the frequency of the matched date for an entity, whereas the temporal ratio score is the ratio of the absolute temporal score of the entity to the total frequency of all dates for that entity.

Detecting and scoring spatial relations

The spatial relations that we detect between two geo-entities include relative direction ("This state which lies to the NE of Nebraska"), borders ("This country that shares its boundary with Argentina. . ."), containment ("Kosovo is now an autonomous province within this Balkan republic"), and the subjective notions of "near" or "far" ("This port close to Norfolk"). These relations were chosen because of their prominence in Jeopardy! questions and the good coverage they get in our structured sources.

Having detected spatial relations in the question, we map the entity argument strings to resources in structured knowledge bases (DBpedia and Freebase) using the EDM algorithm [12] and use a manually compiled list of DBpedia relations to obtain corresponding geospatial information. For example, latitude information is given by any one of the following relations in DBpedia: georss.org/georss/point, DBpedia.org/property/latitude, or wgs84_pos#lat. If the entity does not have any latitude or longitude information associated with it but does have a geographical location specified (via a relation such as `locatedIn`, `headquarters`, etc.), we obtain coordinates for the location and use it as being representative for the entity. For example, the entity IBM does not have any coordinates associated with it; however, IBM has a *headquartered* relation with the value "Armonk, NY", and thus, we infer

geo-coordinates for IBM as those of "Armonk, NY", which is useful in cases where the question asks for, e.g., a U.S. company.

Geo-coordinate information is used to compute the relative direction of two geo-entities and for computing spatial distance (for "near" or "far" relations). Obviously, coordinate information alone is not sufficient to compute borders or containment relations, which are explicitly checked in Freebase data. The spatial answer scorer adds a binary feature in the model indicating whether the spatial constraints in the clue are satisfied by the candidate answer. When trying to satisfy these constraints, the scorer takes advantage of the symmetry of the borders relation and transitivity of the containment relation.[1]

A particular challenge in geospatial analytics is to interpret subjective notions of proximity such as being near or far, which depend on the magnitude of size of entities and are thus difficult to estimate. Our approach is to use the spatial distance between the candidate answer and the geo-entities in the clue and add one of two spatial-distance features (*spatial-distance-near* and *spatial-distance-far*) based on whether the question specifies the answer to be near or far from geo-entities mentioned. We then take advantage of standardized features in the model [5], which normalize a feature score relative to the mean and standard deviation of the feature across all candidate answers for the given question. As a result, the standardized spatial-distance features measure the relative difference in spatial distance between candidate answers from geo-entities in the clue, and the machine learning uses this information during training to learn an appropriate weight for the features, based on the proximity desired—e.g., the *spatial-distance-near* feature gets a negative weight in the model, as the further away the candidate answer is from the geo-entities in the clue (compared with the other candidates), the less likely it is to be correct.

We also employ the spatial-distance-near feature in cases where an explicit "near" relation is not detected. Whenever a question makes a reference to a geospatial entity and the answer is (according to its LAT) a geospatial entity as well, an implied proximity relationship is hypothesized. For example, given the question "River that connects Lake Powell in Utah with the Gulf of California", the answer is "near" both Utah and California.

Evaluating the impact of temporal and spatial analysis

We evaluated the impact of the temporal and spatial answer scorers in Watson by running on a blind Jeopardy! question set of 3,508 questions. The baseline configuration used was the DeepQA system with only one answer-scoring component—the HUTT Type Coercion answer scorer [12]. Results of adding the components are shown in **Tables 1** and **2**, respectively.

Table 1 Temporal questions scored; 662 temporal questions out of 3,508-question test set.

	Baseline	*Plus temporal*
Accuracy	66.62%	67.37% (+0.75%)
Precision@70	80.82%	81.9% (+1.08%)

Table 2 Spatial questions scored; 374 spatial questions out of the 3,508-question test set.

	Baseline	*Plus spatial*
Accuracy	65.78%	67.65% (+1.87%)
Precision @ 70	78.63%	80.15% (+1.52%)

Both components fired on a relatively small set of questions; however, they brought about an improvement of approximately 1% to 2% in QA accuracy and Precision@70 (i.e., accuracy for the top 70% of the questions that the system is most confident about) on the questions fired. The end-to-end impact is not statistically significant (hence, not reported in the table); although, as noted earlier, the presence of temporal and spatial evidence is very useful for explanation purposes.

We do not provide component-level evaluation of temporal and spatial relation detection because we have no annotated gold standard. In any case, our temporal analysis is shallow, in the sense that we are detecting TLinks but attempt no deeper semantic labeling, e.g., `birthDate`, `deathDate`, `creationOf`, and so forth. The TLinks are read off directly from the predicate-argument structure, and their correctness is reflected in an overall syntactic analysis evaluation [20]. As for spatial relations, our development data is tagged at question class level only—whether geospatial reasoning and scoring is appropriate for this question—and hence not rich enough for relation detection evaluation.

Furthermore, although most of the spatial relation detectors described here use patterns over well-formed syntactic structures, in the geospatial domain, we also utilize shallower patterns to account for the variability of expressing relative locational, or bordering, information. Such patterns typically overgenerate, particularly if applied over unconstrained text (e.g., from supporting passage retrieval). However, the constraint validation in the geospatial domain is focused on candidate answers only and the relational information

[1] It is worth noting that in the infamous wrong answer of "Toronto" to the Final Jeopardy! question in Watson's televised Jeopardy! match, the "U.S. City" constraint was in the category, not the clue, and since spatial analytics is not applied to the category, no spatial constraint was detected or processed.

about them in structured knowledge bases. Spatial relations are not considered by our passage-matching framework [4]; hence, we get the benefit of detecting more relations than just the local syntax would yield, without having to worry about managing the noise that would result if those patterns were applied to passages.

Taxonomic reasoning in DeepQA

The TyCor components in the DeepQA system check whether a particular candidate answer's type matches the LAT of the question. DeepQA uses a suite of TyCor algorithms to accomplish this task [12]. Here, we describe one specific component of the TyCor suite called YAGO TyCor and evaluate its impact on the DeepQA system. The YAGO TyCor performs taxonomic reasoning to address the type-matching problem and uses structured data from DBpedia and YAGO.

Two important aspects of the TyCor problem are mapping the textual mention of a candidate answer to an entity resource in a structured knowledge base and matching the LAT of the question to a class in a type system or ontology. The former, which is referred to as EDM, was already mentioned earlier; the latter is the Predicate Disambiguation and Matching (PDM) problem.

The YAGO TyCor EDM implementation is shared across many of the TyCors and maps an entity mention string to a ranked list of DBpedia resources with associated confidence scores, using a variety of unstructured, semistructured, and structured resources such as Wikipedia disambiguation pages, redirects, and anchor-link information, as well as the context of the candidate string (see [12] for more details).

The YAGO TyCor PDM algorithm starts by obtaining the set of all YAGO WordNet concepts, whose labels or IDs explicitly match the lexical type string. We then rank the concepts on the basis of two pieces of information—a concept's WordNet sense rank and the number of instances of it in DBpedia—and provide a confidence score based on a weighted combination of the two factors. The rationale for this is to use the notion of type popularity, since it seems to produce reasonably good approximations for the intended semantics of the lexical type in our use cases.

There are two additional features of our PDM algorithm that help improve its precision and recall. The first is an optional domain-specific type-mapping file that records observed differences in the domain of top word senses. For example, based on a historical analysis of Jeopardy! questions, we found the LAT "star" refers to the sense of star as a *movie star* roughly 75% of the time, with the remaining cases referring to the *astronomical object*, despite the latter being WordNet's top sense. Hence, in the type-mapping file, we map this LAT to the corresponding YAGO WordNet Concepts with confidences of 0.75 and 0.25, respectively.

The second heuristic we use in PDM helps improve its recall. We estimate a statistical relatedness between two types by computing the conditional probability that an instance with type A also has type B using the metric: *NI* (A and B)/*NI* (A), where *NI* is the number of instances of the concept in DBpedia (including instances of its subtypes). In PDM, if the lexical type matches some YAGO type, we expand it to include related types based on their conditional probabilities exceeding some threshold (0.5).

Computing TyCor using ontology type alignment

EDM matches the candidate answer to a set of instances in DBpedia, obtaining corresponding YAGO WordNet types. PDM maps the LAT to a set of YAGO WordNet concepts. After performing EDM and PDM, the TyCor problem is now reduced to that of ontology type alignment. We compare the instance types with the LATs and produce a final TyCor score based on the nature and strength of alignment. The type alignment is done using a prioritized rule-based approach. The following set of matching rules are used (in order of preference):

- *Equivalent/subclass match*—When the instance type and the LAT type are either equivalent (synonyms) in YAGO or the instance type is a subclass (hyponym) of the LAT type.
- *Disjoint match*—When the instance type and the LAT type are found to be mutually exclusive or disjoint based on the disjointness axioms added to YAGO. In this case, we produce a strong *negative* type alignment match, which generates an anti-TyCor feature [12].
- *Sibling match*—When the instance type and the LAT are siblings (i.e., share the same parent concept) in YAGO. In this case, we also consider the depth of the types in the tree; the more low-level the types, the more likely that semantics of the siblings are closely related, and hence, the stronger the degree of match.
- *Superclass match*—When the instance type is a superclass (hypernym) of the LAT, we assign a small type alignment score. This seems counterintuitive since the candidate answer is supposed to be an instance of the LAT and not vice versa. However, we have seen cases where checking the type alignment in the opposite direction helps, either due to inaccuracies in the EDM or PDM step or due to source errors or because the question itself asks for the type of a particular named entity.
- *Statistical relatedness exceeds threshold*—When the statistical type relatedness between the instance type and the LAT, computed as described above, exceeds an empirically determined threshold.
- *Depth of lowest common ancestor (LCA) exceeds threshold*—When the LCA of the instance type and the LAT is deep in the taxonomy; it could imply that the types are strongly related, although there

Table 3 Evaluation of accuracy and precision.

Baseline	*Baseline accuracy (Precision@70)*	*Plus YAGO (An)TyCor*	*Plus all TyCors except YAGO (An)TyCor*	*Plus all TyCors including YAGO (An)TyCor*
1. DeepQA system with no answer scorers	50.03% (Precision@70: 63.44%)	54.28% (Precision@70: 67.63%)	58.55% (Precision@70: 75.37%)	58.64% (Precision@70: 75.24%)
2. DeepQA system with all answer scorers except TyCors	65.48% (Precision@70: 81.43%)	68.39% (Precision@70: 83.84%)	69.38% (Precision@70: 86.93%)	70.35% (Precision@70: 87.42%)

may not be a direct-subclass or sibling relationship among them.

The thresholds used in the type matching conditions above and the weights of the respective rules are manually assigned, based on an empirical evaluation conducted by running the algorithm on a large number of test cases.

Evaluation

We evaluated the impact of the YAGO (An)TyCor scorers on two baseline DeepQA systems using a blind test set of 3,508 Jeopardy! questions (the same set used for evaluating spatial and temporal scoring components). The first baseline consisted of a DeepQA system that used all the question analysis and candidate generation components but did not include any answer scorers. The second baseline used all answer scorers except for the TyCor components. Results of the experiments, including final QA accuracy and the corresponding Precision@70, are shown in **Table 3**.

Adding only the YAGO TyCor and (An)TyCor answer-scoring components to the baselines showed a significant improvement in both accuracy (3%–4%) and Precision@70 (2.5%–4%), demonstrating the value of this component. Using all the TyCor scorers except YAGO produced a much larger gain in accuracy and Precision@70 over the baseline, which is expected, given the suite of TyCor algorithms developed [12] that use a wide variety of unstructured, semistructured, and structured resources. However, adding the YAGO TyCor and (An)TyCor scorer to the TyCor suite (last column in the table) produced an additional improvement in accuracy, particularly on the second baseline (which resembles our full DeepQA system), where it went up almost 1%.

Knowledge-based QA using semantic frames

Despite the known brittleness (described in the introduction of this paper) of a traditional structured lookup approach to QA, there are some narrow topic areas that occur in Jeopardy! for which this approach is well suited. Thus, we developed a distinct subsystem for Watson that departs from the overall DeepQA design philosophy and, instead, provides an independent pipeline—narrow, brittle, but precise—for handling questions in those topic areas. This pipeline complements the full DeepQA pipeline and provides an incremental improvement in accuracy on the very small subset of questions for which it is applicable. As described below, some of the questions in that subset are ones for which the core DeepQA pipeline is poorly suited, which makes this independent subsystem a particularly useful complement to the core pipeline in those cases.

Given a natural-language question, this subsystem uses a collection of frame recognition algorithms to select frames and maps the textual representation of the question to the various parts of the frames. In general, the interpretation of a piece of text may span more than one frame, and there may be multiple possible interpretations that generate alternative sets of frame instances. Some of this is addressed by the general-purpose DeepQA semantic analysis components [20], and we added some recognition capabilities for the frame subsystem. For example, in some cases, when a topic area for a question has been identified, it is very likely that certain relationships hold between two entities of certain types; e.g., if a question refers to a language and a country, there is a high likelihood that the language is a national language of the country. In those cases, we assume that the relationship holds as long as we can find no evidence to the contrary. For example, consider the following clue:

> LANGUAGE: The lead singer of the band Dengue Fever is from this country & often sings in Khmer. (Answer: "Cambodia")

The clue does not explicitly ask for the country whose language is Khmer. However, given that we are able to identify the topic area and the frame elements involved, the frame subsystem assumes that the country being asked for is a country whose national language is Khmer. For arbitrary text, we would not want to assume that any sentence containing a country and a language is asserting that the latter

is the national language of the former. As a result, our general-purpose relation detection components [13] do not have a rule of this sort. However, in the context of QA, we have observed that this rule is accurate enough for these two semantic types. We also have other rules for identifying frame slot-fillers that are less aggressive than simple co-occurrence but are more aggressive than our general-purpose relation detectors.

When the frame recognizers are applied to an input question, it is possible that the focus of the question (i.e., the thing being asked for) corresponds to one of our predefined frames (in this example, the focus indicates the frame *country*). In such cases, we can answer the question by searching all the instance frames of the given type. We currently employ an exhaustive search method for finding frame instances and have begun working on an artificial intelligence planning approach (see the section on future work).

Exhaustive search is applicable when the focus frame is of a closed type with relatively few instances (e.g., U.S. presidents or U.S. states) and when our structured sources have good coverage for the slots detected in the question. We can simply enumerate all possible values for the focus frame element, look up their slot values in our sources, and check for contradictions with what is stated in the question. Sometimes, the question asks for a pair (or an *n*-tuple) of related elements, and this can be solved by enumeration as well. For example, consider the following questions:

> PRESIDENTS: The only 2 consecutive U.S. presidents with the same first name. (Answer: "James Madison and James Monroe")
>
> TRICKY QUESTIONS: Of current U.N. member countries with 4-letter names, the one that is first alphabetically. (Answer: "Chad")

These are examples of questions that are poorly suited to be handled well by the core DeepQA pipeline, which focuses on generating candidates via unstructured search and gathering primarily unstructured evidence supporting the answer. It is unlikely that we would find a textual source that explicitly asserts that Madison and Monroe are consecutive U.S. presidents with the same first name and even more unlikely that we would find one that asserts that Chad is the alphabetically first four-letter member of the United Nations. For these questions, the frame subsystem enumerates all entities of the given type (U.S. presidents or countries) and looks up the relevant characteristics. For characteristics that are comparative (e.g., "the same first name" or "the first alphabetically"), this approach has the complete set of answers that meet the other constraints in the question (e.g., being consecutive and having a four-letter name) and can compare those answers to select ones that satisfy those comparative requirements, again assuming we can recognize the comparative requirements expressed in English and process them. We have tailored constraint recognizers for the Jeopardy! domain that include alphabetical, directional (e.g., "furthest north"), numbers of letters, start/end/contains letter, numerical (e.g., "most" and "first"), and a few others.

If multiple alternative answers satisfy the constraints that we formally encode, we attempt to rank them using the core DeepQA system's candidate answers. Specifically, Watson attempts to answer questions using both the main DeepQA QA components and the frame subsystem. When the frame subsystem identifies a question as being in a topic area for which it has exhaustive knowledge and it provides answers to that clue, those answers are given precedence. If there are answers that come from both the frame subsystem and the main pipeline, we use the confidence from the DeepQA answer ranking [5] for the main pipeline to determine the final confidence and ranking for the answers. This allows us to combine insights about constraints from the frame subsystem with insights about the remaining text of the clue, which may include relevant concepts that our knowledge base does not cover.

For 90,771 non-Final Jeopardy! questions, the frame subsystem provided one or more answers for 2,131 questions (2.3%). In only 257 questions (0.3%) did it identify exactly one answer. Of those, 223 were correctly answered, for an accuracy of 87%, which is much higher than Watson's overall accuracy of approximately 70%.

On a set of 1,240 Final Jeopardy! clues, the system provided one or more answers on 86 (6.9%) questions and unique answers on 5 (0.4%). Of those five clues, four were right (80%). It is not surprising that coverage is a little higher on Final Jeopardy!, since we selected topic areas to address based (in part) on observed frequency in Final Jeopardy!. When the frame subsystem gets a question wrong, it is almost always a result of a failure in the recognition, such as assuming that some relationship holds when it does not. However, given the very low coverage, we did not want to make the recognition any more cautious than it already is.

The frame subsystem has met our expectations for this approach. It is more precise than the core DeepQA system alone, and it is able to address a few infrequently occurring types of questions for which the core DeepQA system is not well suited. The low coverage of the frame subsystem makes it unsuitable as a general solution to open-domain QA, but it does complement the core DeepQA approach on a narrow range of questions.

Other uses of structured knowledge in Watson

Apart from the three main areas discussed above, several other DeepQA components use some amount of structured knowledge when performing their tasks. The *Answer Lookup* candidate generator [11] uses relations detected in the clue

during question analysis to generate candidate answers by querying a structured source for those relations. Similarly, the *Evidence Diffusion*-based answer merger [5] uses relations found between two candidate answers in a structured source to diffuse or transfer evidence from one candidate answer to another, depending on the context of the clue.

The *Answer-in-Clue* component, which down-weighs candidate answers that are referenced in the clue (explicitly or using alternate names), uses entity equivalence relations from structured sources to determine if the candidate answer is the *same as* some named entity in the clue.

In addition, in the context of Jeopardy!, when the system is given access to previous answers in the same category, within-category learning components use relationship information to infer common links for all answers in that category. For example, consider the following set of Jeopardy! clues that appear in the same category:

($200) WHO SENT ME TO THE SC: Ruth Bader Ginsberg. (Answer: "Bill Clinton")

($400) WHO SENT ME TO THE SC: Clarence Thomas. (Answer: "George H W Bush")

($600) WHO SENT ME TO THE SC: Thurgood Marshall. (Answer: "Lyndon Johnson")

In each of the clues, the question is asking for the president who nominated the corresponding Supreme Court (SC) justice mentioned. The system "gleans" this fact, not by understanding the category but by using Bayesian inference instead, observing that the same relationship is present in structured sources between previous answers (within the same category) and corresponding entities in the clue.

Related work

QA systems were historically conceived as natural-language interfaces to databases in order to relieve the user from dealing with complex structured query languages. Initial attempts along these lines were the BASEBALL [1] and LUNAR [2] systems, where the queries had to (in effect) conform to a controlled natural-language syntax, and the target domain was inherently bound to the underlying database. A more recent example is the WolframAlpha "Computational Knowledge Engine" (http://www.wolframalpha.com), which is designed to perform searches and calculations over manually curated structured data. The system has a natural-language interface, and many answers include dynamically generated data often involving mathematical computations. This engine has limitations similar to other structured data-based QA systems in that it is extremely brittle in its performance; it has high precision in certain narrow domains such as science and math (provided the question is well-defined and hence correctly interpreted) but suffers from recall issues, both in question understanding given syntactic and/or lexical variability, and in the coverage of the manually curated data.

A more prominent move was observed toward using unstructured information and document retrieval in QA systems [21] because of the exponential increase in machine-readable textual information, particularly with the advent of the Web. A notable example is the START system developed at MIT [22], which used both structured and unstructured information. It focused on "reversible" transformation rules that translate natural language into T-expressions (T stands for ternary) of the form ⟨subject relation object⟩ and then back into natural language when providing answers. START also exploits Omnibase [23], a "virtual" database aimed at executing START-generated queries against multiple Web knowledge sources, thus providing a uniform access interface to them. While START essentially relies on explicitly matching natural-language annotations across the question and the underlying content, Watson, which is based on the DeepQA framework, uses a much more elaborate machine learning scheme, along with a rich and diverse set of analytics, to combine information from both structured and unstructured sources, thus giving it higher recall.

With regard to our frame processing system in DeepQA, we note that frame semantics, which was first introduced in cognitive linguistics by Charles Fillmore [24], became popular in NLP through the development of FrameNet [25]. QA systems that explicitly take into account local frame contexts include those described in [26] and [27]. These systems take the common approach of running frame annotators on both the question text and the candidate answer passages and applying different semantic matching algorithms against the frames detected. This approach is inflexible because of the sparseness of automatic annotation and the lack of lexical coverage. In contrast, the approach we take in Watson is more open-ended in that we allow partial interpretation of frames, only filling in certain frame slots on the basis of information available in the question, and then subsequently filling the knowledge gap iteratively (on the basis of local and global frame context), using the whole structured and unstructured knowledge embedded in Watson.

Future work

We are exploring various extensions to our structured data algorithms. For the problem of ontology-based type coercion, we plan to use a wider and more diverse set of ontologies available in the Linked Open Data cloud to increase our domain coverage and obtain finer-grained type information [28].

For frames, we are working on an artificial intelligence planning-based solution to interpreting and populating

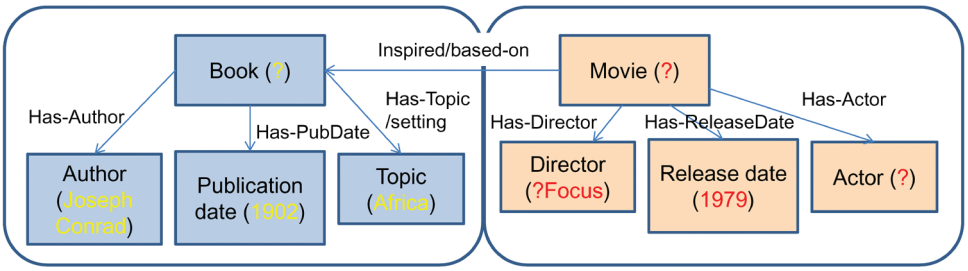

Figure 2

Frame recognizers map this question text to the inter-related *Book* and *Movie* frame instances: "WAR MOVIES: A 1902 Joseph Conrad work set in Africa inspired this director to create a controversial 1979 war film."

frames. In this approach, a planner starts with what is currently known (i.e., initial populated frame components) and move toward the focus frame, acquiring necessary information along the path (filling the knowledge gap) and spanning multiple frames if necessary. For example, consider the following question:

> WAR MOVIES: A 1902 Joseph Conrad work set in Africa inspired this director to create a controversial 1979 war film.

The frame recognizers map this question text to instances of the *Book* and *Movie* frames, as shown in **Figure 2**.

Based on the frame instances populated, the planner generates the following plan:

1. Solve for the Book by calling DeepQA with the generated question: "This 1902 book by Joseph Conrad is about Africa." (Answer: "*Heart of Darkness*").
2. Use our structured data to verify that the publication date of the book is 1902.
3. Invoke DeepQA to solve for the Movie given the Book, using the question "*Heart of Darkness* inspired this controversial 1979 war film." (Answer: "*Apocalypse Now*").
4. Use our structured data to verify that the release date of the movie is 1979.
5. Use our structured data to look up the director of the movie (answer: "Francis Ford Coppola").

Some steps (in particular, the calls to DeepQA) associate probabilities with their results, and the planner uses these to merge results from different analyses. The planner generates multiple plans for a question and executes the plans in parallel. Finally, it selects the top answer to the question as the result with the highest confidence across all frame interpretations.

An advantage of this planning paradigm is that it helps the DeepQA system use information from unstructured text analysis to aid structured data inference and vice versa. Another advantage is the declarative nature of the framework, making it possible for developers to add and refine semantic frame definitions and recognizers. The underlying planner works off the declarative specifications, composing plans and executing them on the basis of content extracted from the question.

Finally, we are working on techniques to automatically induce typical concepts and relations in a frame from a large corpus using data mining. For example, by observing frequent occurrences of "authoring" and "publishing" relationships in the context of books (in a background corpus), we can induce author and publication-date elements in a Book frame. Such techniques also extend to discovering inter-frame relationships (e.g., between the Book and Movie frame shown above) using custom corpus processing resources [17]. We expect these techniques to substantially reduce the human effort needed to engineer a general-purpose semantic frame-based system.

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

**Trademark, service mark, or registered trademark of Jeopardy Productions, Inc., Wikimedia Foundation, Cycorp, Inc., Wolfram Group, or Trustees of Princeton University in the United States, other countries, or both.

References

1. B. Green, A. Wolf, C. Chomsky, and K. Laughery, "BASEBALL: An automatic question answerer," in *Proc. Western Joint IRE-AIEE-ACM Comput. Conf.*, Los Angeles, CA, 1961, pp. 219–224.
2. W. Woods, "Lunar rocks in natural English: Explorations in natural language question answering," in *Linguistic Structures Processing, Fundamental Studies in Computer Science*. New York: Elsevier North-Holland, 1977, pp. 521–569.
3. D. A. Ferrucci, "Introduction to 'This is Watson'," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 1, pp. 1:1–1:15, May/Jul. 2012.

4. J. W. Murdock, J. Fan, A. Lally, H. Shima, and B. K. Boguraev, "Textual evidence gathering and analysis," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 8, pp. 8:1–8:14, May/Jul. 2012.
5. D. C. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. Duboue, L. Zhang, Y. Pan, Z. M. Qiu, and C. Welty, "A framework for merging and ranking of answers in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 14, pp. 14:1–14:12, May/Jul. 2012.
6. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "DBpedia—A crystallization point for the web of data," *J. Web Semantics, Sci. Serv. Agents World Wide Web*, vol. 7, no. 3, pp. 154–165, Sep. 2009.
7. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.
8. J. Broekstra, A. Kampman, and F. Harmelen, "Sesame: A generic architecture for storing and querying RDF and RDF schema," in *Proc. ISWC*, 2002, pp. 54–68.
9. J. Curtis, G. Matthews, and D. Baxter, "On the effective use of cyc in a question answering system," in *Proc. IJCAI Workshop Knowl. Reason. Answering Questions*, Edinburgh, Scotland, 2005, pp. 61–70.
10. WolframAlpha. [Online]. Available: http://www.wolframalpha.com/
11. J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty, "Finding needles in the haystack: Search and candidate generation," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 6, pp. 6:1–6:12, May/Jul. 2012.
12. J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. C. Gondek, L. Zhang, and H. Kanayama, "Typing candidate answers using type coercion," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 7, pp. 7:1–7:13, May/Jul. 2012.
13. C. Wang, A. Kalyanpur, J. Fan, B. K. Boguraev, and D. C. Gondek, "Relation extraction and scoring in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 9, pp. 9:1–9:12, May/Jul. 2012.
14. J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz, "TimeML: Robust specification of event and temporal expressions in text," in *Proc. IWCS-5*, 2003, pp. 337–353.
15. F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago—A core of semantic knowledge," in *Proc. 16th Int. WWW Conf.*, 2007, pp. 697–706.
16. G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
17. J. Fan, A. Kalyanpur, D. C. Gondek, and D. A. Ferrucci, "Automatic knowledge extraction from documents," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 5, pp. 5:1–5:10, May/Jul. 2012.
18. M. Minsky, "Minsky's frame system theory," in *Proc. Workshop Theor. Issues Natural Lang. Process.*, Cambridge, MA, 1975, pp. 104–116.
19. A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll, "Question analysis: How Watson reads a clue," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 2, pp. 2:1–2:14, May/Jul. 2012.
20. M. C. McCord, J. W. Murdock, and B. K. Boguraev, "Deep parsing in Watson," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 3, pp. 3:1–3:15, May/Jul. 2012.
21. E. M. Voorhees, "Overview of the TREC 2002 question answering track," in *Proc. 11th Text REtrieval Conf.*, 2002, pp. 115–123.
22. B. Katz, "Annotating the world wide web using natural language," in *Proc. 5th RIAO Conf. Comput. Assisted Inf. Searching Internet*, 1997, pp. 136–159. [Online]. Available: http://groups.csail.mit.edu/infolab/publications/Katz-RIAO97.pdf
23. B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. McFarland, and B. Temelkuran, "Omnibase: Uniform access to heterogeneous data for question answering," in *Proc. 7th Int. Workshop Appl. NLDB*, 2002, pp. 230–234.
24. C. J. Fillmore, "Frame semantics," in *Linguistics in the Morning Calm*. Seoul, South Korea: Hanshin Publ. Co., 1982, pp. 111–137.
25. C. F. Baker, C. Fillmore, and J. Lowe, "The Berkeley FrameNet project," in *Proc. Joint Annu. Meet. Assoc. Computat. Linguistics Int. Conf. Computat. Linguistics*, Montreal, QC, Canada, 1998, pp. 86–90.
26. D. Shen and M. Lapata, "Using semantic roles to improve question answering," in *Proc. EMNLP Conf.*, Prague, Czech Republic, 2007, pp. 12–21.
27. B. Ofoghi, J. Yearwood, and L. Ma, "FrameNet-based fact-seeking answer processing: A study of semantic alignment techniques and lexical coverage," in *Proc. 21st Australasian Joint Conf. LNAI*, vol. 5360. Auckland, New Zealand: Springer-Verlag, 2008, pp. 192–201.
28. Y. Ni, L. Zhang, Z. Qiu, and C. Wang, "Enhancing the open-domain classification of named entity using linked open data," in *Proc. 9th ISWC*, 2010, pp. 566–581.

Received July 18, 2011; accepted for publication December 9, 2011

Aditya Kalyanpur *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (adityakal@us.ibm.com)*. Dr. Kalyanpur is a Research Staff Member at the IBM T. J. Watson Research Center. He received his Ph.D. degree in computer science from the University of Maryland in 2006. His research interests include knowledge representation and reasoning, natural-language processing, statistical data mining, and machine learning. He joined IBM in 2006 and worked on the Scalable Highly Expressive Reasoner (SHER) project that scales ontology reasoning to very large and expressive knowledge bases. Subsequently, he joined the algorithms team on the DeepQA project and helped design the Watson question-answering system. Dr. Kalyanpur has over 25 publications in leading artificial intelligence journals and conferences and several patents related to SHER and DeepQA. He has also chaired international workshops and served on W3C Working Groups.

Branimir K. Boguraev *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (bran@us.ibm.com)*. Dr. Boguraev is a Research Staff Member in the Semantic Analysis and Integration Department at the IBM T. J. Watson Research Center. He received an engineering degree in electronics from the Higher Institute for Mechanical and Electrical Engineering in Sofia, Bulgaria (1974) and a diploma and Ph.D. degrees in computer science (1976) and computational linguistics (1980), respectively, from the University of Cambridge, England. He worked on a number of U.K./E.U. research projects on infrastructural support for natural-language processing applications, before joining IBM Research in 1988 to work on resource-rich text analysis. From 1993 to 1997, he managed the natural-language program at Apple's Advanced Technologies Group, returning to IBM in 1998 to work on language engineering for large-scale, business content analysis. Most recently, he has worked, together with the Jeopardy! Challenge Algorithms Team, on developing technologies for advanced question answering. Dr. Boguraev is author or coauthor of more than 120 technical papers and 15 patents. Until recently, he was the Executive Editor of the Cambridge University Press book series *Studies in Natural Language Processing*. He has also been a member of the editorial boards of *Computational Linguistics* and the *Journal of Semantics*, and he continues to serve as one of the founding editors of *Journal of Natural Language Engineering*. He is a member of the Association for Computational Linguistics.

Siddharth Patwardhan *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (siddharth@us.ibm.com).* Dr. Patwardhan is a Post-Doctoral Researcher in the Knowledge Structures Group at the T. J. Watson Research Center. He received a B.E. degree in computer engineering from the University of Pune in 2001, an M.S. degree in computer science from the University of Minnesota in 2003, and a Ph.D. degree in computer science from the University of Utah in 2010. He has been working at the IBM T. J. Watson Research Center since 2009, exploring research projects in natural-language processing and artificial

intelligence. He is a member of the Algorithms Team working on the IBM Jeopardy! challenge and is an author or coauthor of more than 25 technical publications covering his work on information extraction, opinion/sentiment analysis, computational lexical semantics, and question answering. Dr. Patwardhan is a member of the Association for Computational Linguistics and a member of the Association for the Advancement of Artificial Intelligence.

J. William Murdock *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (murdockj@us.ibm.com).* Dr. Murdock is a member of the IBM DeepQA Research Team in the T. J. Watson Research Center. In 2001, he received a Ph.D. degree in computer science from Georgia Tech, where he was a member of Ashok Goel's Design and Intelligence Laboratory. He worked as a post-doctorate with David Aha at the U.S. Naval Research Laboratory. His research interests include natural-language semantics, analogical reasoning, knowledge-based planning, machine learning, and self-aware artificial intelligence.

Adam Lally *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (alally@us.ibm.com).* Mr. Lally is a Senior Technical Staff Member at the IBM T. J. Watson Research Center. He received a B.S. degree in computer science from Rensselaer Polytechnic Institute in 1998 and an M.S. degree in computer science from Columbia University in 2006. As a member of the IBM DeepQA Algorithms Team, he helped develop the Watson system architecture that gave the machine its speed. He also worked on the natural-language processing algorithms that enable Watson to understand questions and categories and gather and assess evidence in natural language. Before working on Watson, he was the lead software engineer for the Unstructured Information Management Architecture project, an open-source platform for creating, integrating, and deploying unstructured information management solutions.

Chris Welty *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (cawelty@gmail.com).* Dr. Welty is a Research Staff Member in the Semantic Analysis and Integration Department at the T. J. Watson Research Center. He received a Ph.D. degree in computer science from Rensselaer Polytechnic Institute in 1995. He joined IBM in 2002, after spending 6 years as a professor at Vassar College, and has worked and published extensively in the areas of ontology, natural-language processing, and the Semantic Web. In 2011, he served as program chair for the International Semantic Web Conference, and he is on the editorial boards of the *Journal of Web Semantics*, the *Journal of Applied Ontology*, and *AI Magazine*.

John M. Prager *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (jprager@us.ibm.com).* Dr. Prager has been working in technical fields related directly or indirectly to question answering for most of his professional career. Most recently, while at the T. J. Watson Research Center, he has been part of the Watson project, building a system that plays the *Jeopardy!* quiz-show game. He has been involved in both the algorithms area, in particular working on puns and other wordplay, and the strategy area. Previously, he led IBM's successful entries in Text REtrieval Conference Question-Answering (TREC-QA) tasks, an annual evaluation at the National Institute of Standards and Technology (NIST). Prior to that, he worked in various areas of search, including language identification, web search, and categorization. He has contributed components to the IBM Intelligent Miner for Text product. For a while in the early 1990s, he ran the search service on www.ibm.com. While at the IBM Cambridge Scientific Center, Cambridge, MA, he was the project leader of the Real-time Explanation and Suggestion project, which would provide users with help by taking natural-language questions and processing them with an inference engine tied to a large repository of facts and rules about network-wide resources. He has degrees in mathematics and computer science from the University of Cambridge and in artificial intelligence from the University of Massachusetts; his publications include conference and journal papers, nine patents, and a book on Alan Turing.

Bonaventura Coppola *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (bcoppola@us.ibm.com).* Dr. Coppola is a Post-Doctoral Researcher in the Semantic Analysis and Integration Department at the IBM T. J. Watson Research Center. He received his bachelor's degree (2001) and master's degree (2003, with honors), both in computer engineering, from the University of Rome "Tor Vergata," and his Ph.D. degree (2009) in information and communication technologies from the University of Trento. He has been a Research Assistant at FBK-Irst and a Post-Doctoral Researcher at University of Trento, and he coauthored more than 15 technical papers in human language technology. In 2010, he joined the IBM DeepQA Team for the Jeopardy! challenge. Dr. Coppola focuses on frame-based semantic analysis of free-text document collections and its application to knowledge acquisition and question answering. His current effort is in porting the IBM Watson system to specialized application domains such as healthcare. He also contributes to the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program.

Achille Fokoue-Nkoutche *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (achille@us.ibm.com).* Mr. Fokoue-Nkoutche is a Research Staff Member in the Software department at the T. J. Watson Research Center. He received an M.S. degree in computer science in 1999 from Ecole Centrale Paris, France. In 2001, he joined IBM at the T. J. Watson Research Center, where his research work has focused on knowledge representation and reasoning, data management, information integration, programming, and query languages for Extensible Markup Language (XML). He is author or coauthor of 5 patents and 25 research papers.

Lei Zhang *IBM Research Division, China Research Lab, Haidian District, Beijing 100193, China (tozhanglei@qq.com).* Dr. Zhang received his Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University in China in 2005. His research interests include knowledge representation, Semantic Web, information retrieval, and statistical machine learning. After his Ph.D. study, he worked as a Research Staff Member in the Information and Knowledge Department of IBM Research—China. Since 2005, he and his team have worked on Semantic Web technologies and semantic search and their applications in the healthcare domain. Since 2008, he and his team have worked on using structured knowledge (including Semantic Web data) to help question-answering in the DeepQA project. He is active in several academic communities and is one of the major initiators of the China Semantic Web Symposium series, which started in 2007. He has been program committee member of conferences such as WWW, IJCAI (International Joint Conferences on Artificial Intelligence), ISWC (International Semantic Web Conference), etc. More recently, he was one of the local organizers of the ISWC 2010 conference.

Yue Pan *IBM Research Division, China Research Lab, Haidian District, Beijing 100193, China (panyue@cn.ibm.com).* Mr. Pan is the Chief Scientist for Information Analytics and Executive for Healthcare of IBM Research—China. He is a major advocator of ontology technology and Semantic Web research in IBM. He and his team's research work focuses on knowledge representation and reasoning, Semantic Web, information integration, search and question-answering systems, and medical informatics. In addition to research at IBM, he also serves as doctoral advisor in Shanghai Jiao Tong University. He has several patents and has published tens of papers in international conferences and journals. He is a member of the Association for Computing Machinery (ACM), IEEE (Institute of Electrical and Electronics Engineers), CCF

(China Computer Federation), and a committee member of YOCSEF (CCF Young Computer Scientists and Engineers Forum). He was the program co-chair of the International Semantic Web Conference 2010.

Zhao Ming Qiu *IBM Research Division, China Research Lab, Haidian District, Beijing 100193, China (qiuzhaom@cn.ibm.com)*.
Dr. Qiu was a Research Staff Member in the IBM China Research Laboratory (CRL). He received a Ph.D. degree in physics from Columbia University in 1985, after which he then joined the Institute of Theoretical Physics, Chinese Academy of Science. In 1996, he joined the IBM CRL. He has received an IBM Outstanding Technical Achievement award for his work on homepage translator in the Internet solution set. Dr. Qiu retired in 2006 and is now working part time in IBM CRL.