Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2007), *21*, 155–171. Printed in the USA.

DOI: 10.1017/S0890060407070205

Answering engineers' questions using semantic annotations

SANGHEE KIM, ROB H. BRACEWELL, AND KEN M. WALLACE
Engineering Design Centre, Department of Engineering, University of Cambridge, Cambridge, United Kingdom
(RECEIVED October 27, 2005; ACCEPTED July 25, 2006)

Abstract

Question–answering (QA) systems have proven to be helpful, especially to those who feel uncomfortable entering keywords, sometimes extended with search symbols such as +, ∗, and so forth. In developing such systems, the main focus has been on the enhanced retrieval performance of searches, and recent trends in QA systems center on the extraction of exact answers. However, when their usability was evaluated, some users indicated that they found it difficult to accept the answers because of the absence of supporting context and rationale. Current approaches to address this problem include providing answers with linking paragraphs or with summarizing extensions. Both methods are believed to be sufficient to answer questions seeking the names of objects or quantities that have only a single answer. However, neither method addresses the situation when an answer requires the comparison and integration of information appearing in multiple documents or in several places in a single document. This paper argues that coherent answer generation is crucial for such questions, and that the key to this coherence is to analyze texts to a level beyond sentence annotations. To demonstrate this idea, a prototype has been developed based on rhetorical structure theory, and a preliminary evaluation has been carried out. The evaluation indicates that users prefer to see the extended answers that can be generated using such semantic annotations, provided that additional context and rationale information are made available.

Keywords: Information Retrieval; Natural Language Processing; Question–Answering; Rhetorical Structure Theory; Semantic Annotations

1. INTRODUCTION

Electronic documents are one of the most common information sources in organizations, and approximately 90% of organizational memory exists in the form of text-based documents. It has been reported that 35% of users find it difficult to access information contained in these documents, and at least 60% of the information that is critical to these organizations is not accessible using typical search tools (80-20 Software, 2003). There are two main problems. The first is that there is simply too much information to be searched. The second is that differences exist between the indexing approaches used in search engines and the way people perceive and access the contents of documents. This means that most users find searching for relevant information difficult because it is not possible for them to enter keywords in a sufficiently precise form for them to be used effectively by current search engines.

Current retrieval systems accept queries from users in the form of a few keywords and retrieve a long list of matching documents. Users then have to sift through the documents to locate the information they are looking for. For simple fact-based queries, for example, *What material should be used for this turbine blade?*, most users can enter satisfactory keywords that rapidly find the required answers. However, keyword-based systems cannot cope with questions involving the following: (1) comparing, for example, *What are the advantages and disadvantages of using aluminum compared with steel for this saucepan?*; (2) reasoning, for example, *How safe are commercial flights?*; and (3) extracting answers from different documents and fusing them into a complete answer. To obtain useful answers to these types of question, users currently have to expend considerable time and effort.

Reprint requests to: Sanghee Kim, Engineering Design Centre, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK. E-mail: shk32@eng.cam.ac.uk

In previous research in this domain, automatic query expansion and taxonomy-based searches have been proposed. Query expansion improves on keyword searching for short questions that require only a few documents to be located to provide the answers. Taxonomy-based searches require the hierarchical organization of domain concepts. This relieves the user of having to enter accurate keywords as information is searchable by selecting concepts. However, considerable effort is required to create the classifications and maintain the hierarchy. For example, Yahoo (http://www.yahoo.com) employs around 50 subject experts to maintain their directories and indexes. Not many organizations can afford to adopt such a strategy, and current automatic classifiers only achieve 60–80% accuracy (Mukherjee & Mao, 2004). This means that in automatic classification around a third of the documents will be missed or misclassified. Manual classifications are subjective, and often based on the few sentences that are deemed important for individual indexers. Clearly, when information is sought that does not match the indexing, such a taxonomy-based approach does not help.

Offering users the facility to enter their queries in natural language might greatly enhance current search engine interfaces and be particularly helpful for less experienced users who are not adept at advanced keyword searches. Recent research into natural language-based retrieval systems has mainly pursued a question–answering (QA) approach. QA systems have successfully retrieved short answers to natural language questions instead of directing users to a number of documents that might contain the answers. Typically, QA uses a combination of information retrieval (IR) and natural language processing (NLP) techniques. IR techniques are used to pinpoint a subset of documents and to locate parts of those documents that are related to the questions. NLP techniques are used for extracting brief answers. There is great interest in developing robust and reliable QA systems to exploit the enormous quantity of information available online to answer simple questions such as *Who is the president of USA?* This is relatively easy because straightforward NLP techniques, such as pattern matching, are sufficient to answer it. The numerous occurrences and multiple reformulations of the same information available on the Web greatly increase the chance of finding answers that are syntactically similar to the question (Brill et al., 2001). On the intranets run by organizations, the quantity of information, although large, is much less than on the Web, and the number of occurrences and reformulations is less exhaustive.

Unlike users searching on the Web, those in organizations are likely to ask questions that are not easily answered by simply looking up syntactic similarities in databases. Such answers can be considered complex, and may need to be inferred from different parts of a single text or from multiple texts. The initial question posed by a user may be ill formed, that is, too broad or too specific, making it difficult for the retrieval system to interpret, and hence, further interaction with the user is often necessary. Answering such complex questions has received little attention within QA research. To answer such questions, the issues of correctly interpreting the question and presenting the answer must both be addressed. When presenting answers to complex questions, it is not sufficient just to present the answer, that is, the user needs additional supporting information with which to assess the trustworthiness of the answer. For example, *hemlock poisoning* and *drinking hemlock* can both be considered correct answers to the question, *How did Socrates die?* (Burger et al., 2001). Users who have some background knowledge about poisoning might appreciate a brief answer, as they do not want to read through a long text to extract the answer themselves. Other users with less background knowledge might prefer to see where the answers came from and want to read more text explaining the answers before accepting them. Searching precisely for *How did Socrates die?* on the Web using Google produces around 748,000 results. It is clear that with a question such as this, answers with varying formulations were mentioned in numerous documents or in many parts of a single document. Answers appearing as multiple instances need to be fused efficiently to reduce repeated information. Apart from what is simply stated in the question, the user's real intention might have been to know the reason why Socrates chose to die by poisoning. For questions that are ill formed, it is important that answers are extended with related information that increases a user's understanding of the answers. It is therefore necessary to research suitable ways of presenting answers in a clear and coherent manner, and providing sufficient supporting information to allow users to decide whether or not to trust the answers.

A combination of two approaches, both using semantic relations, is therefore proposed for presenting clear and coherent answers. First, duplicate information is removed. Second, answers are synthesized from multiple occurrences, and then justified by adding supporting information. To achieve this, semantic analysis is necessary of both the questions and the texts from which the answers are to be extracted. Figure 1 shows an example of how these ideas might be implemented. An example text is from http://www.tsb.gc.ca/en/reports/air/1996/a96o0125/a96o0125.asp. The initial question posed by the user, that is, *What triggered the engine fire alarm in Boeing 727-217?*, was aimed at understanding the cause or causes of the fire alarm going off. The question itself is ambiguous, because it does not specify the engine or the date of the flight. Assuming that the system correctly understands the question, it can return *the failure of the number 2 engine starter* as the cause. However, for complex incidents such as this case, it is difficult to pinpoint particular causes and regard them as independent of the remaining information. That is, there might be more than one cause for a single incident, and some causes may depend on other causes. For example, in the example above there are other contributing causes, for example, *the start valve had reopened because of a short circuit or the engine starter had failed because of overspeeding*. There could also be consequent effects, for example, *residual smoke and fire damage to the structure surrounding the number 2 engine*. For presenting answers like this, it is

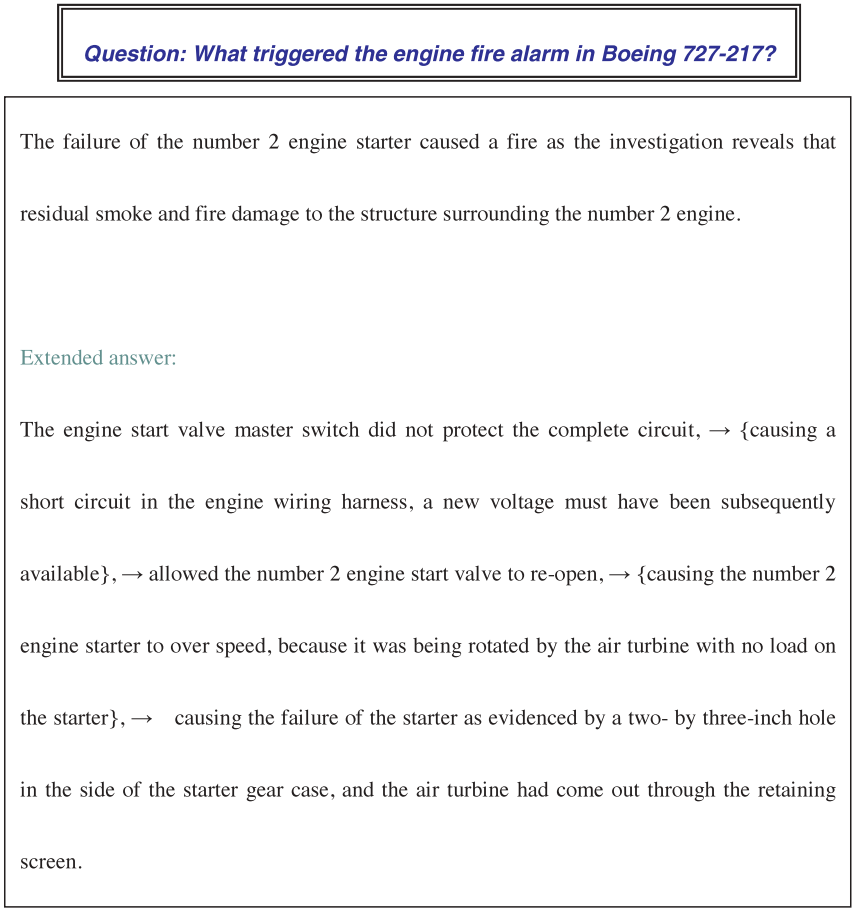

Fig. 1. An example of generating a coherent and justified answer. [A color version of this figure can be viewed online at www.journals.cambridge.org]

necessary to consider the actual information needs of the user from a knowledge level. For example, when faced with an unexpected observation (problem), engineers first assess whether or not the problem is serious and requires diagnosis. Diagnosis normally proceeds by finding reasons or causes that impact on the observation. Once the causes are identified, then it is likely that solutions are required to prevent recurrences. The impact of making various hypothetical changes is likely to be assessed, along with the advantages and disadvantages of the various solutions proposed. This example demonstrates that the information needs for users in specific organizations are complex, requiring not only sophisticated retrieval processing but also the presentation of retrieval results in as natural a form as possible. Successful synthesis and presentation of such answers depend on the ability to compare information on a semantic level such that it produces a chain of semantic relations.

A prototype of semantic-based QA system implementing these ideas has been developed. The underlying approach is based on identifying various discourse relationships between two spans, such as *cause–effect* and *elaboration*. These types of relationship are derived from a computational linguistic theory known as rhetorical structure theory (RST). This theory defines a set of rhetorical relations and uses them to describe how the sentences are combined to form a coherent text (Mann & Thompson, 1988). As such, RST analysis discovers relationships within a sentence or among sentences. Because sentences are not usually comprehensible when isolated, this approach provides a more sophisticated content analysis. These annotations are then used to remove duplicate information and synthesize answers from multiple occurrences. Finally, these answers are justified by adding supporting information. As information is compared at the semantic level rather than at the string level, it is possible to determine whether a causal link exists between two events. This paper mainly addresses questions related to causal inference and describes a prototype system to test the ideas. The proposed system is targeted at the engineering area; however, the methodology is generic and can be applied to other domains.

2. LITERATURE REVIEW

Users find QA systems helpful as they do not need to go through each retrieved document to extract the information they need. Until recently, most QA systems only functioned on specifically created collections and on limited types of questions, but some attempts have been made to scale sys-

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

tems to open domains like the Web (Kwok et al., 2001). Experiments show that in comparison to search engines, for example, *Google*, QA systems significantly reduce the effort to obtain answers. AskJeeves (http://www.ask.com) and Brainboost (http://www.brainboost.com) are the examples of Internet search engines with a QA interface, but neither provides full-fledged QA capabilities. AskJeeves relies on hand-crafted question templates that enable automatic answer searches, and returns lists of documents instead of intelligently extracting brief answers. Brainboost supplies answers in plain English, but the correctness of its answers is limited to specific questions only, and for many questions neither relevant texts nor exact answers are found.

Currently, developments in QA have focused on improving system performance through more advanced algorithms for extracting exact answers (Voorhess, 2002). A project organized by the US National Institute of Standards and Technology has established benchmarks for evaluating QA systems. Two new QA system response requirements were introduced in 2002: to return an exact answer and to return only one answer. Previous requirements had allowed systems to return five candidate answers, and the answers could be between 50 and 250 bytes in length. This demonstrates that current QA systems are focusing on retrieving exact answers to factual questions. For these systems, performances of over 80% correct answers have been reported. However, user evaluations consistently highlight the fact that the usability is hindered by the absence of context information that would allow them to evaluate the trustworthiness of an answer. For example, user studies conducted by Lin et al. (2003) suggest that users prefer to see the answer in a paragraph rather than as an exact answer, even for a simple question like, *Who was the first man on the Moon?*

In comparison to open-domain QA systems, for example, Web, domain-specific QA systems have the following additional characteristics (Diekema et al., 2004; Hickl et al., 2004; Nyberg et al., 2005):

- a limited amount of data are available in most cases,
- domain-specific terminologies have to be dealt with, and
- user questions are complex.

Shallow text processing methods are mostly used for QA systems on the Web because of Web redundancy, which means that similar information is stated in a variety of ways and repeated in different locations.

However, in the engineering domain, suitable data can be scarce, and answers to some questions might only be found in a few documents, and these may exhibit linguistic variations from the questions. Therefore, intensive NLP techniques that can analyze unstructured texts using semantics and domain models are more appropriate. Domain ontologies and thesauri are required to define domain-specific terminologies. Hai and Kosseim (2004) used information in a manually created thesaurus to rank candidate answers by annotating the special terms occurring both in the queries and candidate answers. They also used a concept hierarchy for measuring similarities between a document and a query. Ontologies have also been used for expanding terms in the questions and clarifying ambiguous terms (Nyberg et al., 2005). Because ontologies can be regarded as storing information as triples, for example, person–work-for–organization, users can submit questions linked to such classes and relations in natural language (Lopez et al., 2005). For example, the question, *Is John an employee of IBM?*, can be answered by recognizing that John is a person, IBM is an organization, and employee is inferred from "someone who works for an organization." Questions other than factual ones need special attention, and a profile of the user can help to improve system performance (Diekema et al., 2004). Suitable ways of presenting answers and how much information should be provided must also be determined. To address these problems, some researchers proposed interactive QA. To that end, some QA systems rephrase the questions submitted to confirm whether or not users' information needs have been correctly identified (Lin et al., 2003). Advanced dialog implementations have also been suggested. However, Hickl et al. (2004) argue that the decomposition of user questions into simpler ones with which answer types are associated could be a more practical solution than a dialog interaction.

Generally, semantic annotations are treated as a similar task to named-entity (NE) recognition, which identifies domain concepts and their associations in a single sentence (Aunimo & Kuuskoski, 2005). This paper extends the notion of semantic annotation to include discourse relations that identify what information is generated from the extended sequences of sentences. This goes beyond the meanings of individual sentences by using their context to explain how the meaning conveyed by one sentence relates to the meaning conveyed by another. A discourse model is essential for constructing computer systems capable of interpreting and generating natural language texts. Such models have been used to assess student essays with respect to their writing skills, to summarize scientific papers, to extend the answers to a user' question with important sentences, and to generate personalized texts customized for individual reading abilities (Teufel, 2001; Williams & Reiter, 2003; Burstein et al., 2003; Bosma, 2005).

3. ENGINEERING TAXONOMY

Retrieval systems in engineering need to employ domain-specific terminologies that differentiate between specific and general terms. Specific terms are essential to understand users' questions and characterize documents in relation to those questions. Some general terms have specific meanings in engineering. For example, the term *shoulder* has multiple meanings in a dictionary, and in most cases, it means the part of the body between the neck and the upper arm. However, in engineering, it can refer to a locating upstand on a shaft. Domain taxonomies arrange such terms into a hierarchy. An example of an engineering taxonomy is

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

the engineering design integrated taxonomy (EDIT), and this taxonomy is used throughout this paper. It consists of four root concepts (Ahmed, 2005):

1. the design *process*, that is, a description of the different tasks undertaken at each stage of product development, for example, *conceptual design*, *detail design*, *brainstorming;*
2. the physical *product* to be produced, for example, assemblies, subassemblies, and components, using part-of relations, for example, *a motor and shaft of a motor;*
3. the *functions* that must be fulfilled by the particular component or assembly, for example, *one of the functions of a compressor disk is to secure the compressor blade and one of the functions of a cup is to contain liquid;* and
4. the *issues*, namely, the considerations, that a designer must take into account when carrying out a design process, for example, *considering the unit costs or production processes*.

A detailed description of the development of a generic methodology to develop engineering design taxonomies that was used for EDIT can be found in Ahmed et al. (2005).

4. THE PROPOSED METHOD

In general, a document can be encoded with various semantics, for example, *customer reviews or causal accounts of engineering failures*, and accessed by users who have very different interests. For example, in the case of product reviews by customers, negative and positive customer opinions are the main messages for market researchers. Conversely, designers are more interested in design-related issues, comments, and problems associated with engineering failures. It would therefore be beneficial to include those semantics that facilitate searching for information in a way that reflects the interests of the users. For example, for a designer whose task is to reduce fan noise, guidance on how to minimize aerodynamic noise should be retrieved. In contrast, if that designer is more interested in using a specific method for noise reduction, then documents describing the methods along with their advantages or disadvantages are more useful. With keyword-based indexing, it is not feasible to extract such semantics because most natural language texts have annotations that are too basic and no explicit descriptions of the concepts are available. Annotations are formal notes attached to specific spans of text. Their complexity and representation depend on the mark-up language used.

The proposed method works as follows:

1. a document is annotated with a set of relations derived from RST,
2. the document is classified with EDIT indexes,
3. the document is parsed using NLP indexing techniques,
4. the RST-annotated document is converted into predicate–argument forms for effective answer extraction, and
5. a user question is analyzed using the same NLP technique.

Steps 1, 2, and 3 can proceed independently, but step 1 must precede step 4.

4.1. Semantic annotations based on RST

Discourse analysis (DA) is crucial for constructing computer systems capable of interpreting and generating natural language texts. DA studies the structure of texts beyond sentence and clause levels, and structures the information extracted from the texts with semantic relations. It is based on the idea that well-formed texts exhibit some degree of coherence that can be demonstrated through discourse connectivity, that is, logical consistency and semantic continuity between events or concepts. This is in contrast with most keyword-based indexing that exclusively addresses the subsentence level, omitting the fact that sentences are interconnected to create a whole text. To establish a more robust and linguistically informed approach to identify important entities and their relations, a deeper understanding is necessary.

Annotating a text with a discourse structure requires advanced text processing, linguistic resources such as taxonomies, and possibly, manual intervention by experts. It certainly increases the work required to develop QA systems. However, if QA systems are only targeted at certain domains, where a limited number of texts has to be searched, and experts are available to assist, then detailed linguistic analysis is feasible. DA generates a discourse structure by defining discourse units, either at sentence or clause level, and assigning discourse relations between the units. Discourse structures can reveal various text features and attempts have been made to use them to identify important sentences that are key to understanding the contents of documents (Marcu, 1999; Kim et al., 2006*b*). Discourse structures can be used to compare units in multiple documents to evaluate similarities and differences in their meanings, as well as to detect anomalies, duplications, and contradictions.

Rhetorical relations are central constructs in RST and convey an author's intended meaning by presenting two text spans side by side. These relations are used to indicate why each of the spans was included by the author and to identify the nucleus spans that are central to the purpose of the communication. Satellite spans depend on the nucleus spans and provide supporting information. Nucleus spans are comprehensible independently of the satellites. For example, consider the following two text spans: *given that the clutch was functional*, and *it is unlikely that the engine was driving the starter*. A *condition* relation is identified, with span 2 being the nucleus. Satellite span 1 is only used to define the condition in which the situation in span 2 occurs.

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

These two spans are coherent because the person who reads them can establish their relationship.

Rhetorical relations between spans are constrained in three ways: constraints on a nucleus, constraints on a satellite, and constraints on the link between a nucleus and a satellite. They are elaborated in terms of the intended effect on the text reader. If an author presents an argument in a text that is identified as an *evidence* relation, then it is clear that the author was intending to increase a reader's belief in the claim represented in a nucleus span by presenting supporting evidence in a satellite span. Such relations are identified by applying a recursive procedure to a text until all relevant units are represented in an RST structure (Taboada & Mann, 2006). The procedure has to be recursive because the intended communication effect may need to be expressed in a complex unit that includes other relations. The results of such analyzes are RST structures typically represented as trees, with one top-level relation encompassing other relations at lower levels.

It is difficult to determine the correct number of relations to be used and their types. In the simplest domains only two relation types may be required, whereas some complex domains may require over 400 (Hovy, 1993). Hovy argued that taxonomies with numerous relation types represent subtypes of taxonomies with fewer types. Some relation types are difficult to distinguish, for example, *elaboration* and *example*. If there are too many types, inconsistencies of annotation are likely. If there are too few, it may not be possible to capture all the different types of discourse. Mann and Thompson (1988), for example, listed 33 relation types to annotate a wide range of English texts. To reduce inconsistencies of annotation, our method combines similar relation types and eliminates those that do not appear frequently. A preliminary examination with sample engineering domain data from aircraft incident reports (see Section 5.1) resulted in the following nine types: *background, cause–effect, condition, contrast, elaboration, evaluation, means, purpose*, and *solutionhood*. Each of them is described below, along with an example taken from the sample domain data, that is, aircraft incident reports, using (N) to indicate a nucleus span and (S) a satellite span.

4.1.1. Background

This type of relation is used to increase a reader's background understanding (S) of the nucleus span (N).

(S) *While the helicopter was approximately 25 feet above ground level en route to Tobin Lake, Saskatchewan, to pick up a bucket of water*

(N) *the engine fire warning light came on and the pilot saw smoke coming out of the engine cowling*.

4.1.2. Cause–effect

This type of relation is used to link the cause in the nucleus span to the to the effect in the satellite span or vice versa.

(N–S) *Analysis of the fuel hose indicates that the steel braid strands failed*

(S–N) *as a result of chafing*.

4.1.3. Condition

This type of relation is used to show the condition (S) under which a hypothetical situation (N) might be realized.

(S) *Given that the clutch was functional*,

(N) *it is unlikely that the engine was driving the starter*.

4.1.4. Contrast

This type of relation is used to contrast incompatibilities between situations, opinions, or events, and there is no distinction between (N) and (S).

(N–S) *It is considered likely that the fire was momentarily suppressed*

(N–S) *but because of the constant supply of fuel and ignition, it reignited after the retardant was spent*.

4.1.5. Elaboration

This type of relation is used to elaborate (S) on the situation in (N).

(N) *The variable inlet guide vane actuator (VIGVA) hose, which provides fuel pressure to open the variable guide vanes, was found pinched between the top of the starter/generator and the impeller housing assembly*.

(S) *Further inspection of the pinched fuel hose revealed a hole through the steel braiding and inner lining*.

4.1.6. Evaluation

This type of relation is used to provide an evaluation (S) of the statement in (N).

(N) *The second option, the engine start valve master switch*,

(S) *does not provide a positive indication to the flight crew of the start valve operation*.

4.1.7. Means

This type of relation is used to explain the means (S) by which (N) is realized.

(N) *The rest of the fire was extinguished*

(S) *using a fire truck that arrived on the site.*

4.1.8. Purpose

This type of relation is used to describe the purpose (S) achieved through (N).

(N) *At 13.5 flight hours prior to the occurrence, the starter/generator had been removed*

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

(S) *to accommodate the replacement of the starter/ generator seal, then reinstalled.*

4.1.9. Solutionhood

This type of relation is used to link the problem (S) with the solution (N).

(N) *The number 2 engine start control valve and starter were replaced, and the aircraft was returned to service.*

(S) *It was determined that the number 2 engine starter had failed.*

Figure 2 shows a screenshot of the RST annotation of the sample domain data. A software tool, RSTTool, is used to complete the annotation (O'Donnell, 2000). RSTTool offers a graphical interface with which annotators segment a given text into text spans and specify relation types between them. A computer program written in Perl by the first author has been developed to automatically extract the RST annotations stored by RSTTool. The box at the bottom of Figure 2 shows the decomposed text spans with the individual spans identified by brackets. The top shows the corresponding RST analysis tree.

It is common to use discourse connectives (or cue phrases) for automatic discovery of discourse relations from texts. For example, by detecting the word *but*, a *contrast* relation between two adjacent texts can be identified. This approach is easy to implement but can lead to a low coverage, that is, the ratio of correctly discovered discourse relations to the total number of discourse relations. A study by Taboada and Mann (2006) showed that the levels of success using cue phrases ranged from 4% for the *summary* relation to over 90% for the *concession* relation. To improve the coverage, machine learning methods have been used. Marcu and Echihabi (2002) used Naive Bayesian probability to generate lexical pairs that can identify relation types without relying on cue phrases. For example, the approach can extract a *contrast* when one text contains *good* words and another *bad* words, even when *but* does not appear. Whereas this approach produces a good performance, the assumption that the lexical pairs are independent of each other can lead to a considerable number of training sentences being required, sometimes over 1,000,000. Although a low presence of cue phrases can lead to many undiscovered relations, they can serve as a reference for annotators. Discourse text spans are inserted at every period, semicolon, colon, or comma and at every cue phrase listed in Table 1. Annotators first refer to the cue phrases to test whether the corresponding relation types can be used for a given text. If no direct match is identified, then they select the closest one using their judgement. Table 1 summarizes cue phrases extracted from Knott and Dale (1995) and Williams and Reiter (2003).

RST-annotated texts are converted into a predicate–argument structure, that is, *predicate* (tag_1:*argument*$_1$, . . . ,

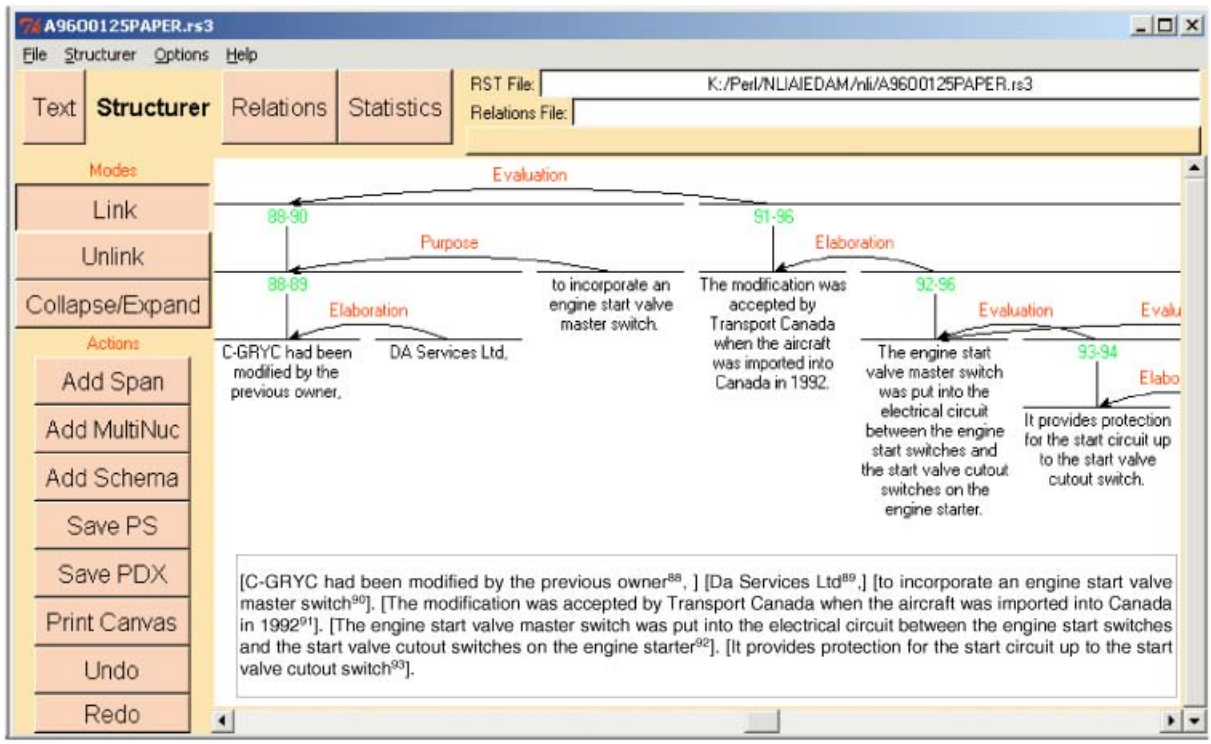

Fig. 2. A screenshot of the rhetorical structure theory (RST) analysis using RSTTool. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

Table 1. *Cue phrases for identifying relation types*

Relation Types	Cue Phrases
Background	With, probably
Cause–effect	Because, as, as a consequence, as a result, thus, therefore, due to, lead to, consequently
Condition	As long as, if–then, if, so long as, unless, until
Contrast	Although, by contrast, even though, however, though, whereas, while
Elaboration	Also, in addition, in particular, for example, in general
Evaluation	With, so, but, which, even so
Means	By, with, using
Purpose	In order to, for the purpose of
Solutionhood	Proposed solution, options

tag_n:$argument_n$). Predicates represent the main verbs in sentences and tags include subjects, objects, and prepositions. For example, consider the following sentence: *Analysis of the fuel hose indicates that the steel braid strands failed as a result of chafing*. For this sentence the *evidence* relation type is used to annotate it as follows: *evidence((indicate (subject:analysis of the fuel hose)), (fail(subject:the steel braid strand, pp:as a result of chafing)))*.

4.2. Semantic-based QA description

4.2.1. Term indexing

In general, it is difficult to extract good index terms because of inherent ambiguity in natural language texts. A term in a text, that is, an alphanumeric expression, can have different meanings depending on the domain in which it is being used, and a term can appear more frequently in one domain than in another. Publicly accessible dictionaries, for example, WordNet (Miller et al., 1993) are good resources for obtaining the meanings of the terms, both manually and automatically. For example, according to WordNet, *blade* has nine meanings. One definition is *especially a leaf of grass or the broad portion of a leaf as distinct from the petiole*. However, another in the engineering domain is *flat surface that rotates and pushes against air or water*. Terms can also be used in different domains with the same meaning. For example, *certification* does not have a different meaning in the engineering domain. Most keyword-based search systems index a document with a list of keywords ranked with relevance weightings. Whereas these keywords might be sufficient to describe superficially the contents of a document, it is difficult to interpret the true message if their precise meanings are not established.

In contrast, NLP produces a rich representation of a document at a conceptual level. To achieve humanlike language processing, NLP includes a range of computational techniques for analyzing and representing natural texts at one or more levels of linguistic analysis (Liddy, 1998). It is common to categorize such techniques into the six levels listed below, each of which has a different analysis capability and implementation complexity (Allen, 1987). The application of NLP on a text can be implemented at the simplest level, for example, morphological level and then extended into a full-fledged pragmatic analysis that shows a superior understanding, but requires large resources and extensive background information. In this paper, NLP processing includes the first five levels and excludes the pragmatic level:

- *morphological level*: component analysis of words, including prefixes, suffixes and roots, for example, *using* is stemmed into *use*;
- *lexical level*: word level analysis including a lexical meaning and a part of speech (POS) analysis, for example, *apple* is a kind of fruit and is tagged as noun;
- *syntactic level*: analysis of words in a sentence in order to determine the grammatical structure of the sentence;
- *semantic level*: interpretation of the possible meanings of a sentence, including the customization of the meanings for given domains;
- *discourse level*: interpretation of the structure and the meaning conveyed from a group of sentences; and
- *pragmatic level*: understanding the purposeful use of language in situations particularly those aspects of language, which require world knowledge.

Figure 3 shows the steps of the indexing process. The text in the box at the bottom of Figure 2 is used as an example.

Step 1. Preprocessing: One paragraph is identified in the example text, which is then decomposed into four sentences. The first sentence is the following:

C-GRYC had been modified by the previous owner, DA Services Ltd, to incorporate an engine start valve master switch.

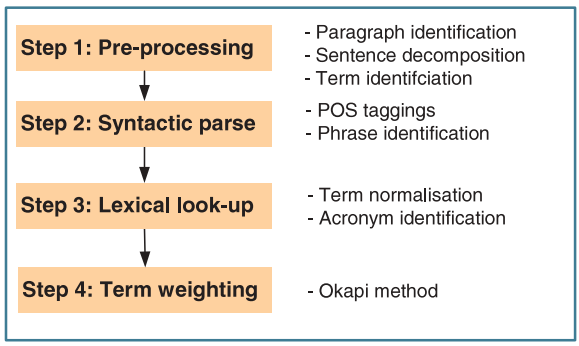

Fig. 3. The steps of the indexing process. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

Terms are identified as words lying between two spaces including a period.

Step 2. Syntactic parse: The apple pie parser (Sekine & Grishman, 2001) is used for a syntactic parse that tags POS and identifies phrases. POS identifies not what a word is, but how it is used. It is useful to extract the meanings of words because the same word can be used as a verb or a noun in a single sentence or in different sentences. In a traditional grammar, POS classifies a word into eight categories: verb, noun, adjective, adverb, conjunctive, pronoun, preposition, and interjection. The apple pie parser refers to the grammars defined in the Penn Treebank to determine the POSs (Marcus et al., 1993). For example, the first word *C-GRYC* is tagged as *NNPX*, that is, proper single noun. The remain POSs for the sentence above are shown below:

POS taggings: *C-GRYC/NNPX had/VBD been/VBN modified/VBN by/IN the/DT previous/JJ owner/NN DA/NNPX Services/NNPS Ltd/NNP to/TOINF incorporate/VB an/DT engine/NN start/NN valve/NN master/NN switch/NN.*

Phrase identification groups words grammatically, for example, into noun phrases such as {*the previous owner DA Services Ltd*} and {*an engine start valve master switch*}.

Step 3. Lexical look-up: Each POS-tagged word is compared with WordNet definitions to achieve term normalization. Acronym identification extends an acronym found in a text fragment with its full definition. An example of term normalization is

$$\text{modified} \rightarrow \text{modify}$$

and an example of acronym identification is

$$\text{DA} \rightarrow \text{Dan-Air.}$$

Step 4. Term weighting: Although it is possible to analyze the full contents of a document, this becomes computationally expensive when the documents are large. For an effective retrieval, it is desirable to extract only those portions of a document that are useful and to transform them into special formats. Text indexing determines the central properties of the content of a text to differentiate relevant portions of text from irrelevant ones. The quality of each index term is evaluated to determine if it is an effective identifier of the text content. A relative importance weighting is then assigned to each index term. A common approach is to index a document divided into paragraph-sized units. In this paper, the Okapi algorithm is used (Franz & Roukos, 1994; Robertson et al., 1995). It weights (w_{jk}) a term (t_k) in a paragraph (P_j) as follows:

$$w_{jk} = \frac{c_{jk}}{0.5 + 1.5 * \dfrac{\text{len}(P_j)}{\text{ave\_len}} + c_{jk}} * \frac{\log(N - n + 0.5)}{n + 0.5}, \quad (1)$$

where c_{jk} is the frequency of the t_k, N is the total number of paragraphs in a data set, n is the number of paragraphs that have contents containing the t_k, $len(P_j)$ is the total number of frequencies of all terms presented in a P_j, and ave_len is the average number of terms per paragraph. Using the term weighting method, the example sentence is stored into a vector model, that is, each term is associated with its calculated weighting.

4.2.2. Domain knowledge in QA

An engineering taxonomy such as EDIT is a useful means to identify domain-specific terms in a document. The successful extraction of domain-specific terms can improve the accuracy of QA. For example the answer to the question, *What material should be used for this turbine blade?*, is more easily identified if *Titanium* is marked up as a type of material. Among the four root concepts defined in EDIT, only two are used: issue and product. These two concepts exhibit different characteristics. According to Ahmed (2005), issue root concepts are considerations designers must take into account when carrying out a design process. These can be the descriptions of problems arising during a product's lifecycle or new design requirements to be satisfied. In contrast, product root concepts comprise a hierarchy list of product names, decomposing an overall technical product or system into smaller elements. Different techniques are therefore needed to handle them in the documents. For issue concepts, any technique that automatically classifies a document into predefined categories is suitable. For product concepts, the technique of NE recognition is used. In the QA method proposed in this paper, the techniques developed by Kim et al. (2006*a*, 2006*b*) are used. The technique for the classifying issue concepts is described in Kim et al. (2006*b*) and the one for classifying product concepts, using probability-based NE identifiers, is described in Kim et al. (2006*a*).

4.2.3. QA overview

Figure 4 shows the overall architecture of the proposed QA system. State-of-the-art QA systems can achieve an accuracy of up to 80%, as demonstrated by recent tests undertaken using TREC data sets, which mainly consist of newspaper documents (Voorhess, 2002). However, this level of performance is not expected to be repeated in other environments. The questions in the above tests were carefully constructed, that is, no misspellings, and they were mostly factual and based on a single interaction with a user, so no dialogue.

The prototype system proposed in this paper does not aim at achieving better accuracy in question analysis or in finding answers. Instead, its main objective is to demon-

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

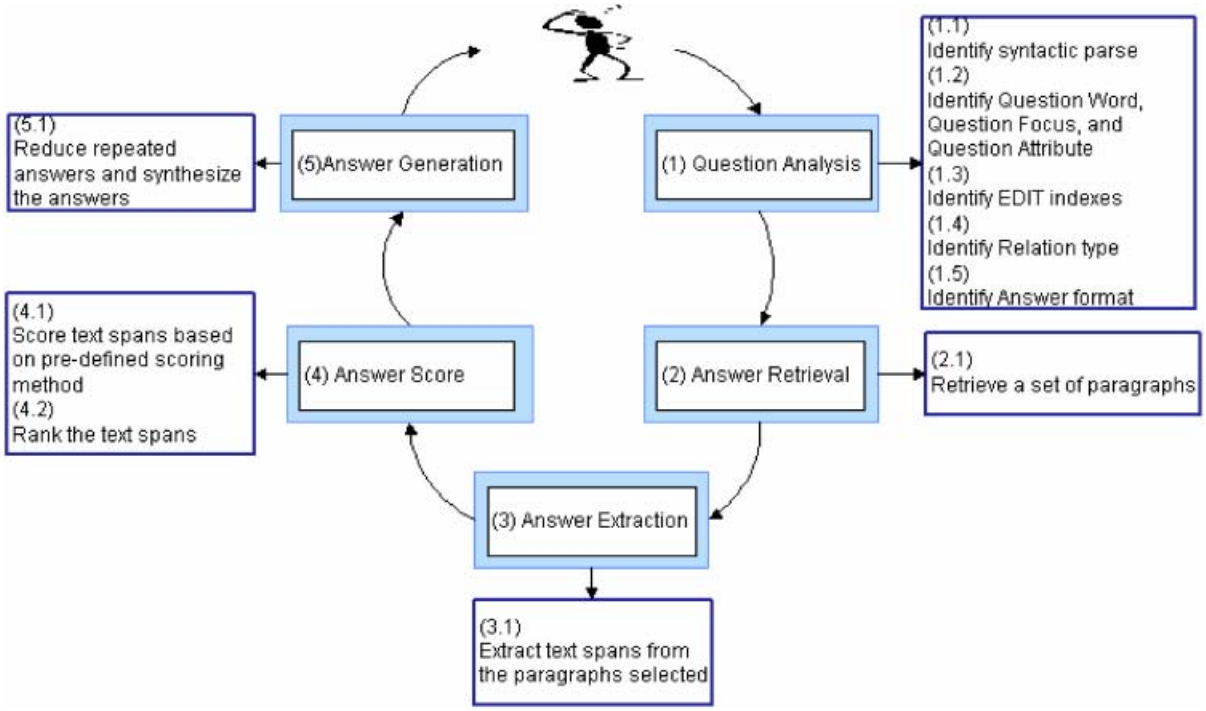

Fig. 4. The overall architecture of the proposed QA system. [A color version of this figure can be viewed online at www.journals.cambridge.org]

strate the efficiency of RST-based annotations for coherent answer generation (see step 5 in Figure 4). Each of the steps in Figure 4 are now described.

STEP 1. Question analysis: The question analysis module decomposes a question into three parts: question word, question focus, and question attribute. The question word indicates a potential answer type, for example, *where, when*, and so forth. The question focus is a word, or a sequence of words, that describe the user's information needs that are expressed in the question. The question attribute is the part of the question that remains after removing the question word and the question focus. It is used to rank candidate answers in a decreasing order of relevance. An example is given below.

Question: *What were the consequences of the vibration of the starter/generator?*

(1.1) Syntactic parse
POS: *What|WP were|VBD the|DT consequences|NNS of|IN the|DT vibration|NN of|IN the|DT starter|NN /|SYM generator|NN* Phrase identification: NPL{*the consequences*} PP{*the vibration of the starter/generator*}

(1.2) Question Word {*what*}, Question Focus {*consequences*}, Question Attribute {*the vibration of the starter/generator*}

(1.3) EDIT indexes: ⟨Product concept='Starter_Ducting'⟩ *starter* ⟨Product concepts='Electrical_Generator'⟩ *generator* ⟨Issue concepts='Vibrations'⟩ *vibration*

(1.4) Relation type: effect

(1.5) Answer format: cause–effect(Question Attribute, ⟨Answer⟩)

The question focus, that is, consequences, for the example question above is matched with the *effect* in the *cause–effect* relation type. Therefore, the possible answers should be the effects of the events described in the question attribute. For an automatic matching, a semantic similarity between the question focus and the relation type is computed using the method proposed by Resnik (1995). This method is based on the number of the edges in a semantic hierarchy, for example, *WordNet*, encounted between two terms when locating them in the hierarchy.

STEP 2. Answer retrieval: The answer retrieval module uses the question attribute identified by the question analysis module to select paragraphs that might contain candidate answers. A cosine-based similarity calculation is used for ranking the selected paragraphs in order of relevance to the keywords that appear in the question attribute (Salton, 1989).

$$\mathrm{sim}(q, p_j) = \frac{\sum_{i=1}^{t} w_i * w_{ij}}{\sqrt{\sum_{i=1}^{t} w_i^2} * \sqrt{\sum_{i=1}^{t} w_{ij}^2}}, \quad (2)$$

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

where p_j is a given paragraph, q is a question attribute, w_{ij} is the weight of the term t_i in the p_j. The similarity value is normalized by the total weights of common words.

STEPS 3 and 4. Answer extraction and answer scoring: The answer extraction module examines the paragraphs selected in the answer retrieval module in order to select text spans from which candidate answers can be extracted. The EDIT indexes, question focus, and question attribute are used to determine whether the text spans contain the answer. In doing so, it is necessary to measure how well they are related to the question. An overall similarity between the text spans and the question is computed by summing following three similarity scores: the score reflecting whether a given text span is classified with the EDIT indexes, the score reflecting whether a given text span contains the question focus, and the score reflecting the degree of similarity between a given text span and the question attribute. They are summed as follows:

$$s(ts_{ij}) = \alpha * s\_edit(ts_i) + \beta * s\_rst(ts_i) + (1 - (\alpha + \beta)) * s\_attr(ts_i), \quad (3)$$

where $s(ts_{ij})$ is the score of the text span ts_i in the p_j, $\alpha (0 \leq \alpha \leq 1)$ and $\beta (0 \leq \beta \leq 1)$ are used to normalize the score $s(ts_{ij})$ to lie between 0 and 1, and $s\_edit(ts_i)$ is defined as

$$s\_edit(ts_i) = \sum_{k=\text{positions}} s(ind_{ki}) \Big/ N, \quad (4)$$

where $s(ind_{ki})$ is a Boolean indicating whether a given text span is classified with an EDIT index number, ind_{ki}, *positions* is the set of matches against the EDIT indexes returned by the question analysis module, N is the total number of elements in this set, $s\_rst(ts_i)$ is a Boolean variable that is true if the RST annotation for the text span matches the annotation for the question, and $s\_attr(ts_i)$ is defined as

$$s\_attr(ts_i) = \sum_{m=1}^{n} s(t_{mi}) \Big/ M, \quad (5)$$

where $s(t_{mi})$ is a Boolean indicating whether or not a given term, t_m in the text span, ts_i is matched with a term in the question attribute t_m, and M is the total number of terms in the question attribute. The scored text spans are sorted in decreasing order by value and those above a predefined threshold selected.

STEP 5. Answer generation: For coherent answer generation, duplicate sentences or clauses should be removed. Two sentences or clauses are recurrent if they are exactly equivalent or if they differ only in the level of generality. For example, the following two clauses are equivalent: *the starter had failed* and *the failure of the number 2 engine starter*. In contrast, sentence 1 below can be replaced by sentence 2 without loss of information because sentence 2 subsumes the information in sentence 1.

Sentence 1: *It is probable that a short circuit in the engine wiring harness allowed the number 2 engine start valve to reopen, causing the number 2 engine starter to over speed and subsequently fail.*

Sentence 2: *It is probable that a short circuit in the engine wiring harness allowed the number 2 engine start valve to reopen, causing the number 2 engine starter to over speed and subsequently fail, resulting in an engine fire.*

Automatic text summarization systems employ various approaches to compare similar sentences that have different wordings (Mani & Maybury, 1999). In general, these systems use the following two steps to produce summaries from a document:

Step 1 identifies and extracts important sentences to be included in a summary.

Step 2 synthesizes the extracted sentences to form a summary.

There are two common methods of synthesis in step 2. The nonextractive summary method suppresses repeated sentences either by extracting a subset of the repetitions or by selecting common terms. It then reformulates the reduced number of sentences to produce the summary (Barzilay et al., 1999; Jing & McKeown, 2000). The extractive summary method focuses on the extraction of important sentences and assembles them sequentially to produce the summary. The objective of these automatic summarization systems is to create a shortened version of an original text to reduce the time spent reading and comprehending it. In contrast, the objective of our proposed approach is to extend and synthesize text spans to allow the generation of coherent answers.

In the proposed approach, two text spans are compared to determine whether or not both are similar using Eq. (2). Text spans that have higher similarity values than the predefined threshold are excluded. The algorithm of the answer generation module is shown below as pseudocode.

Variable definition:

answerList: chains of "cause and effect" to be generated by the answer generation module

textspanList: a list of text spans returned by the answer scoring module

relationlinkedList: a list of text spans linked through the "cause and effect" relation obtained from the RST annotations.

t: one text span being examined and extracted from the textspanList

thresh: a predefined threshold used to compare similarity between two text spans

t1: temp variable, t2: temp variable

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

```
Initialization:
        answerList ← empty
Repeat (
        t ← retrieve(textspanList), thresh ← 0.8, t1 ← empty, t2 ← empty.
        IF (answerList does not contain t)
        THEN (build chains of "cause and effect" for t and merge them with answerList
                {
                        foreach t1 ← relationlinkedList(t)
                        {
                                foreach t2 ← answerList
                                        {       compute similarity between t1 and t2
                                                IF (similarity > thres) { NOTHING }
                                                ELSE { update(answerList,t1) }
                                                }
                        }
                }
        ELSE { NOTHING }
        remove(textspansList,t)
        t ← retrieve(textspansList)
        )
Until (textspanList is EMPTY)
```

5. PILOT STUDY

This section presents a preliminary evaluation of the prototype QA system. The evaluation tests the following two hypotheses: the proposed system is efficient at extracting and presenting answers to causal questions using relation types, and the presentation of the synthesized answers helps users to understand the retrieved results. The first hypothesis was evaluated by comparing the performance of the prototype system against that of a standard QA system. The standard system is different from the prototype system in the following ways:

STEP 1. The answer format is revised as ⟨Answer⟩ question focus question attribute. This format indicates that the system should find text spans that have syntactic variations with the question focus and semantic similarities with the question attribute. Using the example question in Section 4.2.3, a potential answer text span can be ⟨Answer⟩ is the consequence of ⟨Question Attribute⟩.

STEP 2. Not revised.

STEPS 3 and 4. Equation (3) is revised as $s(ts_{ij}) = s\_attr(ts_i)$.

STEP 5. Not used.

With the standard system, multiple instances were extracted without synthesizing them. This comparison examined whether the answer generation method described in Section 4.2 avoids repeated information and generates coherent answers. The second hypothesis was evaluated by measuring user performance in a simple trial.

5.1. Trial data set

For the trial, three official aircraft incident investigation reports were downloaded from the following websites:

1. http://www.tsb.gc.ca/en/reports/air/1996/a96o0125/a96o0125.asp
2. http://www.tsb.gc.ca/en/reports/air/2002/A02C0114/A02C0114.asp
3. http://www.aaib.gov.uk/sites/aaib/cms_resources/dft_avsafety_pdf_029538.pdf

Although the incidents happened to three different aircraft types (Boeing 727-217, Bell 205A-1 helicopter, and Boeing 737-8AS), the incidents share a common cause, that is, an in-flight engine fire. Although the reports were written by different incident investigation teams, they share a broadly similar terminology, for example, *emergency landing, engine starter valve*, and so forth. After removing embedded HTML tags and images, the average document length was 1820 words, or 24 paragraphs. Each document was first indexed as described in Section 4.2.1, and RST annotations were applied by the first author using RST-Tool. A total of 194 relations were annotated. The total numbers for each relation type were 20 background, 45 cause–effect, 16 condition, 30 contrast, 32 elaboration, 24 evaluation, 10 means, 12 purpose, and 5 solutionhood.

5.2. The trial

Six engineering graduate students and two members of the Engineering Department staff of the University of Cam-

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

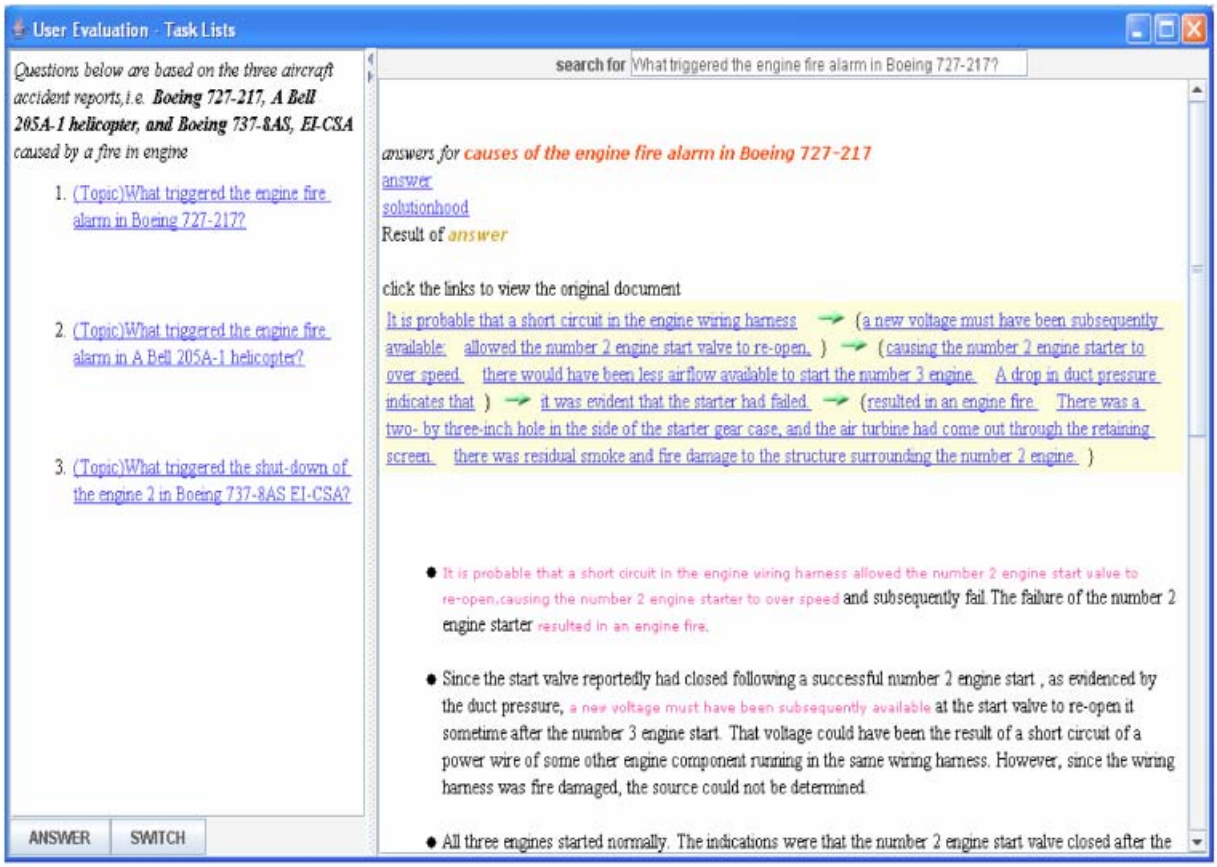

Fig. 5. An example screenshot of the proposed system. [A color version of this figure can be viewed online at www.journals.cambridge.org]

bridge participated in the trial. A brief introduction to the trial and trial data set was given to the participants. Each participant was asked to answer multiple questions and their performance and accuracy were measured. The trial consisted of reviewing the answers to three questions, that is, one for each incident report. These answers were split into two groups. The first group was extracted and synthesized using the prototype QA system and the second was extracted using a standard QA system. For a fixed period of time, the participants were instructed to read the answers on-line for both systems (see Fig. 5), and follow links to the original document if desired. After this, a further list of questions related to the answers the users had just read was given to the users. Their answers to these questions were used to test their understanding of the answers they had just seen. To avoid the evaluation problems caused by the inclusion of incorrect answers, both groups of answers were examined in order to verify that they were all true.

Table 2 shows the three initial questions, along with the associated questions that were used to test the users' understanding.

5.3. Trial results

Answers to the three original questions shown in Table 2 were extracted and synthesized using the method described in Section 4.2. The answers then were compared to the set of answers prepared in Section 5.2. The threshold for the answer retrieval module, that is, the value for Eq. (2), was set as 0.5, meaning that the paragraphs that had run s2 cosine similarity values over 0.5 were selected. The values for α and β specified in Eq. (3) were set as 0.3 and 0.2, respectively. The threshold for the answer scoring module, that is, the values for Eq. (3), was set as 0.2.

The results are shown using Table 3 and Table 4. Table 3 summarizes the results of the "cause and effect" chains generated by the proposed QA. Table 4 compares the performance of the proposed QA on retrieving correct text spans with that of the standard QA. Precision and recall were used to measure the performance. In this paper, precision is defined as the percentage of the retrieved text spans that are identified as right among the total number of retrieved text spans. Recall is the percentage of the retrieved right text spans among the total number of right text spans.

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

Table 2. *The three original questions and their associated questions*

Question 1: What triggered the engine fire alarm on the Boeing 727-217?
1. On which engine of the Boeing 727-217 was the fire alarm observed?
2. What were the consequences of the failure of the number 2 engine starter?
3. Why did the number 2 engine starter overspeed?
4. Did the starter valve of the number 2 engine close after the engine was started?
5. Why did the number 2 engine starter valve reopen?
6. How can we determine if the starter valve is open?
Question 2: What triggered the engine fire alarm on the Bell 205A-1 helicopter?
1. Why did the starter/generator start to vibrate?
2. What were the consequences of the vibration of the start/generator?
3. Why was the hold-down nut at the 12 o'clock position left out?
4. Was the engine fire alarm activated due to the abrasion of the cooling fan?
Question 3: What triggered the shut-down of engine 2 on the Boeing 737-8AS?
1. Does this incident have the same engineering problem as the Boeing 727-217?
2. Did the failure of the number 4 bearing in the number 2 engine contribute to the event?
3. What were the consequences of the presence of engine vibration?

The second column in Table 3 specifies the number of paragraphs returned by the answer retrieval module and the third one specifies the number of text spans returned by the answer extraction and scoring modules. The fourth one specifies the depth of the chains, that is, the number of cause and effect nodes along the longest path from the root node down to the farthest leaf node in the chains. For example, for Question 1, one "cause and effect" chain with a depth of 4 was generated by synthesizing and extending the 12 text spans extracted from the 17 paragraphs.

The following are examples of correct and incorrect text spans for question 1.

5.3.1. *Correct text spans*

1. *The failure of the number 2 engine starter resulted in an engine fire.*
2. *The hazard associated with an engine fire caused by a starter failure was recognized and addressed in AWD 83-01-05 R2.*
3. *It is probable that a short circuit in the engine wiring harness allowed the number 2 engine start valve to reopen, causing the number 2 engine starter to over speed and subsequently fail, resulting in an engine fire.*

5.3.2. *Incorrect text spans*

1. *Because of the engine's proximity to the elevator and rudder control systems, a severe in-flight fire in the number 2 engine is potentially more serious than a fire in either the number 1 or 3 engine.*
2. *Fire damage to the engine component wiring precluded any significant testing of the wiring harness.*
3. *Two fire bottles were discharged into the number 2 engine compartment; however, the fire warning light remained on.*

As shown in Table 4, on average, the proposed QA achieved 76% precision and 86% recall when retrieving text spans for three questions. In contrast, the standard QA achieved 57% precision and 58% recall. This suggests that the proposed QA has considerable potential for extracting and synthesizing answers to causal questions. The task of retrieving text spans is similar to the sentence selection task in automatic text summarization systems.

Table 3. *The overview of cause and effect chains generated by the proposed questions and answers*

	Number of		Depth of Chains
	Paragraphs	Text Spans	
Question 1	17	12	4
Question 2	8	9	4
Question 3	11	6	2

Table 4. *Comparison of two question and answer systems for the task of retrieving correct text spans*

	Standard		Proposed	
	Precision	Recall	Precision	Recall
Question 1	0.45	0.5	0.67	1
Question 2	0.6	0.67	0.78	0.78
Question 3	0.67	0.57	0.83	0.79
Average	0.57	0.58	0.76	0.86

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

The text summarization systems referred to earlier in this paper are by Barzilay et al. (1999) and by Jing and McKeown (2000). In the context of multidocument summarization, Barzilay et al. (1999) focused on the generation of paraphrasing rules that were used to compare semantic similarity between two sentences. They tested the rules for the task of identifying common phrases among multiple sentences. The automatically generated common phrases were then reviewed by human judges. The reviews identified 39 common phrases and among them the system correctly identified 29 of them. In addition, the identified phrases contained 69% of the correct subjects and 74% of the correct main verbs. On average, the system achieved 72% accuracy.

Jing and McKeown (2000) carried out three evaluations. The first tested whether the automatic summarization system could identify a phrase in the original text that corresponds to the selected phrase in a human–written abstract. When tested with 10 documents, the automatic system achieved 82% precision and 79% recall on average. The second evaluation tested whether the automatic system could remove extraneous sentences, that is, sentence reduction. The result showed that 81% of the reduction decisions made by the system agreed with those of humans. The third evaluation tested whether the automatic system could generate coherent summaries. The system achieved 6.1 points out of 10, that is, 61% accuracy for generating coherent summaries. Only the first evaluation focused on the sentence selection.

The performance of our proposed QA when retrieving correct text spans with 76% precision and 86% recall is slightly better than the work by Barzilay et al. (1999), which is 72% accuracy, and comparable to the work by Jing and McKeown (2000), which is 82% precision and 79% recall.

On average, out of 13 questions, the users in the first group, that is, those who read the answers given by the proposed QA, incorrectly answered *two* questions, whereas the users in the second group incorrectly answered *five* questions. On average, the users in the first group completed the trial within *19* min, and the users in the second group completed the trial within *25* min. *Five* of the 13 questions were correctly answered by all the users in the first group, whereas just *one* question was correctly answered by all the users in the second group. All users in the second group incorrectly answered question number 6: *How can we determine if the starter valve is open?*

Although the preliminary results are encouraging, it is difficult to draw firm conclusions from this trial for the following reasons: the low number of users in the two groups; and the number of causal relations in the trial data set were small. Users in the first group expressed the opinion that the synthesized chains of "cause and effect" description were helpful in understanding the causes of the three incidents.

6. CONCLUSION AND FURTHER WORK

Researchers in computational linguistics have speculated that the relation types defined in RST can improve the performance of QA systems when answering complex questions. The class of causal reasoning questions, either predictive or diagnostic, is one that we have shown might be better answered using these relation types. The reason for this is that the majority of causal questions can be answered in multiple ways, that is, it is difficult to pinpoint particular causes and regard them as independent of the remaining information. Generally, identifying the causes of a specific event involves creating chains of "cause and effect" relations. Without a deep understanding of all the relevant information contained in a document, it is not possible to derive such causal chains automatically. It is still not known how users would like such causal chains to be presented, and it is not suggested that the interface proposed in this paper is necessarily the best. The contribution of this paper is the demonstration of a method for synthesizing causal information into coherent answers. The source information can be scattered over different parts of a single document or over multiple documents. The pilot study indicated that the proposed QA was more efficient at extracting and synthesizing answers when compared with standard QA, that is, 19 percentage points increased precision and 28 percentage points increased recall. The pilot study also indicated that the synthesized chains of "cause and effect" descriptions were helpful not only for quickly understanding the direct causes of the three incidents but also for being aware of related contexts along with the rationales for the causes of the incidents.

The main objective is to improve the understanding of the answers generated by QA systems. An answer is considered to be coherent if duplicate expressions are eliminated and if it is appropriately extended with additional information. This additional information should help users verify the answers and increase their awareness of relevant domain information. Using RST annotations, it has been shown that it is feasible to compare and integrate the information at a semantic level. This leads to a way of presenting answers in a more natural manner. A pilot trial demonstrated that the answers generated by the prototype QA system led to more rapid and improved understanding of those answers.

Further work is planned with the aim of improving the performance of the prototype system in three ways. First, because engineers have varying levels of domain expertise, the system should consider the preferences and profiles of individuals. Inexperienced engineers might have very broad information requests and prefer to explore the domain, whereas experienced engineers might have detailed information requests aimed at refining their existing knowledge. Novice engineers require more background information, probably assembled using *elaboration* or *background* relation types.

https://doi.org/10.1017/S0890060407070205 Published online by Cambridge University Press

Second, synthesizing sentences extracted from different documents is crucial to generate answers that are longer than one sentence. When writing a sequence of linked sentences, authors often replace noun phrases by pronouns, or shortened forms of the phrase, in subsequent sentences, for example, *the number 2 engine starter* is replaced by *it* or *the starter*. Coreference (anaphora) resolution is a process for determining the multiple representations of a noun phrase and is a key issue in computational sentence synthesis. However the main focus of research in this area has been on the resolution of personal pronouns, for example, *he*, *him*, *his*, and so forth. Various techniques have been proposed for automatic coreference identification, and it is planned to extend the prototype QA system by adapting these techniques.

Third, the crucial issue of automatic RST annotation will be addressed because this is essential for the practical application of the system. Kim et al. (2004) have applied a machine learning algorithm, that is, inductive logic programming, to analyze documents created using the design rationale editor (DRed). This enabled the automatic identification of the relation types (Bracewell & Wallace, 2003, Bracewell et al., 2004). Tests have demonstrated approximately 80% accuracy. This high figure can be attributed partly to the structure of the DRed documents in the data set. These documents are carefully structured using an argumentation model derived from that of IBIS (Kunz & Rittel, 1970). The documents comprise linked textual elements of a predefined set of types. These element types include "issue," "answer," and "argument." The links between them are directed but untyped. This algorithm will be extended to deal with other types of documents, for example, Web pages and unstructured texts.

The main objective of this research was to answer more complex questions than current QA systems are capable of answering. There are five modules in the architecture of the proposed QA system: question analysis, answer retrieval, answer extraction, answer score, and answer generation. The question analysis module analyzes the question in terms of the question word, question focus and question attribute. The next three modules retrieve, extract, and score answers from documents that have been manually annotated and semiautomatically indexed. The manual annotation is based on nine of the 33 relation types defined in RST. The semiautomatic indexing uses the issue and product root concepts of the EDIT engineering taxonomy. The main contribution of this research lies in the fifth module. This module synthesises causal information into coherent answers, drawing information from both different parts of a single document and from multiple documents. A prototype implementation shows promise, but additional testing is required. Further developments are proposed that will allow the system to take into account the preferences and profiles of users, extend the system to include coreference identification, and eliminate the manual annotation of documents. As with all computer support systems, the interface is critical and here further empirical research is needed.

ACKNOWLEDGMENTS

This work was funded by the University Technology Partnership for Design, which is a collaboration between Rolls-Royce, BAE SYSTEMS, and the Universities of Cambridge, Sheffield, and Southampton. We thank S. Banerjee and T. Pedensen for the software implementing the similarity measurement method proposed by Resnik.

REFERENCES

80-20 Software. (2003). 80-20 Retriever Enterprise Edition. Accessed at http://www.80-20.com/brochures/Personal Email Search Solution.pdf

Ahmed, S. (2005). Encouraging reuse of design knowledge: a method to index knowledge. *Design Studies Journal 26(6)*, 565–592.

Ahmed, S., Kim, S., & Wallace, K.M. (2005). A methodology for creating ontologies for engineering design. *Proc. ASME 2005 Int. Design Engineering Technical Conf. Computers and Information in Engineering*, Paper No. DETC 2005-84729.

Allen, J. (1987). *Natural Language Understanding*. London: Benjamin/Cummings.

Aunimo, L., & Kuuskoski, R. (2005). Question answering using semantic annotation. *Proc. Cross Language Evaluation Forum (CLEF)*.

Barzilay, R., McKeown, K.R., & Elhadad, M. (1999). Information fusion in the context of multi-document summarization. *Proc. Annual Computational Language*, pp. 550–557.

Bosma, W. (2005). Extending answers using discourse structure. *Proc. Workshop on Crossing Barriers in Text Summarization Research in RANLP*, pp. 2–9.

Bracewell, R.H., & Wallace, K.M. (2003). A tool for capturing design rationale. *Proc. 14th Int. Conf. Engineering Design*, pp. 185–186, Stockholm.

Bracewell, R.H., Ahmed, S., & Wallace, K.M. (2004). DRed and design folders: a way of capturing, storing and passing on knowledge generated during design projects. *Proc. ASME Design Automation Conf.*

Brill, E., Lin, J., Banko, M., Dumais, S.T., & Ng, A.Y. (2001). Data-intensive question answering. *Proc. Tenth Text Retrieval Conf. (TREC 2001)*, pp. 183–189.

Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R., and et al. (2001). *Issues, Tasks and Program Structures to Roadmap research in Question & Answering (QA)*. Washington, DC: NIST.

Burstein, J., Marcu, D., & Knight, K. (2003). Finding the WRITE stuff: automatic identification of discourse structure in student essays. *IEEE Intelligent Systems 18(January/February)*, 32–39.

Diekema, A.R., Yilmazel, O., Chen, J., Harwell, S., He, L., & Liddy. E.D. (2004). Finding answers to complex questions. In *New Directions in Question Answering* (Maybury, M.T., Ed.), pp. 141–152. Cambridge, MA: AAAI–MIT Press.

Franz, M., & Roukos, S. (1994). TREC-6 ad-hoc retrieval. *Proc. Sixth Text Retrieval Conf. (TREC-6)*, pp. 511–516.

Hai, D., & Kosseim, L. (2004). The problem of precision in restricted-domain question-answering: some proposed methods of improvement. *Proc. Workshop on Question Answering in Restricted Domains in ACL*, pp. 8–15, Barcelona.

Hickl, A., Lehmann, J., Williams, J., & Harabagiu, S. (2004). Experiments with interactive question answering in complex scenarios. *Proc. North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL)*.

Hovy, E.H. (1993). Automated discourse generation using discourse structure relations. *Artificial Intelligence 63(1–2)*, 341–385.

Jing, H., & McKeown, K.R. (2000). Cut and paste based text summarization. *Proc. 1st Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pp. 178–185.

Kim, S., Bracewell, R.H., & Wallace, K.M. (2004). From discourse analysis to answering design questions. *Proc. Int. Workshop on the Application of Language and Semantic Technologies to Support Knowledge Management Processes*, pp. 43–49.

Kim, S., Ahmed, S., & Wallace, K.M. (2006*a*). Improving document accessibility through ontology-based information sharing. *Proc. Int. Symp. Series on Tools and Methods of Competitive Engineering*, pp. 923–934.

Kim, S., Bracewell, R.H., Ahmed, S., & Wallace, K.M. (2006*b*). Semantic annotation to support automatic taxonomy classification. *Proc. Int. Design Conf. (Design 2006)*, pp. 1171–1178.

Knott, A., & Dale, R. (1995). Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes 18(1)*, 35–62.
Kunz, W., & Rittel, H.W.J. (1970). *Issues as Elements of Information Systems*, pp. 55–169, Working Paper 131, Center for Planning and Development Research, Berkeley, CA.
Kwok, C., Etzioni, O., & Weld, D. S. (2001). Scaling question answering to the Web. *Proc. 10th Int. Conf. World Wide Web*, pp. 150–161, Hong Kong.
Liddy, E.D. (1998). Enhanced text retrieval using natural language processing. *Bulletin of the American Society for Information Science and Technology 24(4)*, 14–16.
Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., & Karger, D.R. (2003). What makes a good answer? The role of context in question answering. *Proc. IFIP TC13 Ninth Int. Conf. Human–Computer Interaction*.
Lopez, V., Pasin, M., & Motta, E. (2005). AquaLog: an ontology-portable question answering system for the semantic Web. *Proc. Second European Semantic Web Conf. (ESWC)*, pp. 546–562.
Mani, I., & Maybury, M. (1999). *Advances in Automatic Text Summarisation*. Cambridge, MA: MIT Press.
Mann, W., & Thompson, S. (1988). Rhetorical structure theory: toward a functional theory of text organization. *Text 8(3)*, 243–281
Marcu, D. (1999). Discourse trees are good indicators of importance in text. In *Advances in Automatic Text Summarization* (Mani, I., & Maybury, M., Eds.). Cambridge, MA: MIT Press.
Marcu, D., & Echihabi, A. (2002). An unsupervised approach to recognising discourse relations. *Proc. 40th Annual Meeting of the Association for Computational Linguistics*, pp. 368–375.
Marcus, M.P., Santorini, B., & Marcinkiewicz, M.A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19(2)*, 313–330.
Miller, G.A., Beckwith, R.W., Fellbaum, C., Gross, D., & Miller, K. (1993). Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography 3(4)*, 235–312.
Mukherjee, R., & Mao, J. (2004). Enterprise search tough stuff. *ACM Queue 2(2)*, 36–46.
Nyberg, E., Mitamura, T., Frederking, R., Pedro, V., Bilotti, M., Schlaikjer, A., & Hannan, K. (2005). Extending the JAVELIN QA system with domain semantics. *Proc. Workshop on Question Answering in Restricted Domains at AAAI*.
O'Donnell, M. (2000). RSTTool 2.4—a markup tool for rhetorical structure theory. *Proc. Int. Natural Language Generation Conf. (INLG'2000)*, pp. 253–256.
Resnik, P. (1995). Using information content to evaluate semantic similarity in taxonomy. *Proc. 14th Int. Joint Conf. Artificial Intelligence*, pp. 448–453.
Robertson, S.E., Walker, S., Jones, S., & Hancock-Beaulieu, M.G. (1995). Okapi at TREC-3. *Proc. Third Text Retreval Conf. (TREC-3)*, pp. 550–225.
Salton, G. (1989). Advanced information-retrieval models. In *Automatic Text Processing* (Salton, G., Ed.), chap. 10. Reading, MA: Addison–Wesley.
Sekine, S., & Grishman, R. (2001). A corpus-based probabilistic grammar with only two non-terminals. *Proc. Fourth Int. Workshop on Parsing Technologies*, pp. 216–223.
Taboada, M., & Mann, W. (2006). Rhetorical structure theory: looking back and moving ahead. *Discourse Studies 8(3)*, 423–459.
Teufel, S. (2001). Task-based evaluation of summary quality: describing relationships between scientific papers. *Proc. Int. Workshop on Automatic Summarization at NAACL*.
Voorhess, E.M. (2002). Overview of the TREC 2002 question answering track. *Proc. Text Retrieval Conf. (TREC)*.
Williams, S., & Reiter, E. (2003). A corpus analysis of discourse relations for natural language generation. *Proc. Corpus Linguistics*, pp. 899–908.

Sanghee Kim is a Research Associate in the Engineering Design Centre at the University of Cambridge. She works on the University Technology Partnership project, which is a collaboration between Rolls-Royce, BAE SYSTEMS, and the Universities of Cambridge, Sheffield, and Southampton. She previously worked in the Intelligence, Agents, Multimedia Group at the University of Southampton for the Semantic and content-based multimedia exploitation for European benefit project. Her research interests include text mining focusing on Bayesian probability, symbolic learning, natural language processing, and ontology engineering. Applying question answering components and information extraction into mining Web pages is also of interest.

Rob H. Bracewell is a Senior Research Associate and Assistant Director of the Engineering Design Centre at the University of Cambridge. A mechanical engineer, he has previously researched in the fields of ocean wave-energy exploitation, mechatronics, and advanced robotics. Recently, his main interests have been in creating software tools to help designers, in generating and evaluating innovative design solutions, and unobtrusively capturing the rationale behind their decisions. His software tool, DRed, researched and developed in close collaboration with Rolls-Royce PLC, has rapidly gained acceptance as the standard tool for capturing and communicating design rationale across the company.

Ken M. Wallace joined the University of Cambridge in 1978. In 1991 he was responsible for setting up the Engineering Design Centre and he is currently Chairman of the Centre. He is Co-Director of the BAE SYSTEMS/Rolls-Royce University Technology Partnership for Design. In 1999 he was elected a Fellow of the Royal Academy of Engineering. In 2001 he was awarded the ASME Ruth and Joel Spira Outstanding Design Educator Award and in 2002 he was awarded the Sir Misha Black Award for Innovation in Design Education. Ken translated and edited the English editions of *Engineering Design A Systematic Approach* by Pahl and Beitz.