Finding needles in the haystack: Search and candidate generation

J. Chu-Carroll
J. Fan
B. K. Boguraev
D. Carmel
D. Sheinwald
C. Welty

A key phase in the DeepQA architecture is Hypothesis Generation, in which candidate system responses are generated for downstream scoring and ranking. In the IBM Watson™ system, these hypotheses are potential answers to Jeopardy!™ questions and are generated by two components: ***search*** *and* ***candidate generation****. The search component retrieves content relevant to a given question from Watson's knowledge resources. The candidate generation component identifies potential answers to the question from the retrieved content. In this paper, we present strategies developed to use characteristics of Watson's different knowledge sources and to formulate effective search queries against those sources. We further discuss a suite of candidate generation strategies that use various kinds of metadata, such as document titles or anchor texts in hyperlinked documents. We demonstrate that a combination of these strategies brings the correct answer into the candidate answer pool for 87.17% of all the questions in a blind test set, facilitating high end-to-end question-answering performance.*

Introduction

A key component in the IBM Watson* system is Hypothesis Generation, which is the process of producing possible answers to a given question. These candidate answers are scored by the Evidence Gathering and Hypothesis Scoring components [1–4] and are ranked by the Final Merging and Ranking component [5] to produce the final ranked list of answers. Since the outcome of Hypothesis Generation represents all possible candidates that the system will consider, it is crucial that a wide net be cast at this stage. It is also important, however, that the system includes the correct answer among its candidates without overwhelming the downstream answer scorers with noise. Too many wrong candidates reduce system efficiency and can potentially hamper the system's ability to identify the correct answer from the overly large pool of candidates.

In the IBM Watson system, the Hypothesis Generation phase consists of two components: *search* and *candidate generation*. In its search component, Watson adopts a multipronged approach to retrieve relevant content from its diverse knowledge resources. The search results may be documents or passages from unstructured sources or arguments that satisfy partially instantiated predicates from structured sources. Watson's search strategies extend the passage search approach adopted by most existing question-answering (QA) systems in two ways. First, Watson employs specific search strategies to exploit the relationship between titles and content in title-oriented documents (e.g., encyclopedia articles) to improve search recall. Second, Watson uses structured resources through queries based on syntactic and semantic relations extracted from the question.

The candidate generation component identifies potential answers to a question from the retrieved unstructured content. Most existing QA systems adopt a type-based approach to candidate generation with respect to a predefined type ontology. However, because of the broad range of lexical answer types (LATs) [6] observed in the Jeopardy!** domain, Watson relies on a type-independent approach to its primary candidate generation strategies—those of producing candidate answers from unstructured content. It uses the knowledge inherent in human-generated text and associated metadata, as well as syntactic and lexical cues in the search results, to identify salient concepts from text and hypothesize them as candidate answers.

Digital Object Identifier: 10.1147/JRD.2012.2186682

Note that the strategies described in this paper focus on questions whose answers can plausibly be found in Watson's standard knowledge sources. These questions constitute the vast majority of Jeopardy! questions and typically focus on entities and facts that are true about them in the world. Questions that require specialized techniques for producing candidate answers are described in [7]. Examples of these special question types are those in the "RHYME TIME" category, whose answers are typically made up of phrases of words that rhyme with each other. Excluding these special question types, our combined search and candidate generation strategies placed the correct answer in the candidate answer list for 87.17% of the questions in a 3,344-question unseen test set, facilitating high end-to-end QA performance.

Question analysis overview

In this section, we briefly discuss Watson's question analysis output leveraged by the search and candidate generation components. The output encapsulates syntactic and semantic analysis of the question, as described in [4, 6, 8], including dependency parse, semantic relations, LAT, and focus.

For parsing, we employ ESG (English Slot Grammar), a comprehensive deep Slot Grammar parser [8]. Each node in its dependency parse tree contains 1) a headword and its associated morpho-lexical, syntactic, and semantic features and 2) a list of "children" that are generally modifiers of the node, along with the slot that each modifier fills. The focus detection component identifies the question focus, which is the part of the question that refers to the answer and is often the head of the noun phrase with demonstrative determiners "this" or "these". The relation recognizer annotates certain types of unary and binary relations in the question; of particular interest here are semantic relations recognized over the focus whose predicates can be mapped to codified relationships in Watson's structured knowledge resources, such as `actorIn` and `authorOf`. Finally, the LAT detection module identifies the LAT of the correct answer to the question. The LAT is typically a term in the focus but may also include other nouns that are co-referential with the focus in the question and/or category.

Consider the following Jeopardy! question:

(1) MOVIE-"ING": Robert Redford and Paul Newman starred in this depression-era grifter flick. (Answer: "The Sting")

The focus identification component recognizes the focus to be "flick", the head of the noun phrase "this depression-era grifter flick". Since the focus "flick" does not have any co-referential terms in the question, the only LAT is the focus. Finally, the relation detection component recognizes two relation instances: `actorIn`(`Robert Redford`, *flick* : `focus`) and `actorIn`(`Paul Newman`, `flick : focus`). These recognized relations are used in the Answer Lookup component described below.

Search and candidate generation overview

The DeepQA architecture is designed to be a large-scale hypothesis generation, evidence-gathering, and scoring architecture [9]. **Figure 1** illustrates this architecture, focusing on how Watson's search and candidate generation components fit into the overall QA pipeline. The Hypothesis Generation phase takes as input results from question analysis, summarized in the previous section. The first four primary search components in the diagram show Watson's Document and Passage search strategies, which target unstructured knowledge resources such as encyclopedia documents and newswire articles [10]. On the other hand, the last two search components, namely, Answer Lookup and PRISMATIC search, use different types of structured resources. One or more candidate generation techniques are applied to each search result to produce plausible candidate answers. In our effort, we explored how to exploit the relationship between the title and content of title-oriented documents (such as encyclopedia articles; see below for more detail) and how to use metadata present in linked data (such as web documents) to help with effective candidate generation.

Searching unstructured resources

Watson's text corpora contain both title-oriented documents, such as encyclopedia documents, and non-title-oriented sources, such as newswire articles [10]. For title-oriented sources, the document title is typically an entity or a concept, and the content of the document provides salient information about that entity or concept. This relationship between document title and content inspired us to devise special search strategies to take advantage of the relationship for more effective search. We analyzed Jeopardy! question/answer pairs and the title-oriented documents that provide answers to the questions. We observed three possible relationships between the question/answer pair and those relevant documents.

In the first case, the correct answer is the title of the document that answers the question. For example, consider the Jeopardy! question "This country singer was imprisoned for robbery and in 1972 was pardoned by Ronald Reagan." The Wikipedia** article for Merle Haggard, the correct answer, mentions him as a country singer, his imprisonment for robbery, and his pardon by Reagan and is therefore an excellent match for the question. Jeopardy! questions, which often contain multiple facts about their answers, are frequently well-matched by these encyclopedia documents that cover most of the facts in the question. To exploit the strong association between the title and content for title-oriented documents, Watson

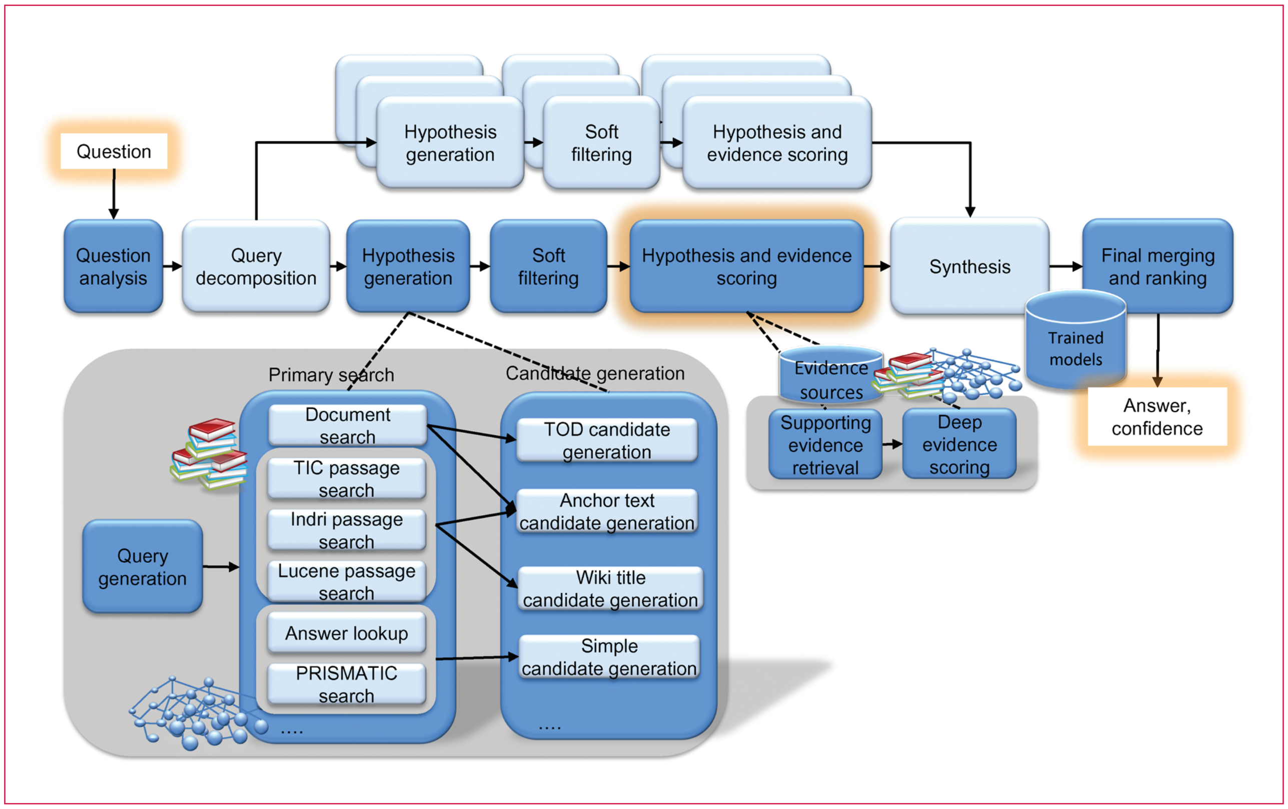

Figure 1

DeepQA architecture with details for search and candidate generation. (Modified and used, with permission, from D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, “Building Watson: an overview of the DeepQA project,” *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010; ©2010 Association for the Advancement of Artificial Intelligence.)

adopts a document-oriented search strategy to find documents that, as a whole, best match the question.

In the second case, the title of a document that answers the question is in the question itself. For instance, consider the question “Aleksander Kwasniewski became the president of this country in 1995.” The first sentence of the Wikipedia article on Aleksander Kwasniewski states, “Aleksander Kwasniewski is a Polish socialist politician who served as the President of Poland from 1995 to 2005.” Motivated by this observation, Watson adopts a passage search strategy against a subcorpus of documents, consisting of those documents whose titles appear in the question. We refer to this search strategy as a TIC (Title-in-Clue) Passage search.

In the third case, the answer-justifying document is a third-party document whose title is neither the answer nor in the question. We expect traditional passage search strategies adopted in existing QA systems [11] to be effective on those questions.

Although the three sets of search results have a fair degree of overlap, we have empirically shown that each strategy brings a unique contribution to the aggregated search results and thus helps improve the overall recall of our search process [12]. We discuss how we formulate search queries and how the three search strategies are performed in Watson in the remainder of this section.

Search query generation

Search queries for a question are generated from the results of question analysis. For all questions, a full query is generated on the basis of content terms or phrases extracted from the question, as well as any LAT detected in the category. Arguments of relations recognized over the focus are considered more salient query terms and are weighted higher. For instance, in Example (1) above, the following full query is shown, where the arguments of the `actorIn` relations are given empirically determined higher weights:

(2.0 “Robert Redford”) (2.0 “Paul Newman”) star depression era grifter (1.5 flick)

For those questions where the LAT has modifiers, as in the current example, a LAT-only query is generated in addition to the full query, based on the LAT and its modifiers. In this example, the full noun phrase that contains the LAT is "this depression-era grifter flick"; therefore, the LAT-only query contains the terms

depression era grifter flick

The motivation for the LAT-only query is that in some cases, the LAT and its modifiers uniquely identify the answer or narrow the candidate answer space to a small number of possibilities. This example falls into the latter case. Some LAT-only queries that identify a unique entity include *capital Ontario* and *first 20th century US president*. Whereas some LAT-only queries, such as *French composer* and *20th century playwright*, are too imprecise to be useful, we have found that, overall, these LAT-only queries are helpful in increasing system performance.

In the search strategies described below, the full query is applied to both Document search and Passage search. The LAT-only queries are applied to Passage search only because these queries tend to be short and are less effective for matching against full documents. The different queries and search strategies produce results that are aggregated for further processing.

In determining the sizes of the search hit lists, we attempted to strike a balance between increased candidate recall, processing time for all candidates, and the effect of the incorrect candidates on Watson's ability to rank the correct answer in top position. We empirically determined the hit list sizes reported below on the basis of experiments that measure hit list sizes against candidate recall and end-to-end system performance.

Document search

Watson's Document search component uses the open-source Indri search engine [13] and targets title-oriented documents that, as a whole, best match the question. Two separate Indri indices are used, one consisting of long documents such as encyclopedia articles and the other of short documents such as dictionary entries. The two separate indices are necessary because the significant size differences between the documents caused highly relevant short documents to be drowned out by longer documents when combined in one index. The full query, constructed as previously described, is used to retrieve the top 50 most relevant long documents and the top 5 most relevant short documents. A Document search rank and a search score are associated with each result, which are used as features for scoring candidate answers, in conjunction with additional features generated by other downstream answer scorers.

Passage search

To retrieve relevant passages from its unstructured knowledge resources, Watson extends common existing approaches [11] along two dimensions. First, we adopt the TIC Passage search strategy to target search against a small subset of highly relevant documents, i.e., those whose titles appear in the question. Second, we leverage multiple search engines in our implementation of traditional passage search to increase diversity and hence recall of the results. To this end, we adopt Indri's existing passage search capability and extend Lucene [14] to support passage retrieval. Indri and Lucene passage retrieval differ in two major aspects. The first concerns the retrieval model used, where Indri uses a combination of language modeling and inference network for ranking [15], and Lucene leverages tf (term frequency) and idf (inverse document frequency) for scoring relevance [16]. The second key difference is in the implementation of passage retrieval, which we discuss in the section on Lucene Passage search below.

Regardless of the strategy or search engine used, Watson's passage search components return one to two sentence passages scored to be most relevant to a given question. Watson retrieves ten passages each from TIC Passage search and Passage search. For the latter, five passages come from Indri and five from Lucene. Our empirical results show that an aggregation of five passages from each search engine achieves higher recall than retrieving ten passages using either search engine alone. The passage rank is used as a feature for scoring candidate answers extracted from that passage. The passage score feature is not used because the search scores returned by the two search engines are not comparable.

Indri Passage search

Indri supports passage retrieval through the use of the prefix #passage[X : Y] for a query. This prefix specifies that passages be evaluated within an X-word window, shifting Y words at a time, using a scoring algorithm analogous to that for document retrieval. In Watson's implementation, X is set to 20 and Y to 6, to balance the quality of results and speed. Watson's passage search component enhances Indri's native passage search results for a QA application in two ways. First, we extend each 20-word passage at both ends to sentence boundaries so that the results can be analyzed by our natural-language processing (NLP) components [4, 8]. Second, we augment Indri's native scoring algorithm to account for coverage of query terms, i.e., rewards are given to passages that cover a large portion of query terms over those that match a small fraction at high frequency. Our experiments showed that the rescoring process led to passages that are more likely to contain the answer to the question.

Title-in-Clue Passage search using Indri

Watson's TIC Passage search component uses characteristics of title-oriented documents to focus search on a subset of potentially more relevant documents. Its implementation makes use of Indri's support for dynamic subcorpus specification and a dictionary that maps canonical Wikipedia document titles[1] (e.g., Naomi) to all Wikipedia documents with that string as their title [e.g., Naomi (band) and Naomi (Bible), among others]. The dictionary is used to identify document titles in the question, and a subcorpus of documents is dynamically constructed by aggregating all target documents in the matched dictionary entries. Although the same query is used as in Indri Passage search, we found that by constraining the corpus to a much smaller and potentially more relevant subset, this search strategy can succeed when the general passage search strategy fails.

Lucene Passage search

Indri's approach to passage search is to treat each passage as a "mini-document" and to apply the same document-scoring algorithm to these passages. However, this approach did not fare well in Lucene. Instead, we observed that passages extracted from documents that are highly relevant to the question are more likely to contain the answer. We therefore introduced a two-phase passage retrieval process to Lucene. First, Lucene Document search is used to retrieve relevant documents. Second, passages are extracted from these documents and are ranked according to several criteria discussed below.

Our Lucene Document search component adopts a modification of Lucene's built-in similarity-scoring scheme, introduced in [17]. It primarily consists of improvements in term frequency normalization and in document length normalization. The search query is constructed from question keywords and phrases and uses Lucene's support for lexical affinities [18].

Our extension to Lucene for passage search consists of evaluating each single-sentence passage from top-scoring documents separately using a set of query-independent features and a relevance measure between the sentence and the query. We identified three query-independent features that affect the *a priori* probability of a sentence's relevance to a Jeopardy! question as follows.

- *Sentence offset*—Sentences that appear closer to the beginning of the document are more likely to be relevant; therefore, sentence offset is used as a scoring feature.
- *Sentence length*—Longer sentences are more likely to be relevant than shorter sentences; thus, sentence length is adopted as a feature.
- *Number of named entities*—Sentences containing more named entities are more likely to be relevant, since most Jeopardy! answers are named entities. We approximate the recognition of named entities in documents through occurrences of anchor texts and document titles in each sentence.

[1]Wikipedia document titles contain disambiguation information for nonprimary senses of ambiguous titles. For example, "Naomi (band)" is the title for the article describing the German electronic duo, and "Naomi (Bible)" is the title for the article about the biblical character Naomi.

For estimating the relevance of a passage for a given search query, given that the passage is extracted from a document relevant to the query, we found that a simple measure of keyword and phrase matching outperforms more sophisticated alternatives. This score is then multiplied by the search score of the document from which the passage is extracted.

The query-dependent similarity score is combined with the query-independent scores described above to determine the overall search score for each passage. In order to increase recall, each top-scoring single-sentence passage is expanded to include the sentence that precedes it in the document. This expansion is helpful because co-reference targets in the current sentence can often be found in the preceding sentence.

Searching structured resources

In addition to searching over unstructured resources, Watson also attempts to identify relevant content from structured syntactic and semantic resources. Watson adopts two primary approaches toward searching against two different types of structured resources, as shown in Figure 1. The first approach, Answer Lookup, targets existing knowledge sources encoding semantic relations, such as DBpedia [19]. The second approach, PRISMATIC search, uses a custom-built PRISMATIC knowledge base [20], which encodes syntactic and shallow semantic relations derived from a large collection of text documents.

Answer Lookup

Early QA systems took the approach of translating natural language into formal machine language, such as first-order logic or Bayesian logic. They then either looked up the answer from a structured knowledge base or performed reasoning on known facts to derive the answer. Most of Watson's QA capability does not depend on this approach, because the problem of translating natural language into a machine-understandable form has proven too difficult to do reliably. There are cases, however, where *parts* of a question can be translated into a machine-understandable form: for instance, when the question asks for a relation between the answer and a named entity in the question, such as the `actorIn` relations shown in Example (1). Watson turns that part of the question into a query against structured sources such as DBpedia [19] and the Internet Movie Database (IMDB) [21] in an attempt to instantiate the variable in each query ("flick" in the example). We call this capability Answer Lookup.

Effective Answer Lookup requires components that match entity names in questions to those in structured sources. More crucially, it directly depends on the quality and

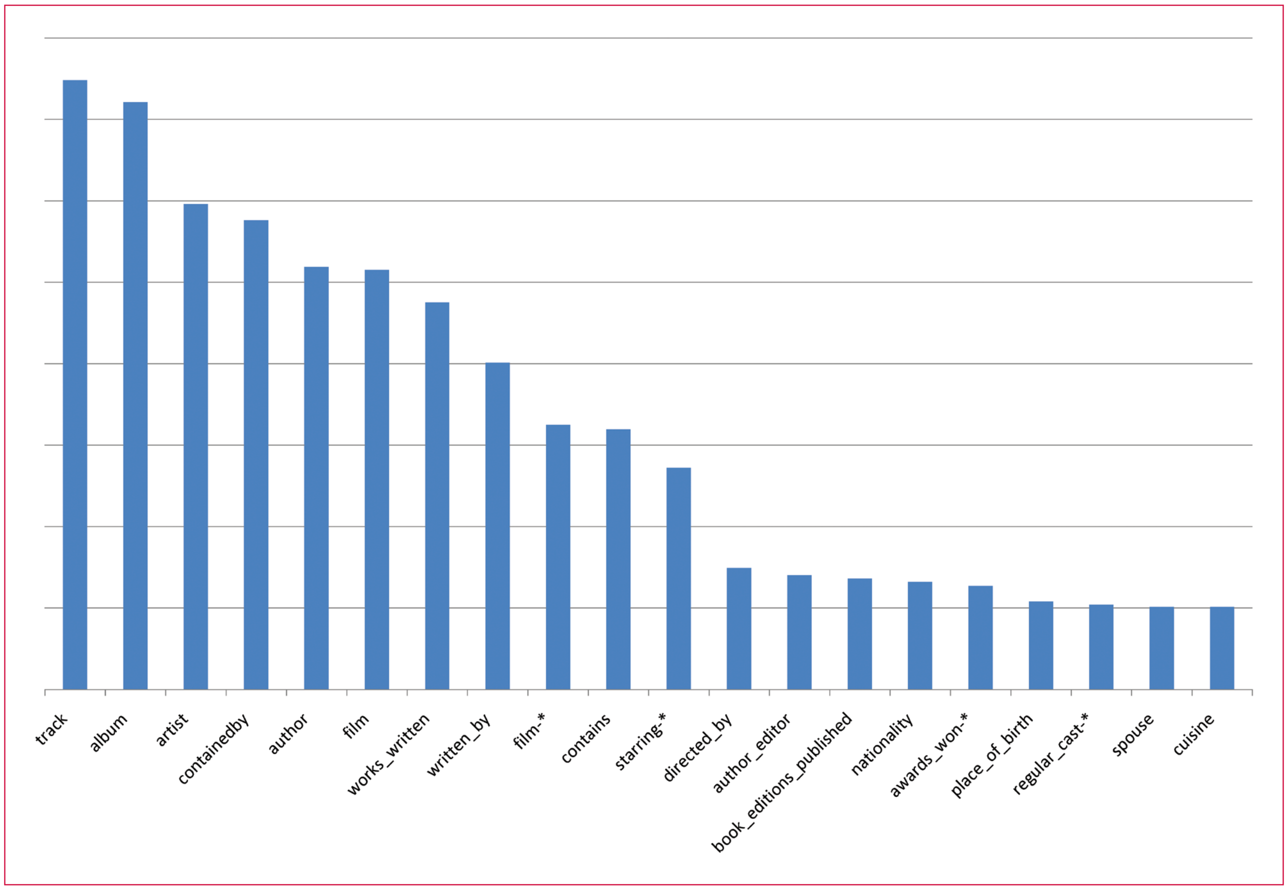

Figure 2

Approximate distribution of the 20 most frequent Freebase relations in 20,000 randomly selected Jeopardy! questions. (Modified and used, with permission, from D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: an overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010; ©2010 Association for the Advancement of Artificial Intelligence.)

quantity of semantic relations that can be identified in questions, and relation detection in text, while a problem studied for some time, remains an open issue. Broadly speaking, work in that area aims to develop recognition capabilities either manually or by statistical means.

For the DeepQA technology base, components have been developed using both recognition styles; these are described in [4]. The Watson system used only rule-based relation detection, largely because of performance constraints. For manual development of relation-bearing patterns, it is critical to analyze the domain—to identify the most frequently occurring relations in questions and to obtain appropriate structured sources that cover those relations.

Our approach, which is predicated on developing recognition grammars for each relation, entails an effort significant enough to drive us to focus only on the most frequent relations. However, the difficulty in automatic relation detection per se hampers our ability to automatically determine the most frequently occurring relations in a question set; in order to count them, one has to be able to detect them. As an approximation, we analyzed 20,000 randomly selected Jeopardy! questions with their correct answers against a few data sources we judged to match the Jeopardy! domain well. For each question, we looked for known relations in any of the chosen sources between a named entity in the question and the answer. The frequency of each relation in the sources linking a named entity in a question to an answer was aggregated over all 20,000 questions. Based on this approximation, **Figure 2** shows the relative histogram of most frequent Freebase [22] relations in these questions.

This ranked list of relations sets priorities for the implementation of relation recognizers in questions. In general, we are looking for English expressions of these known relations between a named entity and the question focus, such as `actorIn(Robert Redford, flick : focus)` and `actorIn(Paul Newman, flick : focus)`. The Answer Lookup component first tries to find the known named entity in the database from the relation argument string (e.g., "Robert Redford") using a step called entity disambiguation and matching (EDM) [2]. If a matching entity is found, a query is generated to find movies starring the given entity. As with this example, if multiple relations are found in the question, each separate relation is individually processed and its results pooled. An identical Answer Lookup score is assigned to each search result, except for those results reinforced by multiple search queries whose scores are boosted.

The top 20 relations in the Jeopardy! domain (see Figure 2) are predominately entertainment and geographical relations, which together appear in approximately 11% of our questions. The relation detection recall is approximately 60%, and its precision is approximately 80%. The EDM step has a recall of approximately 80%, and the data sources cover approximately 80% of the relations we need. Multiplying these probabilities together, we expect Answer Lookup to return the correct answer in approximately 4% of the questions. This estimate is validated by evaluation on a blind test set discussed later in this paper. The Answer Lookup score is discriminating only in a few cases, such as when multiple relation instances are detected and the intersection of the query results contains only one element, or when the relation instance detected has only one answer—such as when the question seeks the author of a book.

PRISMATIC search

PRISMATIC is a large-scale lexicalized relation resource automatically extracted from massive amounts of text [20]. PRISMATIC provides shallow semantic information derived from aggregating over syntactic or semantic relation usage in a large corpus, with generalization over entity types. For example, it gathers the aggregate statistics of relations extracted from syntactic parses, such as the frequency of the string *Tom Cruise* appearing as the subject of the verb *star* modified by a prepositional phrase "in ⟨movie⟩". The aggregate statistics can be used to infer selectional restrictions and other semantics.

One of the semantic relations PRISMATIC records is the occurrence of *isa* relations.[2] This information is particularly useful for search and candidate generation because it can identify the most popular instances of a particular LAT. For example, consider the question "Unlike most sea animals, in the Sea Horse this pair of sense organs can move independently of one another". Watson's search strategies described above failed to retrieve the correct answer as a candidate since the question mentions a relatively obscure fact. PRISMATIC search, on the other hand, focuses on the LAT and its modifiers, in this example *sense organ*, and identifies up to 20 most popular instances of the LAT with modifiers from the PRISMATIC knowledge base. The correct answer, *eyes*, is the third most popular instance of *sense organ* and is returned by the PRISMATIC search component. PRISMATIC search rank and score features are associated with each result for final scoring purposes, in conjunction with additional features generated by other downstream answer scorers.

Generating candidates from search results

Once relevant content is identified from Watson's knowledge sources, candidate answers are extracted from the content. For content retrieved from structured resources, the retrieved results (which are the uninstantiated arguments in the query) are the candidate answers. For unstructured results from Document search and Passage search, additional processing is required to identify plausible candidate answers from those search results. As a baseline, we adopted a named entity recognizer from our TREC (Text REtrieval Conference) QA system [23] and produced as candidate answers all entities it recognizes as an instance of any of the more than 200 types in its type system. This candidate set is a superset of what would be produced as candidates using a type-based candidate generation approach [24, 25] using the same type system. Even with the superset, we found that the type-based approach does not produce high enough candidate recall on Jeopardy! questions [12].

This section describes three general-purpose candidate generation techniques applied to unstructured search results, which improve upon our baseline: Title of Document candidate generation, applied to Document search results only; Wikipedia Title candidate generation, relevant for Passage search results only; and Anchor Text candidate generation, which is appropriate for both types of search results. These three candidate generation strategies are not answer type dependent, apply to the vast majority of questions, and generate most of Watson's candidate answers. The other candidate generation strategies, which we do not discuss in this paper, apply to a small number of questions or produce a small number of candidates only. Some of these strategies may be dependent on question type (e.g., questions seeking verb phrase answers or numeric answers), and others may rely on typographic cues.

Note that although the discussion below demonstrates how Watson uses Wikipedia metadata for candidate generation and the evaluation demonstrates how these strategies affect Watson's performance on Jeopardy!, we have previously demonstrated that the same techniques also effectively perform on questions from the TREC QA track [12]. Furthermore, the techniques we developed can be easily

[2]Subclass relationships between objects; for instance, a dog "is a" mammal.

applied to leverage metadata from other title-oriented documents (for Title of Document and Wikipedia Title candidate generation) and collections of documents with entity-oriented hyperlinks (for Anchor Text candidate generation).

Title of Document candidate generation

Recall that Document search identifies title-oriented documents that, as a whole, best match the facts presented in the question. For these search results, the entity that constitutes the title of a matched document fits the description of the question and is thus proposed as a candidate answer.

To facilitate the process of answer typing [2] in downstream processing, candidate provenance information is recorded for each candidate, if possible, in order to disambiguate the candidate string. In particular, some scorers for answer typing use structured resources, such as DBpedia, whose entries can be disambiguated via Wikipedia uniform resource identifiers (URIs). For example, the candidate Naomi with provenance *Naomi (Bible)* suggests that the candidate is a person, whereas the same candidate with provenance *Naomi (band)* will be typed as a musical group. For candidates generated from Wikipedia documents, the distinction between multiple meanings is available in the candidate generation phase as it corresponds to the document actually retrieved in the search process. This information is recorded in metadata associated with the candidate.

Wikipedia Title candidate generation

We conducted an experiment to evaluate the coverage of Wikipedia articles on Jeopardy! questions and found that the vast majority of Jeopardy! answers are titles of Wikipedia documents [10]. Of the roughly 5% of Jeopardy! answers that are not Wikipedia titles, some included multiple entities, each of which is a Wikipedia title, such as *Red, White, and Blue*, whereas others were sentences or verb phrases, such as *make a scarecrow* or *fold an American flag*. The high coverage of Wikipedia titles over Jeopardy! answers suggests that they can serve as an excellent resource for candidate generation.

The Wikipedia Title candidate generation strategy extracts from a retrieved passage all noun phrases that are Wikipedia document titles and are not subsumed by other titles. These indicate topics in the passage that are sufficiently salient to warrant their own Wikipedia page and, we hypothesize, are worth considering as candidate answers. To identify Wikipedia titles in passages, we use the same Wikipedia title dictionary used to identify document titles in questions for TIC Passage search. The target document titles [e.g., Naomi (Bible)] are used as provenance information for each candidate to help with disambiguation in downstream scoring.

Anchor Text candidate generation

Although Wikipedia Title candidate generation achieved high candidate recall for most passages, we hypothesized that linked metadata extracted from Wikipedia documents can be used to improve the precision of candidates extracted from Wikipedia passages. For example, consider the passage "*Neapolitan pizzas* are made with ingredients like *San Marzano tomatoes*, which grow on the volcanic plains south of *Mount Vesuvius* and *Mozzarella di Bufala Campana*, made with milk from *water buffalo* raised in the marshlands of *Campania* and *Lazio*." Whatever the question for which the passage was retrieved, we expect terms or phrases such as "Neapolitan pizza", "Mount Vesuvius", and "water buffalo" to be plausible candidates and other terms such as "grow" and "ingredients" to be less likely as candidate answers. We observed that plausible candidates typically satisfy two criteria. First, they represent salient concepts in the passage. Second, the candidates have Wikipedia articles about them.

We observed that Wikipedia contains several types of metadata that, in aggregation, make up a set of plausible candidates that represent salient concepts for each document [12]. They include the following:

1. Anchor texts in the document.
2. Document titles of hyperlink targets (which are often synonyms of terms in 1).
3. Title of the current document.
4. Titles of redirect pages to the current document (which are often synonyms of 3).

The metadata is aggregated on a per-document basis, and they become plausible candidates for search results from that document. In other words, for each search result from a document, terms or phrases in the plausible candidate set for that document that appear in the search result are generated as candidate answers. In the above sample passage, the italicized terms satisfy one of the four criteria above and represent the candidate answers that would be generated from that passage. In Watson, Anchor Text candidate generation is applied to Wikipedia document titles from Document search as well as all passages retrieved from Wikipedia (in lieu of Wikipedia Title candidate generation, which has nearly identical candidate recall but significantly lower precision [12]).

For candidate provenance, we take advantage of the linked nature of these candidate answers to accurately identify the senses for candidates that can potentially be ambiguous. For candidate answers extracted from Anchor Text-based resources (1 and 2 in the list above), the target document is used to identify the sense of the candidate. For those extracted from document title or redirects [(3)

TABLE 1 Search and candidate generation evaluation results.

	Document search	Passage search	Answer Lookup	PRISMATIC search	All
Percentage of questions in active subset	99.07%	100.00%	13.64%	43.75%	100%
No. of answers/active question	90.55	162.47	15.11	11.57	216.53
Binary recall	74.43%	79.40%	3.53%	8.31%	87.17%
Percentage unique	7.24%	12.05%	0.03%	0.18%	—
Accuracy	62.65%	62.74%	2.18%	5.65%	71.32%

and (4) in the list above], the current document is used to disambiguate among multiple senses.

Experimental evaluation

Experimental setup

To evaluate the impact of the search and candidate generation strategies described in this paper, we randomly selected 66 previously unseen Jeopardy! games. We excluded from these games special questions that require a tailored process for candidate generation, such as puzzle questions [7] and common bond questions [26], resulting in a test set of 3,344 questions. We evaluate the coverage of each search strategy paired with its corresponding candidate answer generation components and measure performance using the candidate binary recall metric for each strategy separately as well as all strategies combined. Candidate binary recall is computed as the percentage of questions for which the correct answer is generated as a candidate answer. We adopt this evaluation metric to reflect our goal of maximizing candidate recall at the Hypothesis Generation phase of the DeepQA pipeline.

Results and discussion

The results of our experiments are shown in **Table 1**. For each search and candidate generation method, we computed the number of questions for which at least one result was returned, henceforth referred to as the *active subset*. For each approach, we also computed the average number of candidates generated per question in the active subset and the candidate binary recall computed over all questions.

Our results show that Document search and Passage search have very high coverage, returning candidates for all questions for Passage search and all but 1% of the questions for Document search.[3] Although they both yield high candidate binary recall, that is, 74.43% and 79.40%, respectively, they also generate a fairly large number of candidates per question. Contrast that with Answer Lookup and PRISMATIC search, which are active on a much smaller set of questions and correspondingly have much lower overall binary recall. However, for these search strategies, only a small number of candidate answers are added to the pool. Note that Answer Lookup yielded a candidate binary recall of 3.53% on blind data, which is close to the 4% estimate in an earlier discussion.

The row labeled "Percentage unique" in Table 1 examines the unique contribution of each search and candidate generation approach, i.e., the loss in candidate binary recall if that strategy is ablated from the system. Our results show that although the degree of overlap is quite high among the different methods, each approach does make unique contributions to achieve an 87.17% combined candidate binary recall. We analyzed the 429 questions with candidate recall failures and found that roughly three-fourths of them are due to search failures. These questions typically fail because the question contains extraneous information that is relevant but not necessary for identifying the answer. For example, in "Star chef Mario Batali lays on the **lardo, which comes from the back of this animal's neck**", the essential part of the question is the segment in bold. However, the prominent entity "Mario Batali" steered search in the wrong direction and dominated our search results. For the other one-fourth, search returned a relevant passage, but our candidate generation strategies failed to extract the answer as a plausible candidate. In most of these examples, the correct answers are common nouns or verbs, which generally have lower coverage for both our Anchor Text and Wikipedia Title candidate generation strategies.

Finally, we examined the contribution of each search and candidate generation strategy on end-to-end QA performance. The last row in the table shows the QA accuracy when the candidates generated by each approach are scored by Watson's full suite of answer scorers and are ranked. Our results show that Document search and Passage search achieve very comparable performance, about 9%

[3]The 1% of questions in the inactive subset for the Document search pipeline are questions for which a numeric answer is sought. For these questions, a specialized number candidate generation component is employed.

lower than full system performance. We ran an additional experiment in which only the full search query is issued to Indri Passage search to retrieve the ten most relevant passages. This is the search configuration closest to the search strategies adopted in many existing QA systems. When these candidates are scored and ranked, the system achieves an accuracy of 54.9%. These experimental results demonstrate the effectiveness of our search and candidate generation strategies, which, in aggregation, achieve an accuracy of 71.32%.

Related work

From the earliest days of artificial intelligence and NLP, the prevalent vision was that machines would answer questions by first "translating" human language into a machine representation and then matching that representation against background knowledge using a formal reasoning process [27, 28]. To date, however, no one has successfully produced such formal logical representations from unseen natural-language questions in a reliable way.

The first QA system to use structured data sources effectively for candidate generation was the START system [29], whose roots date back at least to 2000 [30]. Similar to our Answer Lookup approach, a retrieval-based QA system was augmented with a capability to recognize relations in questions, and structured sources were queried with the detected relation and question focus. As reported here, the gating factor in exploiting this for QA is the ability to detect the relations in the question.

In the Halo and Aura systems [31], closed-domain questions (on, e.g., chemistry) are answered using structured sources containing many axioms about processes and facts in these domains. Aura addresses classes of questions that are out of scope for Watson, having mainly to do with problem solving or procedural questions, such as "How many ml of oxygen is required to produce 50 ml of water?" However, the natural-language capability of these systems is constrained by question templates that the system can map to underlying structured queries.

Most existing open-domain retrieval-based QA systems adopt a pipeline of passage search against a reference corpus and generation of candidates of the expected answer type from the search results. From the search perspective, some systems have explored the use of web data for the purposes of both generating new candidate answers [32–34] and validating existing candidate answers [35, 36]. Furthermore, online encyclopedias such as Grolier** and Wikipedia have been used as corpora for several QA systems [37, 38] and in the CLEF (Cross Language Evaluation Forum) evaluation effort [39]. However, to our knowledge, these systems treated the new corpus as an extension of the newswire corpus used in earlier organized QA evaluation efforts and did not exploit its inherent characteristics to improve QA performance. In contrast, we analyzed the association between title-oriented encyclopedic documents and question/answer pairs to motivate two additional search strategies for QA: Document search and TIC Passage search. These search techniques are effective for title-oriented documents and have been shown to improve upon the results of traditional passage search alone.

From the candidate generation perspective, the vast majority of existing QA systems adopt a semantic-type-based approach to produce candidates that match the expected answer type on the basis of a static predefined type ontology [24, 25]. In contrast, Watson does not rely on such an ontology but utilizes document metadata, such as document title, anchor texts, and titles of hyperlink target documents, to associate salient concepts with respect to each document and thus to create a pool of plausible candidate answers. Although this approach generates a substantially larger set of candidate answers, we have found it to significantly outperform type-based methods in a broad-domain task such as Jeopardy! [12].

Conclusion

A crucial step in achieving high QA performance is to cast a wide enough net in the Hypothesis Generation phase to include the correct answer in the candidate pool. In this paper, we described Watson's multipronged strategies for identifying relevant content and producing candidate answers that balance high candidate recall and processing time for candidate scoring.

In its Hypothesis Generation phase, Watson uses the results of question analysis to formulate effective queries to identify relevant content from both structured and unstructured resources. For search against textual resources, we extended the common passage search paradigm adopted by most existing QA systems in two ways. First, to take advantage of the relationship between the title and content of title-oriented documents, we adopted a three-pronged search strategy: Document search, TIC Passage search, and Passage search. Second, to increase diversity of search results, we employed two search engines in our Passage search component. We have empirically found that both extensions lead to higher candidate recall. For search against structured resources, Watson employs two strategies: Answer Lookup and PRISMATIC search. Answer Lookup relies on recognition of high-frequency semantic relations involving the focus and retrieves possible instantiations of the focus from existing structured knowledge sources such as DBpedia. PRISMATIC search focuses on *isa* relations mined from large corpora to produce salient instances of the LAT plus modifiers.

For candidate generation from unstructured search results, Watson does not rely on a predefined type ontology because of the broad range of answer types observed in Jeopardy! questions. To identify plausible candidates, Watson uses document metadata such as document titles, anchor texts, and

Wikipedia redirects to identify salient concepts associated with each document. These metadata are used for three candidate generation strategies: Title of Document candidate generation, applied to Document search results; Wikipedia Title candidate generation, applied to Passage search results; and Anchor Text candidate generation, applied to document titles and passages from Wikipedia documents. Whenever possible, candidate answers carry metadata that encode provenance information to help downstream scorers disambiguate the word sense of the candidate.

Evaluation on a blind set of more than 3,000 questions shows that the different search/candidate generation strategy pairs have different performance characteristics. Those targeting unstructured resources generate a larger number of candidates per question at high recall, whereas those focusing on structured resources produce far fewer candidates per question and have a much smaller active set of questions. Overall, all strategies make unique positive contributions, yielding a combined 87.17% in candidate binary recall.

*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, or Grolier Incorporated in the United States, other countries, or both.

References

1. 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.
2. 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.
3. 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, and Z. M. Qiu, "Structured data and inference in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 10, pp. 10:1–10:14, May/Jul. 2012.
4. 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.
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. 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.
7. J. M. Prager, E. W. Brown, and J. Chu-Carroll, "Special questions and techniques," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 11, pp. 11:1–11:13, May/Jul. 2012.
8. 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.
9. 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.
10. J. Chu-Carroll, J. Fan, N. Schlaefer, and W. Zadrozny, "Textual resource acquisition and engineering," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 4, pp. 4:1–4:11, May/Jul. 2012.
11. S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proc. 26th ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2003, pp. 41–47.
12. J. Chu-Carroll and J. Fan, "Leveraging Wikipedia characteristics for search and candidate generation in question answering," in *Proc. 25th Conf. AAAI*, 2011, pp. 872–877.
13. Indri Search Engine. [Online]. Available: http://www.lemurproject.org/indri.php
14. Apache Lucene. [Online]. Available: http://lucene.apache.org
15. D. Metzler and W. B. Croft, "Combining the language model and inference network approaches to retrieval," *Inform. Process. Manage. Special Issue Bayesian Netw. Inform. Retrieval*, vol. 40, no. 5, pp. 735–750, Sep. 2004.
16. E. Hatcher and O. Gospodnetic, *Lucene in Action*. Greenwich, CT: Manning Publ. Co., 2004.
17. D. Cohen, E. Amitay, and D. Carmel, "Lucene and Juru at TREC 2007: 1-Million queries track," presented at the Text REtrieval Conf., Gaithersburg, MD, 2007, Special Publ. 500-274.
18. D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer, "Automatic query refinement using lexical affinities with maximal information gain," in *Proc. 25th ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2002, pp. 283–290.
19. DBPedia. [Online]. Available: http://www.dbpedia.org
20. 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.
21. Internet Movie Database. [Online]. Available: http://www.imdb.com
22. FreeBase. [Online]. Available: http://www.freebase.com/
23. J. Prager, J. Chu-Carroll, E. Brown, and K. Czuba, "Question answering using predictive annotation," in *Advances in Open-Domain Question Answering*, T. Strzalkowski and S. Harabagiu, Eds. Norwell, MA: Kluwer, 2006, pt. 4, pp. 307–347.
24. J. Prager, E. Brown, A. Coden, and D. Radev, "Question answering by predictive annotation," in *Proc. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2000, pp. 184–191.
25. D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus, "The structure and performance of an open-domain question answering system," in *Proc. 38th Annu. Meeting Assoc. Comput. Linguistics*, 2000, pp. 563–570.
26. J. Chu-Carroll, E. W. Brown, A. Lally, and J. W. Murdock, "Identifying implicit relationships," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 12, pp. 12:1–12:10, May/Jul. 2012.
27. E. Rich, *Artificial Intelligence*. New York: McGraw-Hill, 1991.
28. R. V. Guha and D. B. Lenat, "CYC: A mid-term report," *AI Mag.*, vol. 11, no. 3, pp. 32–59, 1990.
29. B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner, and A. Wilcox, "External knowledge sources for question answering," presented at the 14th Annual Text REtrieval Conf. (TREC), Gaithersburg, MD, 2005. [Online]. Available: http://www.mendeley.com/research/external-knowledge-sources-question-answering/
30. B. Katz and J. Lin, "REXTOR: A system for generating relations from natural language," in *Proc. ACL 2000 Workshop NLP&IR*, 2000, pp. 67–77.
31. D. Gunning, V. Chaudhri, P. Clark, K. Barker, S. Chaw, M. Greaves, B. Grosof, A. Leung, D. McDonald, S. Mishra, J. Pacheco, B. Porter, A. Spaulding, D. Tecuci, and J. Tien. (2010). Project Halo update—Progress toward digital Aristotle. *AI Mag.* [Online]. *31(3)*. Available: http://www.cs.utexas.edu/users/pclark/papers/AURA-AIMag.pdf
32. C. Clark, G. Cormack, T. Lynam, C. Li, and G. McLearn, "Web reinforced question answering (MultiText Experiments for TREC 2001)," in *Proc. Text REtrieval Conf.*, 2001, pp. 673–679. [Online]. Available: http://trec.nist.gov/pubs/trec10/papers/mtA.pdf
33. S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng, "Web question answering: Is more always better?" in *Proc. 25th ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2002, pp. 291–298.

34. B. Katz, J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora, "Integrating web-based and corpus-based techniques for question answering," in *Proc. 12th Text REtrieval Conf.*, 2003, pp. 426–435.
35. B. Magnini, M. Negri, R. Prevete, and H. Tanev, "Is it the right answer?: Exploiting web redundancy for answer validation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 425–432.
36. J. Ko, L. Si, and E. Nyberg, "A probabilistic framework for answer selection in question answering," in *Proc. NAACL-HLT*, 2007, pp. 524–531.
37. J. Kupiec, "MURAX: A robust linguistic approach for question answering using an online encyclopedia," in *Proc. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 1993, pp. 181–190.
38. D. Ahn, V. Jijkoun, G. Mishne, K. Muller, M. de Rijke, and S. Schlobach, "Using Wikipedia at the TREC QA track," in *Proc. Text REtrieval Conf.*, 2004. [Online]. Available: http://www.mendeley.com/research/using-wikipedia-in-the-trec-qa-track/
39. D. Giampiccolo, P. Froner, A. Penas, C. Ayaache, D. Cristea, V. Jijkoun, P. Osenova, P. Rocha, B. Sacaleanu, and R. Suteliffe, "Overview of the CLEF 2007 multilingual QA track," in *Advances in Multilingual and Multimodal Information Retrieval*, C. Peters, V. Jijkoun, T. Mandl, M. Henning, D. W. Oard, P. Anselmo, V. Petras, and D. Santos, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 200–236.

Received July 26, 2011; accepted for publication January 9, 2012

Jennifer Chu-Carroll *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (jencc@us.ibm.com).* Dr. Chu-Carroll is a Research Staff Member in the Semantic Analysis and Integration Department at the T. J. Watson Research Center. She received the Ph.D. degree in computer science from the University of Delaware in 1996. Prior to joining IBM in 2001, she spent 5 years as a Member of Technical Staff at Lucent Technologies Bell Laboratories. Dr. Chu-Carroll's research interests are in the area of natural-language processing, more specifically in question-answering and dialogue systems. Dr. Chu-Carroll serves on numerous technical committees, including as program committee co-chair of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT) 2006 and as general chair of NAACL HLT 2012.

James Fan *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (fanj@us.ibm.com).* Dr. Fan is a Research Staff Member in the Semantic Analysis and Integration Department at the T. J. Watson Research Center, Yorktown Heights, NY. He joined IBM after receiving the Ph.D. degree at the University of Texas at Austin in 2006. He is a member of the DeepQA Team that developed the Watson question-answering system, which defeated the two best human players on the quiz show *Jeopardy!*. Dr. Fan is author or coauthor of dozens of technical papers on subjects of knowledge representation, reasoning, natural-language processing, and machine learning. He is a member of Association for Computational Linguistics.

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 Thomas J. Watson Research Center. He received the Engineering degree in electronics from the Higher Institute for Mechanical and Electrical Engineering, Sofia, Bulgaria, in 1974 and the Diploma and Ph.D. degrees in computer science, in 1976 and Computational Linguistics, in 1980, respectively, from the University of Cambridge, Cambridge, U.K.. 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.

David Carmel *IBM Research Division, Haifa Research Lab, Haifa, Israel, 31905 (carmel@il.ibm.com).* Dr. Carmel is a Research Staff Member in the Information Retrieval Group at the IBM Haifa Research Lab. His research is focused on search in the enterprise, query performance prediction, social search, and text mining. For several years, he taught the introduction to information retrieval course in the Computer Science Department at Haifa University, Haifa, Israel. At IBM, Dr. Carmel is a key contributor to IBM enterprise search offerings. He is a cofounder of the Juru search engine, which provides integrated search capabilities for several IBM products and was used as a search platform for several studies in the TREC conferences. He has published more than 80 papers in information retrieval and web journals and conferences, and he serves on the editorial board of the *Information Retrieval* journal and as a senior program committee member or an area chair of many conferences (Special Interest Group on Information Retrieval [SIGIR], WWW, Web Search and Data Mining [WSDM], and Conference on Information and Knowledge Management [CIKM]). He organized a number of workshops and taught several tutorials at SIGIR and WWW. He is coauthor of the book *Estimating the Query Difficulty for Information Retrieval*, published by Morgan & Claypool in 2010, and he is the coauthor of the paper "Learning to Estimate Query Difficulty," which won the Best Paper Award at SIGIR 2005. He earned his Ph.D. degree in computer science from the Technion, Israel Institute of Technology in 1997.

Dafna Sheinwald *IBM Research Division, Haifa Research Lab, Haifa, Israel, 31905 (dafna@il.ibm.com).* Dr. Sheinwald is a Research Staff Member in the Information Retrieval Group in the IBM Haifa Research Lab. She received the D.Sc. degree from the Technion-Israel Institute of Technology, Haifa, and subsequently joined IBM at the Haifa Research Lab, working for two years, at the IBM Almaden Research Center, San Jose, CA. She has worked on data compression for computer systems and in the recent years on information retrieval in the enterprise, directly contributing to several related IBM products. She has taught the introductory course to information theory at the computer science department of Haifa University, and she has served on the program committee of the annual IEEE Data Compression Conference and the committee of the IEEE Conference on Software Science, Technology, and Engineering (SWSTE).

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 the Ph.D. degree in computer science from Rensselaer Polytechnic Institute, Troy, NY, in 1995. He joined IBM in 2002, after spending 6 years as a professor at Vassar College, Poughkeepsie, NY, 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*.