Textual evidence gathering and analysis

J. W. Murdock
J. Fan
A. Lally
H. Shima
B. K. Boguraev

One useful source of evidence for evaluating a candidate answer to a question is a passage that contains the candidate answer and is relevant to the question. In the DeepQA pipeline, we retrieve passages using a novel technique that we call ***Supporting Evidence Retrieval****, in which we perform separate search queries for each candidate answer, in parallel, and include the candidate answer as part of the query. We then score these passages using an assortment of algorithms that use different aspects and relationships of the terms in the question and passage. We provide evidence that our mechanisms for obtaining and scoring passages have a substantial impact on the ability of our question-answering system to answer questions and judge the confidence of the answers.*

Introduction

After analyzing a question [1] and generating a set of candidate answers [2], the DeepQA pipeline employs an extensible set of evidence-scoring components to quantify evidence relating to each of the candidate hypotheses. In this paper, we focus on a specific subset of those components: *passage-scoring components*. Passage-scoring components use text passages that contain a candidate answer and try to relate those passages back to the question text. The passages are retrieved using a novel technique that we call *Supporting Evidence Retrieval* (SER), which inserts the candidate answer back into the original question to form a proposition; it then uses the DeepQA search techniques [2] to find passages most closely related to that proposition. Each passage-scoring component provides separate scores for each candidate answer in each passage found for it. These scores (along with scores from other evidence-scoring components) are weighed and combined using a statistical model during answer merging and ranking [3]. This way, DeepQA computes the likelihood of each hypothesis being correct.

We present the following four passage-scoring algorithms.

- *Passage Term Match*—This assigns a score by matching question terms to passage terms, regardless of grammatical relationship or word order.
- *Skip-Bigram*—This assigns a score by matching pairs of terms that are connected or nearly connected (connected skipping one other node) in the structure of the question to corresponding pairs of terms in the passage.
- *Textual Alignment*—This assigns a score by comparing the words and word order of the passage to those of the question with the focus replaced by the candidate answer.
- *Logical Form Answer Candidate Scorer (LFACS)*—This assigns a score on the basis of how well the structure of the question matches with that of the passage, aligning the focus to the candidate answer.

These algorithms are applied to passages obtained from both primary search [2] and SER. Applying a variety of scoring algorithms to a variety of passages for each answer provides a powerful combination of capabilities for identifying correct answers.

The remainder of this paper begins with our research hypotheses. It then describes the precursors to the passage-scoring algorithms: the SER mechanism that obtains passages and the graph encoding of the content in the questions and the retrieved passages. Next, we describe the four passage-scoring algorithms listed above. Finally, we provide evaluation results, discuss related work, and draw conclusions.

Hypotheses

We evaluated the following two hypotheses.

Hypothesis 1—Analyzing passages using a variety of strategies, at different depths of analysis, is significantly more effective than a single scoring strategy. Specifically,

Digital Object Identifier: 10.1147/JRD.2012.2187249

0018-8646/12/$5.00 © 2012 IBM

we employ the following four techniques described in the introduction of this paper and in more detail below: Passage Term Match, Skip-Bigram, Textual Alignment, and Logical Form. We show that this combination provides an effective set of answer-scoring mechanisms.

Hypothesis 2—SER improves the effectiveness of passage scoring by providing more passages for each answer than are found in primary search alone.

Answer-scoring techniques fail when the initial passage retrieval does not return passages that are amendable for the scoring techniques. By performing additional searches that include the candidate answer in the query, SER finds more passages that express in different ways the key relationship between the candidate answer and the question, even if those passages are not very good matches for some of the other question terms. Some of these passages may be phrased in a way that is more appropriate for the answer-scoring techniques. Furthermore, this takes advantage of the redundancy in a large corpus since we are likely to find more high-scoring passages for the correct answer than for an incorrect answer.

Supporting Evidence Retrieval

Our implementation of SER is very similar to the Indri passage-retrieval algorithm used for finding and generating candidates [2]. This algorithm builds a search query using terms from the question; it does not require any of the terms and gives more credit to passages that match more of them. SER augments this algorithm by adding in the candidate answer string as a required term. We use the custom passage-ranking algorithm described in [2]. The 20 highest ranked passages obtained from SER are sent to the scoring algorithms.

For example, consider the following Jeopardy!** question:

> "In 1840 this German romantic married Clara Wieck, an outstanding pianist and a composer, too."

The DeepQA passage-retrieval query [2] returns a few passages containing the correct answer, "Robert Schumann", causing it to be considered as a candidate answer. None of those passages is highly scored by our answer scorers. Here is one of those passages:

> "Clara Wieck Schumann: a German musician, one of the leading pianists of the Romantic era, as well as a composer, and wife of composer Robert Schumann."

In order to find a good match between this passage and the question, our passage scorers would need to know that "wife of" (with "Clara Wieck Schumann" coreferential with "wife") matches "married". This can be accomplished through semantic relation detection, but we do not have coverage for every semantic relation that Jeopardy! questions ask about.

SER forms a query that includes the required phrase "Robert Schumann" along with other (optional) terms from the question, such as "1840", "German", "romantic", "pianist", "*Clara Wieck*", and "composer". This query returns many more passages, such as

> "Although Robert Schumann made some 'symphonic attempts' in the autumn of 1840, soon after he married his beloved Clara Wieck, he did not compose the symphony until early 1841."

This passage does not contain many of the original query terms such as "German", "romantic", or "pianist", and so was highly ranked only after we included "Robert Schumann" in the query. Since it uses the verb "married" with subject "Robert Schumann" (after applying some anaphora resolution) and object "Clara Wieck", it can be scored well by Logical Form using syntactic relations alone (see below for more details).

Syntactic–semantic graphs

Two of the passage scorers (Skip-Bigram and LFACS) are designed to use structural characteristics of questions and passages and, therefore, operate on syntactic–semantic graphs. In these graphs, the nodes are terms in the clue (e.g., a word or a proper name), and the edges encode syntactic and/or semantic relations among those terms. Such graphs are hybrid, in which they combine the depth and power of deep semantic relationship detection with the breadth and generality of syntactic analysis.

The syntactic portions of the graph are derived from a predicate-argument structure (PAS) abstracted from an English Slot Grammar (ESG) parse [4]. The semantic portions of the graph are derived by semantic relation detectors [5]. Both the question and the passage text are mapped to their graph representation using the same mapping procedure. Some of the DeepQA relation patterns were specifically motivated by challenges in passage scoring; those are described in more detail below.

Matching with syntactic–semantic graphs

The most common source of failure in passage scoring is differences in the way things are expressed in text and the way they are expressed in passages. For example, consider the following question and passage:

> "Who authored 'The Good Earth'?"

> "Pearl Buck, author of 'The Good Earth'"

The question would have a syntactic graph derived directly from the subject and object arguments to its main verb, with edges such as (*author-subj-who*) and (*author-obj-The Good Earth*). We use the notation "(⟨term⟩-⟨edge-label⟩-⟨term⟩)" to indicate the labeled edge of the graph connecting two text terms; thus, (*author-subj-who*) represents the `subj` relation between the two nodes underlying "Who authored". None of these edges is a good match for any of the edges in a purely syntactic graph of the passage (which has no verb and, thus, no subject or object for that verb).

One way to address this challenge is to abstract, from the syntax, a more semantically rich graph; this can allow a question to match a wider variety of semantically equivalent passages. For example, we could encode the question as a single-edged graph, i.e., (*who-authorOf-The Good Earth*), appealing to a semantic relation in a fixed ontology, i.e., *authorOf*. The passage "Pearl Buck, author of 'The Good Earth'" could also be encoded as (*Pearl Buck-authorOf-The Good Earth*). Such abstraction would license aligning the two subgraphs that now have an identical edge.

As described in later sections, Skip-Bigram and LFACS both benefit from aligned edges in the structure of the question and passage. LFACS is particularly dependent on edges such as this because it requires that edge labels match and does not skip over nodes. When semantic relationships such as `authorOf` are detected, Skip-Bigram and LFACS can use those edges. When they are not, these scorers must rely on syntactic edges. The hybrid nature of syntactic–semantic supports both matching regimes.

Relation detectors can find multiple useful relations in a single question. Consider aligning the following question–passage pair:

> "Ambrose Bierce penned this sardonic reference work in 1906."
>
> "One of Ambrose Bierce's most famous works is his much-quoted book, *The Devil's Dictionary*."

Question analysis detects a relation that can be rendered as a subgraph (*Ambrose Bierce-authorOf-work:focus*); passage analysis detects two relations, namely, (*Ambrose Bierce-authorOf-book*) and a (*The Devil's Dictionary-instanceOf-book*). Chaining these two subgraphs across the shared (*book*) node gives LFACS and Skip-Bigram enough information to successfully align the unbound (focus) variable, "this ... work", in the `authorOf` relation on the question side with "The Devil's Dictionary" in the passage, thus acquiring supporting evidence for this candidate answer.

Semantic relations for matching

We detect two different classes of semantic relations. Deep semantic relations connect strongly typed ontological entities, via ontological relationships (e.g., those between persons and works of art, exemplified by `authorOf`). The motivation for, nature, and choice of deep semantic relations used in DeepQA are discussed in [5]. Shallow semantic relations detect alternative syntactic contexts expressing the same (even if unnamed) semantic relationship between two or more entities. Such shallow relations can be also used for passage alignment. Our motivation for developing them comes from the fact that a semantic relationship between entities may be expressed, systematically, in a variety of surface forms, some of which are predictable as a transformation over a (local) syntactic tree.

Consider, for example, noun–noun modification. While compound nouns present the notoriously difficult problem of interpreting the semantic relation implicit in the compounding, a syntactic transform via a linking prepositional phrase is usually meaning preserving: "this Wyoming senator" will match "a senator from Wyoming". We detect, for both forms, the same relation, i.e., (*senator-relatedTo-Wyoming*); note that although no deep semantic label is assigned, subgraph matching in LFACS will be triggered via the identically labeled edges. Thus, shallow semantic analysis abstracts away the syntactic differences between the noun compounding and noun–prepositional phrase postmodification. A similar kind of analysis underscores the equivalence of "five of the senators" and "five senators", "Monroe's presidency" and "the Monroe presidency", and others.

Other examples of shallow semantic relations include appositive constructions, lexical coreference, lexical typing, geopolitical characterization, and temporal stamping. In general, shallow semantic relations, by capturing alternate variants of common expressions, encode intuitions that other work refers to as natural-language axioms (see [6]; we extend this work, at least in coverage).

The shallow semantics inventory (derived from an intensive data analysis) thus systematically encodes numerous patterns to detect alternative surface formulations of relations such as `relatedTo`, `instanceOf`, `subtypeOf`, `sameAs`, `coIndex`, `tLink`, and `gpAssociation`, i.e., the last two referring to temporal links (typically between a date and an event), and geopolitical association expressions such as "Alaska's capital" and "this Alaskan capital". The LFACS component is conditioned to match subgraphs anchored on such labels.

Shallow semantic analysis for structure matching also incorporates the broad set of verb alternation patterns (for English) [7]. These are, in effect, templates that identify alternative syntactic realizations of a predicate-argument's cluster, without change in meaning. As such, they naturally fall within the range of structural transformations that can benefit LFACS and Skip-Bigram.

As an example, consider the common case of the ergativity paradigm, which allows for (surface) subject/object interchangeability. In the following question–passage pair, we would want "material" on the question side to align with "cartilages":

> "The meniscus disks in your knee are a type of this **material** that may **tear** if you twist wrong" to align with "cartilage"
>
> "... it is common to **tear** one or more ligaments or **cartilages**"

On the basis of syntax alone, alignment would fail as the two terms are in different argument positions to the verb "tear": The question focus ("material") is a subject, whereas the passage has the answer ("cartilage") as object to "tear". Shallow semantic relations detect syntactic configurations governed by alternations-licensing verbs and, for both sides of the pattern template, postsemantic roles (in the case above, e.g., *agent* and *theme* are relation labels superimposed on the arguments of "tear"). Our analysis exploits VerbNet [8], augmented with local additions to its verb paradigm classes.

Passage Term Match

Passage Term Match provides a measure of how often a candidate answer appears in the same passages as the question terms. It is useful for two reasons. First, passages that support the answer to a question may do so in a way that uses the same words but in a radically different order and with a dramatically different syntactic structure. Second, passages that do not specifically answer the question but just reflect some co-occurrence may be still correlated with the answer being correct. For example, a passage that states that Robert Schumann and Clara Wieck traveled together does not prove, imply, or even strongly suggest that the two were married, but statistically having more passages that include the two of them is correlated with them being married. Thus, such a passage should be viewed as providing some evidence in support of Robert Schumann having married Clara Wieck. The core weakness of Passage Term Match is that it can be too aggressive, assigning too much credit to candidate answers that are closely related to the content of the clue but are not actually correct; the components described in later sections address this weakness in various ways and thus complement Passage Term Match.

The Passage Term Match scorer evaluates each passage on the basis of which question terms it contains. Question terms $t_1 \ldots t_n$ are extracted from the question. A question term may be a single token, a multiword term taken from a lexicon of common terms, or a proper name. Then, for each passage, a score p_i is computed as the sum of inverse document frequency (idf) values of matching terms, normalized by the sum of idf values of all terms in the question. That is,

$$p_i = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{k=1}^{n} idf(t_k)},$$

where w_{ij} is defined as $idf(t_j)$ if passage i contains term t_j and 0 if otherwise. $idf(t)$ is the idf defined as

$$idf(t) = \log \frac{N}{c(t)+1},$$

where $c(t)$ is the number of documents that contain token t and N is the total number of documents in a large corpus. We used idf as weights for matches/mismatches instead of uniform weights because not all tokens are equally important. Matching a rarely used named entity is more informing than matching a frequently used stop word.

Skip-Bigram

The Skip-Bigram scorer computes a score on the basis of how many pairs of terms from a question and a passage match, where terms are connected or nearly connected in a structural graph. These graphs come from a combination of parsing and semantic relation detection, as described in the section on syntactic–semantic graphs above. The scorer attempts to match Skip-Bigrams, which are pairs of terms that are linked directly in the graphs or indirectly through only one intermediate node.

Like Passage Term Match, the Skip-Bigram scorer is largely a measure of co-occurrence; it determines the extent to which the answer appears in passages with words from the clue. However, Skip-Bigram specifically focuses on whether those words are close to each other in the syntactic–semantic graph. For example, given a question such as "Who invented the motor driven phonograph?," Skip-Bigram will give credit to passages that discuss "motor driven phonograph," "phonograph driven by a motor," "motor driving a phonograph," etc. Skip-Bigram will not give credit for passages that mention motors, driving, and phonographs in separate sentences or clauses and will thus ignore some passages that Passage Term Match awards a high score to. Those passages can provide some evidence in support of answers, but it is generally weaker evidence than the cases where the question terms are closely interconnected. As a result, the DeepQA final ranking algorithm is better off having both scores as distinct features than it would be with only one or the other.

The Skip-Bigram algorithm was first introduced in [9] as an automatic evaluation metric for machine translation, in which system translations are compared with human reference translations. Our algorithm is an extension of the original algorithm; it is capable of extracting Skip-Bigrams from a graph rather than a surface text. We evaluate the similarity between a passage and a question as a numeric score between 0 and 1 as follows.

Let P_{sb} and Q_{sb} be a set of Skip-Bigrams extracted from a passage and a question graph, respectively. We define a Skip-Bigram to be a pair of terms that either are directly connected in the graph or are both directly connected to a single common ("skipped") node in the graph. First, we compute two scores, i.e., the size of the common Skip-Bigrams with length normalization

$$score_P = \frac{|P_{\mathrm{sb}} \cap Q_{\mathrm{sb}}|}{|P_{\mathrm{sb}}|}$$
$$score_Q = \frac{|P_{\mathrm{sb}} \cap Q_{\mathrm{sb}}|}{|Q_{\mathrm{sb}}|}.$$

For purposes of computing common Skip-Bigrams, the scorer assumes that the focus of the question matches the candidate answer; bigrams including the focus in the question match bigrams including the candidate in the passage if the other term of each bigram matches. We have run experiments in which these scores were weighted by the idf scores of the matching terms; however, this configuration did not perform as well on our data.

The Skip-Bigram score is calculated by taking a harmonic mean of the two scores

$$score = \frac{2 \cdot score_P \cdot score_Q}{score_P + score_Q}.$$

The harmonic mean function was chosen for this purpose for roughly the same reason it is used to combine precision and recall scores in information retrieval (IR) studies: It provides a mechanism for integrating values that requires both values to be high to get a high combined score. In contrast, an arithmetic mean would allow a very short question or passage to have a misleadingly high score.

Textual Alignment

The Textual Alignment Candidate Scorer (TACS) judges a candidate on the basis of the surface similarity of a candidate-bearing passage and a question. It is motivated by the fact that sometimes one may find a passage that is similar to the question. For example, consider the question "Who is the president of France?" and the passage "Nicolas Sarkozy is the president of France." The passage is obviously strong justification for "Nicolas Sarkozy" as the answer.

Compared with Passage Term Matching, TACS is stronger when the word order is very important and informative. For the previous question, Passage Term Match would be misled by "President Clinton visited France," since it contains the question keywords, whereas TACS will return a much lower similarity score for these two statements because it contains different words and orders the words that are in common differently (e.g., "Who" in the question occurs before "president" in the question but "Clinton" appears after "president" in the passage; moreover, "visited" in the passage does not match anything in the question but falls between "president" and "France"). In contrast, the passage "France's president is Nicolas Sarkozy" is better handled by Passage Term Match than TACS because it contains the right words and expresses the right content but does not use the same (or even similar) word order.

Although one may find passages that are very similar to the question, they are almost never identical. Consider the following Jeopardy! question:

> "In 1698, this comet discoverer took a ship called the Paramour Pink on the first purely scientific sea voyage."

One can find the following passage:

> "Edmund Halley made probably the first primarily scientific voyage to study the variation of the magnetic compass."

In this example, the passage adds words such as "probably" and "primarily" and lacks the words "purely" and "sea" from the question. Because of these differences, the passage does not completely and unambiguously answer this question. However, it is close enough that a typical reader would interpret the passage as partial evidence in support of "Edmund Halley" as the answer. For our system to also interpret this passage in this way, we must use an algorithm that can handle the mismatches and the matches between the passage and the question. TACS handles this well by being robust with respect to differences and providing partial credit for incomplete alignment.

TACS also needs to be able to score partial matches. The passage justifies only the part of the question about "first purely scientific sea voyage" without any support for "a ship named Paramour Pink". A global aligning algorithm, such as edit distance, does not meet this challenge.

The input to the TACS is made of a question string, a passage string, the focus of the question, and the candidate being scored. The output is a similarity measure. To address the challenges noted above, we adopted the Waterman–Smith [10] algorithm that has been used for local DNA or amino acid sequence matching. Before we apply the alignment algorithm, the input question is transformed into a statement with the focus tokens replaced by the stub "FOCUS" and candidate tokens in the input passage replaced by the stub "CANDIDATE".

Then, the new question and passage are aligned according to an algorithm with three steps. First, it creates and initializes three arrays P, Q, and *score*. Single-dimension arrays P and Q contain the tokens in the passage and question, respectively. The 2-D array *score* is used to store the alignment score, and $score[i][j] = 0$ for all i and j. Second, it computes the value of each cell $score[i][j]$, where $i > 0$ and $j > 0$, using the formula

$$\max\begin{pmatrix} score[i-1][j-1] + \mathrm{sim}(P[i], Q[j]), \\ score[i-1],[j] + \mathrm{sim}(P[i], \phi), \\ score[i][j-1] + \mathrm{sim}(\phi, Q[j]), \\ 0 \end{pmatrix}.$$

Similarity function $\mathrm{sim}(t1, t2)$ is defined as follows:

$$\mathrm{sim}(t1, t2) = \begin{cases} \mathrm{idf}(t_1), & \text{if } t_1 = t_2 \\ -\mathrm{idf}(t_1), & \text{if } t_2 = \phi \\ -\mathrm{idf}(t_2), & \text{if } t_1 = \phi \\ -\mathrm{idf}(t_1), & \text{if otherwise,} \end{cases}$$

$$\mathrm{sim}(FOCUS, CANDIDATE) = \log(N).$$

where $idf(t)$ is as described in the "Passage Term Match" section and N is the number of documents in the corpus used to compute the idf statistics. By defining the similarity of the focus to the candidate to be log of the size of the corpus (i.e., the maximum possible idf score for any term), we ensure that the algorithm provides the most credit allowable for aligning the focus of the question to the candidate answer. Finally, the scorer returns the maximum value in *score* as the similarity score of the input question and passage.

Unlike Levenshtein edit distance or Needleman–Wunsch algorithm [11], Waterman–Smith algorithm finds local alignments, i.e., optimum subsequences of the input that align. The value of each cell $score[i][j]$, where $i > 0$ and $j > 0$, reflects the maximum similarity between $P[1, \ldots, i]$ and $Q[1, \ldots, j]$. It is a nonnegative value that is the maximum value of three possible ways to align $P[1, \ldots, i]$ and $Q[1, \ldots, j]$: aligning $P[1, \ldots, i-1]$ with $Q[1, \ldots, j-1]$ then replacing $P[i]$ with $Q[j]$, or aligning $P[1, \ldots, i-1]$ with $Q[1, \ldots, j]$ then inserting $P[i]$, or aligning $P[1, \ldots, i]$ with $Q[1, \ldots, j-1]$ then inserting $Q[j]$. Function $\mathrm{sim}(t1, t2)$ represents the similarity/cost of replacing term $t1$ with $t2$. If $t1$ is empty (i.e., ϕ), then $\mathrm{sim}(t1, t2)$ represents the similarity/cost of inserting $t2$.

Logical Form scoring

The Logical Form Answer Candidate Scorer (LFACS) attempts to align a syntactic–semantic graph of the content of a question to a syntactic–semantic graph of the content of a passage. LFACS attempts to match the question graph to the passage graph, given the constraint that the node for the focus of the question must correspond to the node for the candidate answer in the passage. The degree of the match constitutes the score that LFACS assigns to the candidate.

For example, for the question "Who wrote The Hobbit?" our syntactic graph has three nodes (*who*, *write*, and *The Hobbit*) and two labeled edges, which we present here using *node-edge-node* notation: (*write-subject-who*) and (*write-object-The Hobbit*). Our parser creates a similar graph of a passage "Tolkien wrote The Hobbit." The focus of the question is "who". If "Tolkien" is a candidate answer to the question, LFACS can attempt to align the two graphs, matching the *who* node in the question graph to the *Tolkien* node in the passage graph. In this case, the graphs perfectly align given that constraint; hence, LFACS assigns a high score to the candidate "Tolkien".

In the example above, we might also have another candidate "Dan Brown" with a supporting passage "Dan Brown wrote The Da Vinci Code." In this case, the passage aligns less well. The subject edge (*write-subject-who*) in the question perfectly aligns with the subject edge (*write-subject-Dan Brown*) in the passage because the focus, "who," aligns with the candidate, "Dan Brown." However, the object edge in the question only matches the object edge in the passage very weakly if at all; "write" in the question matches "write" in the passage perfectly, but depending on the term-matching strategy used, "The Hobbit" in the question may match "The Da Vinci Code" in the passage very little (because of a weak semantic relatedness) or not at all. Thus, LFACS would give relatively little score to the candidate "Dan Brown" based on this passage.

The examples above would also be addressed well by Textual Alignment. However, there are cases where LFACS is able to provide a value that Textual Alignment cannot because the graphs encode a deeper analysis of the question and the parse than mere co-occurrence or proximity. Thus, they can provide power for discriminating between false and true positives that are not available to answer scorers that count the number of term matches or align text based on word order. For example, a system might encounter passages such as the following:

"Dan Brown wrote several books and has read The Hobbit."

"Tolkien, an English author born in the late nineteenth century, wrote The Hobbit."

From the perspective of simple text matching, the former is at least as good a match as the latter for "Who wrote The Hobbit?" However, syntactically (and semantically), the latter passage is a better match.

The additional discriminating power that LFACS provides comes at the cost of substantial brittleness. Often, a passage retrieved for some candidate answer has many of the same words in roughly the same order as the question but does not relate those words using the same grammatical relationships. In some cases (as in the Dan Brown sentence in the previous paragraph), the answer is wrong and the passage should not be interpreted as supporting the answer;

for those examples, LFACS is more effective than any of the other scorers. However, in other cases, the answer is correct and the passage should be taken as supporting the answer. For example, consider the following:

"Tolkien wrote several books and The Hobbit is the one that is easiest to read."

This example should be interpreted as supporting the answer, but syntactically, it does not match the clue any better than the Dan Brown example. In some cases, this discrepancy can be addressed using semantic relation detection, as described earlier. However, relation detectors have limited coverage, and when they fail (as they do in this example) and when the passages use different syntactic structures than the clue, LFACS is less effective than the other answer scorers.

Once LFACS has aligned as much of the question as possible to the passage, it computes a score for each question term as the product of the *degree of match* and the *term weight*. The degree of match itself is a product of a *term match score* and a *structural match score*. The term match score indicates the degree to which that term in the question matched the corresponding term in the passage, e.g., "Bob Dole" matches "Bob Dole" perfectly but "Dole" imperfectly. The structural match score is the maximum across all paths from the question term to the focus of the product of all term match scores and edge match scores for the terms and edges on that path. As with other passage scorers, we have configured LFACS to use idf as the term weight in IBM Watson*. The final score that LFACS assigns to an answer in a single passage is the sum of the scores it assigns to each term. The algorithm for aligning graphs and computing scores in LFACS is described in more detail in [12].

In our experiment on blind data (described in detail in the "Evaluation" section), 1.9% of all questions were correctly answered in the full Watson configuration and were incorrectly answered in the Watson configuration with LFACS ablated. The net impact of LFACS (seen in that experiment) was less than 1.9% because some questions are correctly answered without LFACS but incorrectly with LFACS. Some of the questions for which adding LFACS resulted in a correct answer are the ones for which LFACS is able to thoroughly align the clue to the passage, for example,

Question: "The Mare Frigoris is the sea of this."

Passage: "Mare Frigoris (the "sea of cold") is a lunar mare located just north of Mare Imbrium, and stretches east to north of Mare Serenitatis."

In this example, shallow semantic relation detection is able to determine that the same relationship holds between "Mare Frigoris" and "sea" in both sentences. The PAS then connects "sea" to "of" and "of" to the focus and candidate answer. As a result of this strong score from LFACS, DeepQA answer ranking promotes the correct answer, "cold", over the answer it preferred without LFACS, "Mare Imbrium".

In contrast, there are many other cases where LFACS is able to elevate the correct answer into first place despite only matching a fraction of the clue, for example,

Question: "Silver Springs in northern Florida is one of the state's largest water-filled one of these holes."

Passage: "Devil's Hole is a large water-filled sinkhole close to the southeastern corner of Harrington Sound, Bermuda."

This passage does not strongly support this answer, in that it says nothing about Silver Springs. However, it does suggest that a sinkhole can be both large and water-filled. In contrast, the answer that the Watson configuration with no LFACS selected for this answer was "Ocala", a town near Silver Springs that none of our passages describe as "large" or "water-filled." Thus, LFACS provides inconclusive but still quite useful evidence.

In some cases, LFACS provides only weak inconclusive evidence even when the passage does provide strong support, for example,

Question: "In 'A Christmas Carol', we learn this man once apprenticed with the jolly merchant Mr. Fezziwig."

Passage: "The character Fezziwig owned the business where Scrooge was apprenticed in 'A Christmas Carol'."

Here, LFACS correctly recognizes that the focus ("man") and candidate ("Scrooge") are the objects of the verb "apprenticed" in both sentences. However, it fails to further connect these terms to important matching terms such as "A Christmas Carol" and "Fezziwig" because of differences in the syntactic structure (e.g., the preposition "in" is attached to "learn" in the question and "apprenticed" in the passage) and a lack of semantic relations to compensate. Thus, LFACS gets a fairly weak score for Scrooge, but none at all for the answer that Watson without LFACS prefers: Charles Dickens. If LFACS had made the other connections, Watson would have even higher confidence in the correct answer. However, the simple fact that Scrooge is characterized as being apprenticed in some passage was enough evidence to convince the DeepQA answer-ranking component to promote "Scrooge" ahead of "Charles Dickens" (who has excellent scores from other passage scorers, but 0 from LFACS for this question).

In future work, we intend to pursue approaches that combine some of the ability of LFACS to make subtle distinctions that depend on deep semantic analysis with some of the flexibility of our other passage-scoring components. We intend to do so by more aggressively matching patterns of structures that have equivalent or entailing content, e.g., attempting to more aggressively connect to "A Christmas Carol" or "Fezziwig" in the example above. This is a careful balancing act because drawing connections too aggressively would cause LFACS to behave similarly to Passage Term Match, which would make it redundant and would eliminate its ability to introduce fine-grained distinctions.

Merging across passages

The algorithms described in earlier sections each assigns a distinct score for each passage that a candidate answer appears in. During the final merging and ranking stage of the DeepQA architecture [3], these scores for a given candidate answer are merged so that each algorithm provides a final score for each candidate answer; that merged score is used as input to the statistical model that scores and ranks candidate answers. We have the following three distinct algorithms that we use to merge features across passages.

- *Maximum*—The final score for the candidate answer is the maximum score for that answer in any passages found for that answer.
- *Sum*—The final score for the candidate answer is the sum of the scores for that answer in each of the passages found for that answer.
- *Decaying sum*—The final score for the candidate answer is computed to be $\sum_{i=0}^{m}(p_i/2^i)$, where $p_0 \ldots p_m$ are the scores of the passages that contain the answers, sorted in descending order.

The maximum merging strategy is best suited for scorers for which having a very high score from one passage is much more informative than having many weaker passages. In contrast, the sum merging strategy is best for scorers for which accumulating large quantities of results across many different passages is more important that finding any single passage that matches very well. The decaying sum strategy provides a compromise between these two extremes; a few strong passages will consistently outscore many mediocre ones (as with maximum), but many strong passages still outscore a few strong passages (as with sum). We have found maximum to be most effective for LFACS, which is consistent with the intuition that, if LFACS finds one passage that unambiguously answers the question, that passage should count more than any number of passages that only address part of what the question is asking for. In contrast, no single passage can provide enough evidence that Passage Term Match should be strongly convinced that the answer is right (since co-occurring with *all* of the question terms in one passage is still not particularly solid evidence of correctness). Instead, the value of Passage Term Match lies in accumulating results across passages; right answers tend to frequently co-occur with the question terms across many passages. Thus, a merging strategy such as sum or decaying sum is a better fit for Passage Term Match. Textual alignment and Skip-Bigram lie between these two extremes but also appear to benefit from accumulating results across passages. Specifically, we have found that merging by sum is most effective for Skip-Bigram and merging by decaying sum is most effective for Textual Alignment and Passage Term Match.

One limitation of all three of these merging strategies is that they fail to distinguish between multiple passages that all address the same parts of some question versus multiple passages that address different parts of some question. For example, if we have a question that asks for an actor who appeared in *Stigmata* and *The Usual Suspects*, we would want to give a good score to a candidate that had three passages saying he appeared in *Stigmata* and two saying that he appeared in *The Usual Suspects*, but a much lower score to a candidate that had ten passages saying that he appeared in one of these movies but none for the other movie. We intend to address this limitation in future work and to explore other ways of combining feature values across passages.

Evaluation

We test our hypotheses in the context of two different DeepQA configurations. The full configuration includes all of the components of the complete Watson system configured for Jeopardy! question answering [13]. The Watson answer-scoring baseline configuration includes all of the standard question analysis, search, and candidate generation, but only one answer scorer (which checks answer types using a named entity detector [14]) and a simplified configuration for merging and ranking answers. All experiments were conducted on a previously unseen set of 3,508 questions.

Table 1 shows the overall impact of all of the passage scoring in the full and baseline configurations. The second column of the table shows the performance with all four of the passage-scoring algorithms described above, but no SER. In this condition, the only passages that were available for passage scoring were the ones that came directly from primary passage search, which uses only keywords in the original questions [2]. In contrast, the third column also includes SER providing additional passages for each candidate. Having more passages for each candidate makes the passage-scoring algorithms more effective. Note that the "no passage scoring" configuration in the full system (here and below) omits only the four passage-scoring components described in this paper. Some other components

Table 1 Accuracy with and without SER and passage scoring.

	No passage scoring	*All passage scoring without SER*	*All passage scoring with SER*
		Watson answer-scoring baseline configuration	
Accuracy	54.9%	58.8%	61.7%
Difference vs. no passage scoring		+4.0%	+6.9%
		Full configuration	
Accuracy	67.1%	68.3%	70.4%
Difference vs. no passage scoring		+1.3%	+3.3%

Table 2 Accuracy (with SER) with no passage scoring versus each passage scorer separately.

	No passage scoring	*Passage Term Match*	*Skip-Bigram*	*Textual Alignment*	*LFACS*
		Watson answer-scoring baseline configuration			
Accuracy	54.9%	58.4%	57.6%	58.1%	57.0%
Difference vs. no passage scoring		+3.6%	+2.8%	+3.2%	+2.1%
		Full configuration			
Accuracy	67.1%	68.9%	68.7%	68.6%	67.5%
Difference vs. no passage scoring		+1.8%	+1.7%	+2.1%	+0.4%[a]

[a]Difference is not statistically significant.

do still make limited use of the supporting passages, e.g., for determining whether the answer has the desired type [14]. The four scoring components described in this paper are the only components that attempt to match the entire question to the entire passage and, thus, are the only components that we classify as passage-scoring components.

The baseline and the full systems are both more effective at correctly answering questions with SER and all four passage-scoring algorithms. As noted earlier, results from the passage-scoring algorithms are statistically combined with results from other components (search and other answer-scoring components) in the DeepQA Final Merging and Ranking component [3]. The impact of SER and passage scoring is much larger in the baseline configuration than in the full configuration; they have more room to contribute in those cases. In both the baseline and full configurations, passage scoring with SER provides higher accuracy than passage scoring without SER, and passage scoring (even without SER) provides better accuracy than no passage scoring. All of these differences are statistically significant (here and below, significance is assessed for $p < .05$ using McNemar's test with Yates' correction for continuity). SER is used in all of the experiments described in the remaining tables.

Table 2 shows the results of adding each scorer, alone, to the system with none of these four passage-scoring components. This table shows that each of the passage scorers is, by itself, more effective than no passage scoring. The difference versus no passage scoring is significant in all cases except for LFACS in the full configuration.

Table 3 shows the results of ablating each passage scorer, one at a time, from a system that has all four scorers. In the baseline configuration, ablating any one of the four scorers had a negative impact of 1% or more on accuracy, and these differences were all statistically significant; ablating all four has an impact of nearly 7% on the baseline system (also significant). In the full system, the impact of ablating a single answer scorer is much more muted of course; Skip-Bigram and LFACS had the largest impact, and those differences (versus all passage scoring) were significant; ablating Passage Term Match or Textual Alignment from the system does not show a significant impact on this test set. Ablating all four has an impact of more than 3% on the full system.

One key observation across Tables 2 and 3 is that LFACS has the least impact for both configurations in Table 2 but the greatest impact in Table 3 for the baseline configuration and one of the greatest impacts (along with Skip-Bigram)

Table 3 Accuracy (with SER) with all four passage scoring versus all except each passage scorer.

	All passage scoring	*Ablating Passage Term Match*	*Ablating Skip-Bigram*	*Ablating Textual Alignment*	*Ablating LFACS*	*No passage scoring*
			Watson answer-scoring baseline configuration			
Accuracy	61.7%	60.7%	60.7%	60.0%	58.3%	54.9%
Difference vs. all passage scoring		−1.0%	−1.1%	−1.8%	−3.4%	−6.9%
			Full configuration			
Accuracy	70.4%	70.0%	69.9%	70.5%	70.0%	67.1%
Difference vs. All passage scoring		−0.3%[a]	−0.4%	+0.2%[a]	−0.4%	−3.2%

[a]Difference is not statistically significant.

Table 4 Precision and recall of the components relative to the global mean score from each component, with a correcting factor for class imbalance.

	Passage Term Match	*Skip-Bigram*	*Textual Alignment*	*LFACS*
Relative Precision	73.7%	81.4%	75.3%	86.2%
Relative recall	91.8%	90.4%	84.9%	57.5%
Relative F *measure*	81.7%	85.7%	79.8%	69.0%

for the full configuration. The differences between LFACS and other scorers in Table 2 were all significant except for the difference with Skip-Bigram in the baseline configuration. The differences between LFACS and other scorers in Table 3 were all significant in the baseline configuration, but not in the full configuration. The key distinction here is that if you are only employing one passage-scoring strategy, LFACS is not a particularly strong one, but if you already have several passage-scoring strategies, LFACS is a particularly strong addition. This reflects the additional depth and subtlety of the distinctions that LFACS can make; with a set of scorers in place that extracts much of the available signal to be had from co-occurrence and proximity, there is substantial demand for additional scorers to employ a deeper analysis.

We have also employed a wide variety of component-level metrics to determine how effective each of the different scorers is in isolation. There are a wide variety of established IR metrics for measuring passage-ranking algorithms, but they fail to directly measure the effectiveness of a score at making individual judgments of answer correctness in isolation. Doing so is difficult because each component gives a numerical score rather than a simple binary judgment of correctness; hence, there is not a direct obvious way to compute how often each scorer is "right" on a per-answer basis. One way to artificially approximate such a behavior is to compute a mean score for each feature across all answers to all questions and to treat each score as a judgment of correctness if and only if that score is greater than the mean. One can then measure precision as the percentage of all answers judged correct that are correct and the recall as the percentage of all answers that are correct that are also judged correct. This approach produces misleadingly low precision scores because there is a severe class imbalance (far more wrong candidate answers than right candidate answers). To correct for this, we discount the number of answers judged incorrect for computing the precision by the ratio of right answers to wrong answers (to approximate the behavior, we would see if we tested on the same number of right and wrong answers). We refer to these metrics as "relative precision" and "relative recall." We define "relative F measure" to be the harmonic mean of these metrics.

Table 4 shows these results for the four passage-scoring components. Not surprisingly, Passage Term Match has the highest recall and lowest precision, whereas LFACS has the lowest recall and highest precision. Textual Alignment and Skip-Bigram both lie between these two extremes, with Skip-Bigram having the highest F measure, as it provides nearly as much recall as Passage Term Match while showing substantially higher precision. Passage Term Match, Textual Alignment, and Skip-Bigram all show very high recall, reflecting their ability to provide meaningful scores even when there are no very strong passages. In contrast, the added precision of LFACS comes with much lower recall because there are many questions for which LFACS is not able to align *any* part of the question to any passage containing the correct answer.

Related work

SER is similar to existing work that takes the candidate answers produced by a traditional question-answering system and exploits the redundancy of information in a large corpus (typically the web) to distinguish correct from incorrect answers [15–18]. However, most of that work is statistical, concerned only with the number of documents retrieved by a query, rather than a deep natural-language processing analysis of the returned content. In [16], a content-based approach was tried, but the analysis was limited to the token distance between the answer and question terms. Another difference from much of the prior work is the fact that our implementation of SER does not access the Internet at run time and instead uses local corpora (including Wikipedia** and many other sources) for retrieving supporting evidence. The local implementation allows us to efficiently scale up to large numbers of candidate answers per question, at which point the performance of a web implementation would not be acceptable.

Work on scoring candidate answers in passages using syntax and/or semantics has been largely separated from work on answer validation using a secondary search; the inputs to semantic passage scoring have been, generally, passages from a single primary search using the original question terms, not additional passages separately retrieved for each candidate. Work of this sort includes structural distance [19], parse tree similarity [20, 21], and logical form unification [6].

There is also a very substantial body of work in scoring passages for relevance to a query, e.g., [22–24]. This task is not quite the same as scoring a specific candidate answer, but there is a considerable overlap in the kinds of techniques that are relevant to these tasks. There are three general differences between established algorithms for ranking retrieved passages and the passage scorer approach we describe in this paper.

1. We use a set of passage-scoring algorithms that each provide a distinct score that is used by the final merger as a feature in statistically judging whether each answer is correct. Most existing passage-scoring work uses a single algorithm. An assortment of scorers is combined through voting in [25]; that differs from our approach both in that the combination does not use statistics to weigh the contribution of the different algorithms and in that the results are locally combined as part of the passage-ranking task instead of being combined with a broader set of features to determine answers to questions.
2. All of our scorers, except Passage Term Match, attempt to align the question focus to a designated candidate answer. With that said, however, both Textual Alignment and Skip-Bigram are able to operate (with some minor degradation in effectiveness) without a focus to align to. We would expect that they could be used to rank passages with little or no revision.
3. Our scorers are applied to an assortment of passages for each candidate answer with results aggregated across multiple passages for a single answer, as described in the section "Merging across passages." This is a distinction on how these components are used that suggests substantially different criteria for judging effectiveness. For example, the combined Passage Term Match score across many passages for a single answer is an effective mechanism for judging answer correctness because the right answer tends to co-occur with terms from the question more often than wrong answers do. This does not necessarily suggest that the Passage Term Match score is a particularly useful feature for judging whether any one passage answers the question. Thus, while some of our scorers could be used to rank passages, we have no direct evidence that they would be particularly effective at that task.

Given an annotated set of examples of passages that do or do not provide justification for some answer being correct, it is possible to build a classifier to distinguish among these passages; one promising approach for doing so is kernel methods [26, 27]. Identifying justifying passages in order to persuade a user that some answer is accurate is essential for many real-world applications of question-answering technology. Thus, we expect this goal to be important in our future work. Some combination of the algorithms described here and established techniques such as kernel methods may address this task well.

The use of a harmonic mean of match scores in Skip-Bigram can be compared and contrasted with many systems that combine scores of various types of match. Vector-based approaches such as term frequency-idf (tf-idf) are popular for combining match scores for large amounts of text such as full documents. For smaller amounts of text such as the supporting passages in DeepQA, harmonic mean has been employed for a wide variety of tasks such as translation [28, 29], text summarization [30], and complex question answering with sentence-length answers [31]. Our Skip-Bigram scorer provides a novel combination of this established mathematical foundation with the use of both syntactic and semantic relations to find connected or nearly connected pairs of terms.

The Waterman–Smith algorithm used in Textual Alignment is also widely used for bioinformatics [32]. Unlike these applications, we utilized our knowledge of word frequency to gauge the impact of a term match or mismatch in the form of idf weights.

The core concept of LFACS is answering questions by aligning a graph encoding of the semantics of the question to a graph encoding of the semantics of content containing a potential answer. This approach is relatively common in question answering. In some cases, links in the graph are deep semantic relations in a handcrafted ontology (e.g., [33–35]). In others, links are syntactic relations in a parse or PAS (e.g., [6, 20, 36]). As noted earlier, LFACS uses hybrid syntactic–semantic graphs that combine these two approaches. Some of the aforementioned systems (e.g., [6]) use theorem-proving technology to "prove" that the answer satisfies the requirements of the logical form of the question given the logical form of the passage. These systems treat the *node-edge-node* triples as logical assertions where each edge label is a logical predicate and the nodes are the instances being predicated. LFACS takes a similar but not identical perspective; specifically, it attempts to match the graphs structurally.

The theorem-proving approach allows the reasoner to infer additional assertions (i.e., additional edges in the graph) during matching in a goal-directed way. For example, as noted earlier, natural-language axioms [6] serve a similar function as our shallow semantic relations. Those axioms are applied as needed by the theorem prover to relate a passage to a question, whereas our semantic relation detection is run separately on the question and the passage *before* running the scorers. In theory, this distinction could provide some benefits to theorem proving in efficiency (since it needs to apply only those axioms that are needed to do unification) or even expressive power. However, we are not aware of any shallow semantic relations (either in our set or in the examples reported in [6]) for which it is not feasible to simply apply them as a preprocessing step.

Matching is a more natural paradigm than theorem proving for a component that provides partial credit for an incomplete alignment. This capability is extremely important in Jeopardy! (as in many real-world applications) because Jeopardy! clues are often complex and multifaceted. Theorem proving is generally an all-or-nothing technology, and systems that use theorem proving to align questions to passages typically only consider whether the content in the passage is able to prove *all* of the content requested in the question. This is feasible for extremely simple questions but is generally a poor fit for questions as complex as those found in Jeopardy! and many real-world applications. It would be possible to use a theorem prover as part of a system that provided partial credit, e.g., by treating different subgraphs of the question as distinct queries and scoring based on how many of those subgraphs can be proven. However, this seems awkward and potentially inefficient. Matching graph structures seems like a more useful perspective than theorem proving when the goal is to compute a degree of match.

A structural-matching approach is employed in [35] for answering questions from a knowledge-base using a "similarity strategy," in which characteristics of one entity are hypothesized to apply to similar entities. That work uses structure mapping [37] to judge similarity of entities. The structural-matching algorithm in LFACS is also a form of structure mapping, but the task of matching syntactic–semantic graphs of question and passage text is very different from the task of matching entities in a formal knowledge-base.

Conclusion

We have shown that our suite of passage-scoring strategies has a significant impact on our ability to answer questions. We have further shown that this suite is more effective than any of its constituents. The four algorithms we describe here use different kinds of analyses to assess answers within passages and thus can draw different conclusions. For example, Skip-Bigram and LFACS use the graph structure, which can be helpful when the analysis components [4, 5] are correct and useful passages are similarly structured to questions; Passage Term Match and Textual Alignment do not benefit from the graph structure but are therefore never misled by it. Additional passage-scoring algorithms would also be helpful if they were meaningfully different from the existing algorithms. However, the more passage-scoring algorithms that a system has, the less room there is for additional algorithms to squeeze additional signal out of the same passages.

We have demonstrated that a question-answering system that performs analysis of passages can significantly benefit from an SER phase that uses the candidate answer as part of the query. We have also shown that our passage-scoring algorithms are more effective given passages from SER than they are when limited to passages retrieved by DeepQA's primary search. The SER approach has the downside of being very slow to execute as it requires executing one search query per candidate answer. DeepQA employs a parallel deployment of this system, which can process the candidate answers concurrently across a cluster of machines [38]. As a result, Watson benefits from acquiring and scoring supporting passages and is still able to answer questions very quickly.

We observed that one of our strategies, LFACS, is less effective than the others in isolation but is among the most effective when included in the full suite. LFACS is a very complex scoring component that makes very fine-grained distinctions. Furthermore, LFACS is heavily dependent on relation extraction, which is very challenging to do well. The LFACS approach seems like a poor choice for building a simple inexpensive system to do question answering moderately well; it is very expensive to get much effectiveness out of LFACS, and developers with limited resources can get more impact much more easily from a scorer such as Passage Term Match, which is cheap and easy to implement. However, as a question-answering system becomes more complex and sophisticated, the potential for very shallow approaches to have additional impact becomes increasingly thin. As a result, the need for a deeper analysis that makes more complex distinctions increases. Complex passage-scoring algorithms such as LFACS do provide a substantial value when included as part of a more comprehensive suite. In future work, we expect algorithms such as LFACS that use deep analysis results to make relatively subtle distinctions to be an increasingly important part of our research agenda.

*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., or Wikimedia Foundation in the United States, other countries, or both.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. D. Moldovan, C. Clark, S. Harabagiu, S. Maiorano, and Assoc. Comput. Linguistics, "COGEX: A logic prover for question answering," in *Proc. NAACL Human Lang. Technol. Conf.*, Stroudsburg, PA, 2003, vol. 1, pp. 87–93, DOI:10.3115/1073445.1073467.
7. B. Levin, *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago, IL: Univ. Chicago Press, 1993.
8. K. Kipper, A. Korhonen, N. Ryant, and M. Palmer, "A large-scale classification of English verbs," *Lang. Resour. Eval. J.*, vol. 42, no. 1, pp. 21–40, Mar. 2008.
9. C.-Y. Lin, F. J. Och, and Assoc. Comput. Linguistics, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," in *Proc. 42nd Annu. Meet. ACL*, Stroudsburg, PA, 2004, Article 605, DOI:10.3115/1218955.1219032.
10. T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.
11. S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, Mar. 1970.
12. J. W. Murdock, "Structure mapping for Jeopardy! Clues," in *Proc. 19th ICCBR*, 2011, pp. 6–10.
13. 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.
14. 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.
15. B. Magnini, M. Negri, R. Prevete, and H. Tanev, "Is it the right answer? Exploiting web redundancy for answer validation," in *Proc. 40th Ann. Meet. Assoc. Comput. Linguistics*, 2002, pp. 425–432.
16. B. Magnini, M. Negri, R. Prevete, and H. Tanev, "Comparing statistical and content-based techniques for answer validation on the web," in *Proc. VIII Con-vegno AI*IA*, Sienna, Italy, 2002.
17. C. Clarke, G. Cormack, and T. Lynam, "Exploiting redundancy in question answering," in *Proc. 24th SIGIR Conf.*, 2001, pp. 358–365.
18. E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng, "Data-intensive question answering," in *Proc. 10th TREC*, 2001, pp. 393–400.
19. J. Prager, D. Radev, E. Brown, A. Coden, and V. Samn, "The use of predictive annotation for question answering in TREC8," in *Proc. 8th TREC*, 1999, pp. 399–410.
20. J. Chu-Carroll, K. Czuba, J. Prager, and A. Ittycheriah, "Two heads are better than one," in *Proc. HLT-NAACL*, 2003, pp. 24–31.
21. R. Sun, J. Jiang, Y. Fan, T. Hang, C. Tatseng, and C. M.-Y. Kan, "Using syntactic and semantic relation analysis in question answering," in *Proc. 14th TREC*, 2005, pp. 1–9. [Online]. Available: https://mailserver.di.unipi.it/ricerca/proceedings/TREC2005/papers/nus.qa.pdf
22. J. P. Callan, "Passage-level evidence in document retrieval," in *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, W. B. Croft and C. J. van Rijsbergen, Eds., New York, NY, 1994, pp. 302–310.
23. A. Corrada-Emmanuel, W. B. Croft, and V. Murdock. (2003). "Answer passage retrieval for question answering," Center Intell. Inf. Retrieval, Univ. Massachusetts, Amherst, MA, Tech. Rep. [Online]. Available: http://ciir.cs.umass.edu/pubfiles/ir-283.pdf
24. M. W. Bilotti, E. Nyberg, and Assoc. Comput. Linguistics, "Improving text retrieval precision and answer accuracy in question answering systems," in *Proc. 2nd IRQA Coling Workshop*, Stroudsburg, PA, 2008, pp. 1–8.
25. S. Tellex, B. Katz, J. Lin, A. Fernandes, G. Marton, and Assoc. Comput. Mach., "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, New York, 2003, pp. 41–47.
26. A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar, "Exploiting syntactic and shallow semantic kernels for question/answer classification," in *Proc. 45th ACL Conf.*, Prague, Czech Republic, 2007. [Online]. Available: http://www.ist-luna.eu/pdf/ACL07.pdf
27. A. Moschitti, "Kernel methods, syntax and semantics for relational text categorization," in *Proc. ACM 17th CIKM*, 2008, pp. 253–262.
28. D. Melamed, R. Green, J. P. Turian, and Assoc. Comput. Linguistics, "Precision and recall of machine translation," in *Proc. NAACL-Short—Companion Volume of the Proceedings of HLT-NAACL 2003—Short Papers, Volume 2*, Stroudsburg, PA, 2003, vol. 2, pp. 61–63.
29. A. Agarwal, A. Lavie, and Assoc. Comput. Linguistics, "Meteor, M-BLEU and M-TER: Evaluation metrics for high-correlation with human rankings of machine translation output," in *Proc. 3rd Workshop Statist. Mach. Trans.*, 2008, pp. 115–118.
30. C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. ACL Workshop Text Summarization*, 2004, pp. 25–26.
31. J. Fukumoto, "Question answering system for non-factoid type questions and automatic evaluation based on BE method," in *Proc. 6th NTCIR Workshop*, 2007, pp. 441–447.

32. R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge Univ. Press, 1999.
33. W. Lehnert, "Human and computational question answering," *Cogn. Sci.*, vol. 1, no. 1, pp. 47–73, Jan. 1977.
34. D. B. Lenat, "CyC: A large-scale investment in knowledge infrastructure," *Commun. ACM*, vol. 38, no. 11, pp. 33–38, Nov. 1995.
35. P. K. Paritosh and K. D. Forbus, "Analysis of strategic knowledge in back of the envelope reasoning," in *Proc. 20th AAAI Nat. Conf. Artif. Intell.*, vol. 2, A. Cohn, Ed., 2005, vol. 2, pp. 651–656.
36. U. Hermjakob, E. H. Hovy, and C. Lin, "Knowledge-based question answering," in *Proc. 6th World Multiconf. SCI*, Orlando, FL, 2002. [Online]. Available: http://research.microsoft.com/en-us/people/cyl/sci2002.pdf
37. B. Falkenhainer, K. Forbus, and D. Gentner, "The structure mapping engine: Algorithm and examples," *Artif. Intell.*, vol. 41, no. 1, pp. 1–63, Nov. 1989.
38. E. A. Epstein, M. I. Schor, B. S. Iyer, A. Lally, E. W. Brown, and J. Cwiklik, "Making Watson fast," *IBM J. Res. Dev.*, vol. 56, no. 3/4, Paper 15, pp. 15:1–15:12, May/Jul. 2012.

Received July 18, 2001; accepted for publication November 30, 2001

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.

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 his 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 the Association for Computational Linguistics.

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, Yorktown Heights, NY. He received a B.S. degree in computer science from Rensselaer Polytechnic Institute, Troy, NY, in 1998 and an M.S. degree in computer science from Columbia University, New York, NY, 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.

Hideki Shima *Carnegie Mellon University, Language Technologies Institute, Pittsburgh, PA 15213 USA (hideki@cs.cmu.edu)*. Mr. Shima is a Ph.D. candidate at the Language Technologies Institute in the Carnegie Mellon University School of Computer Science. He received his B.S. degree in information and computer science from Waseda University, Tokyo, Japan, in 2004 and his M.S. degree in language technologies from Carnegie Mellon University in 2006. In 2009, he spent 3 months as a research intern in the DeepQA project, in the IBM Thomas J. Watson Research Center. He organized three NTCIR (NII [National Institute of Informatics] Test Collection for Information Retrieval) shared tasks, ACLIA1, ACLIA2, and RITE; served as referee/program committee at several conferences [ACL (Association for Computational Linguistics), Conference on Information and Knowledge Management (CIKM), Empirical Methods on Natural Language Processing (EMNLP), Special Interest Group on Information Retrieval (SIGIR), etc.]. He is a member of the ACL.

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 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, 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.