Chapter 6

Question Answering in Terminology-Rich Technical Domains

Fabio Rinaldi, Michael Hess, James Dowdall, Diego Mollá, & Rolf Schwitter

6.1 Introduction

Technical documents are very often the only reliable knowledge sources about complex products. Documents are the central repository of information otherwise distributed among numerous experts in an enterprise. To utilize these knowledge sources, methods are needed to answer questions over texts in specific problem situations quickly and with high precision. Research in the area of question answering (QA) over texts has been promoted in the last few years in particular by the QA track of the Text Retrieval Conference (TREC QA) competitions (Voorhees 2001). The TREC QA competitions focus on open-domain systems, i.e., systems that can (potentially) answer any generic question. As these competitions are based on large volumes of text, the competing systems cannot afford to perform resource-consuming tasks, and therefore they usually resort to a relatively shallow text analysis. Very few systems try to do more than skim the surface of the text. In contrast, a question answering system working on technical documents does not have to handle very large volumes of text. It can also take advantage of the formatting and style conventions in the particular text (defined quite strictly in technical documents) and can make use of the specific domain-dependent terminology (Rinaldi et al. 2003b). In other words, technical documentation allows deeper processing strategies which might lead to more accurate results, as we will see in this chapter.

However, technical domains present the additional problem of "domain navigation." Unfamiliarity with domain terminology often results in imprecise or even faulty questions. A useful QA system must try to fix this by detecting

terminological variants and exploiting the relations between terms (like synonymy, meronymy, antonymy).

In this chapter we will first explore (in section 2) the peculiarities of technical documentation. The central role that terminology plays in technical domains (parallel to the role of named entities in open-domain question answering) will be explored in section 3. As an example of the practical application of QA in technical domains we then present (in section 4) a real-world system (ExtrAns), specifically designed for technical domains.

6.2 What's Special About Technical Documentation?

Technical documentation is a low-volume/high-value type of text (Hess et al. 2002). In both respects this is the exact opposite of newswire or newspaper texts whose volume is huge but whose value is modest, and declines rapidly after a few days. These differences have a deep impact on the way such texts must be processed, as we will see in this section.

High Value. Technical documentation is a very important type of text. No technical artifact (from equipment, appliance, or tool to hardware and software to complete factory or plant) comes without technical documentation. The more complex a technical artifact is, the more important its documentation becomes. In safety-critical fields (such as aircraft maintenance) a lack in correctness, completeness, or consistency of the relevant technical documentation can have extremely serious legal consequences.

Low Volume. Despite their importance, technical texts are limited in size. The main UNIX commands are explained in slightly more than 500 "man" pages, and many of these pages contain only a few sentences. Similarly, the *Aircraft Maintenance Manual* (AMM) of the Airbus A320 is about 120 megabytes in terms of pure text, which is at least one order of magnitude inferior to the well over a gigabyte of text used in the TREC conferences (Voorhees 2001).

6.2.1 Peculiarities of Technical Documentation

Technical documentation is particular because of its communicative function. Technical texts describe knowledge about concepts and principles and explain how to use this knowledge to do things and to solve problems in a specific domain. They hardly ever refer to facts (events, objects) that are uniquely located in space and time, unlike newspaper texts. Nevertheless, the objects they refer to can normally be identified uniquely (ultimately, by parts numbers). All this makes such texts easier to process in some respects (no need for named entity recognition or for temporal considerations[1]) but more difficult in others (no situational context available to disambiguate).

Conciseness. Concepts are expressed in technical documents once only and in a concise form. There is no room for data redundancy. The user is expected

to find the information in one specific part of the manual only. Therefore, if a QA system fails to detect a specific nugget of information in one part of a document, it will hardly be able to find a passage elsewhere containing the same information.

Structuring. Technical documents are typically well-structured. For example, the sections of the AMM are divided into chapters numbered according to a standard coding schema. Each chapter is divided into sections, and the numbering of these sections has been standardized so that, for example, the pageblock number 02 of any section of any chapter is about maintenance practices, and pageblock number 03 is always about servicing. The rigid structuring of such texts facilitates easy access to the specific information that the user is looking for.

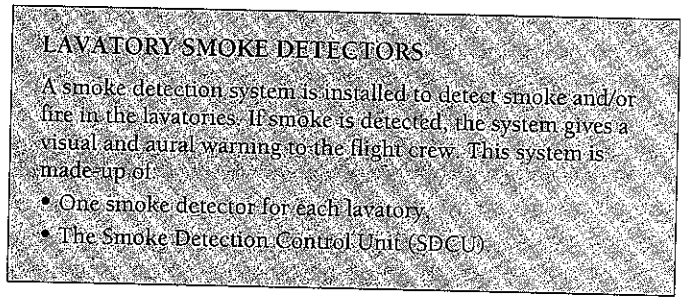
Formatting conventions. Technical text also abounds in formatting conventions. For example, cross-references in the AMM and the man pages are standardized, to the extent that browser software for these manuals can readily use the specific format to convert the cross-references into hyperlinks. Unfortunately, such typesetting conventions are extremely idiosyncratic. What is printed in italics in one manual is rendered in boldface in the next. Completely parameterizable high-performance tokenizers and zone identifiers are indispensable in any system that processes such text.

Terminology. Terminology is ubiquitous in technical texts. Every technical domain has its own terminology, and specialized terminology is unlikely to be included in general lexical resources like WordNet (Fellbaum 1998). Since resources like general grammars and lexica may fail to adequately handle expressions containing these different usages of the terms, the terminology needs always to be built up in relation to the technical domain.

Clarity, simplicity, and reduced ambiguity. The very nature of technical text dictates that the sentences should be easy to understand. This is certainly the case in the AMM which is written in a controlled natural language in order to ease the understanding of operational and functional instructions by technical staff. This simplified form of English is not only easier to read by humans but also easier to process by a computer because the sentence structure is less complex and there are fewer word sense ambiguities.

6.2.2 The Aircraft Maintenance Manual (AMM)

The controlled language used in the AMM of the Airbus A320/330 is AECMA's *Simplified English*, a standard for English-language documentation in the aerospace industry (European Association of Aerospace Industries 2001). Simplified English distinguishes two writing styles: declarative writing and procedural writing. Each section in the AMM starts with a declarative description of a unit and then describes operational and functional procedures that are necessary to maintain the unit.


LAVATORY SMOKE DETECTORS

A smoke detection system is installed to detect smoke and/or fire in the lavatories. If smoke is detected, the system gives a visual and aural warning to the flight crew. This system is made-up of:

- One smoke detector for each lavatory,
- The Smoke Detection Control Unit (SDCU).

Figure 6.1. Example of Declarative Writing

From a cognitive point of view this structure makes a lot of sense for the reader of the document since the acquisition of procedural knowledge is always dependent upon the existence of declarative knowledge and the repeated use of procedures or actions.

Declarative writing has a relatively large proportion of text consisting of full sentences. Still, there are a number of lists with incomplete sentences.

In the example in figure 6.1 the list elements are all noun phrases with a parallel structure. The relation between the elements of a list and the preceding paragraph can be ascertained from the last sentence of the paragraph. Thus, the last sentence in the first paragraph determines that the elements of the first list are the main parts that the smoke detection system consists of. As the paragraph also shows, consistent use of terminology is vital to understand the description.

The use of a low-ambiguity Simplified English makes it comparatively simple to find the underlying logical form of each sentence, as the system described later in this chapter will show. Furthermore, the use of a clear structuring of the text enables the use of text coherence and discourse structure techniques. These features, together with the relatively low volumes of text under consideration, make technical texts very good candidates for experimenting with natural language processing techniques.

6.2.3 Why Typical TREC Techniques Will Not Work

The processing of technical text has an enormous practical potential, especially in cases like the AMM where a quick solution to a technical problem can make a difference between departing on time, departing late, or even cancelling the flight. However, techniques geared towards domain-independent question answering over newswire text or web pages cannot take full advantage of the particularities of technical texts. This is certainly the case with most of the main techniques employed in the QA track of TREC:

Data redundancy approaches that rely on the availability of the same information in different formats cannot be used here because there is very little redundant information, and the information that is repeated is typically repeated in exactly the same way.

Web-based approaches that rely on voting systems need the backing of answers found on the world wide web. However, technical information that can be used to supplement a technical manual is very unlikely to ever be published on the web (for reasons of intellectual property), and if related material is found, there is still the issue of technical accuracy. The amount of accurate data that can be used to back a technical manual is very likely to be insufficient for web-based approaches.

Information extraction approaches that rely on the use of named entities are successful only in domains where fact-based questions are dominant (e.g., "Where can I buy a smoke detector?"). However, questions over technical texts are mostly procedural (e.g., "How do I remove the lavatory smoke detector?").

By operating outside the TREC QA model, systems require novel techniques. This applies to generating multi-sentence answers to definitional questions (see chapter 4) as well as biographical answers aggregated across multiple documents (see chapter 5). For technical domains the most obvious novelties are the techniques for processing the terminology of the domain.

6.3 How to Exploit Terminology (and Why)

Where general knowledge ends, terminology provides the means to name concepts and objects specific to the domain at hand. For the AMM, this includes parts of the aircraft, tools, and procedures which account for more than 30% of the running text. Extending everyday language to name a domain's jargon produces terms exhibiting the production rules of canonical phrases (Sager 1990). For example, terms behave like nominal compounds as: (i) the head is the rightmost word, (ii) only the head carries inflectional morphology, and (iii) the compound is a hyponym of its head. However, terms distinguish themselves by excluding extra-linguistic (pragmatic or idiomatic) interpretation.

Just as term formation is modeled as the interaction between tokens with morpho-syntactic and semantic properties (Kageura 2002), so the reference of a term is recovered by examining its context. Where a term fits into the terminology depends on how its constituent tokens interact with each other. In ExtrAns, this interaction identifies taxonomic and synonymous relations.

The high frequency of terms in technical text produces two main problems when locating answers.

The parsing problem is the difficulty of identifying phrasal boundaries when presented with multiword terms (MWTs). The frequency of tokens unknown to a generic lexicon (e.g. "acetic") increases the number of possible parses for a sentence. As lexical categories are 'guessed', the number of possible syntactic

parses for the sentence increases as the parser tries odd combinations of the tokens belonging to distinct phrases.

The paraphrase problem (Woods, 1997) resides in the imperfect knowledge of users, who are not completely familiar with the domain terminology. Even experienced users, who can be described as domain experts, will not remember the exact form of a term and use a paraphrase to refer to the underlying domain concept. Besides, even in the documents themselves, various paraphrases of the same compound will appear, and they need to be identified as co-referent. However, it is not enough to identify all paraphrases within the manual; novel paraphrases might be created by the users each time they query the system (Rinaldi et al. 2003a).

Overcoming these problems involves the extraction of the document collection's terminology, its analysis to uncover any semantically related terms and the exploitation of this knowledge during the QA process. These three tasks were performed for our QA system.

6.3.1 Extraction

Different sources of information, both internal and external, were invaluable in the extraction process. First, several kinds of external sources (glossaries of abbreviations used in aircraft industry and different specifications (Air Transport Association 1997)) were used. Internally, different types of structures in AMM can indicate the presence of a term. Some of the terms are already explicitly denoted through the use of markup (e.g., element CONNAME for consumable material, element TOOLNAME for tools etc). Other terms can be detected making use of recurring lexical patterns, such as "Power Transfer Unit (PTU)," where capitalized words are followed by an acronym in parentheses which can be detected and properly processed using simple regular expressions. However, the bulk of terminology is detected using the two separate approaches described below.

The first approach is based on a stop-phrase method that chunks certain SGML-zones (titles, paragraphs) using a list of generic (nonterminological) phrases, that often hint at the presence of an adjacent term. For example, from the task title: "Check of the Electrical Bonding of External Composite Panels with a CORAS Resistivity-Continuity Test Set," we cut the prepositions and determiners to obtain the list of candidate terms: "Check," "Electrical Bonding," "External Composite Panels," and "CORAS Resistivity-Continuity Test Set." Given the high incidence of technical terms in the material we are dealing with, even such crude techniques provide interesting results.

The second approach is a fully automatic statistical method (Dias et al. 1999). This is very general, using no linguistic analysis, allowing *n*-grams to be of any length and allowing them to be non-contiguous (i.e., they can contain "holes"). It uses mutual expectation as an association measure, which evaluates

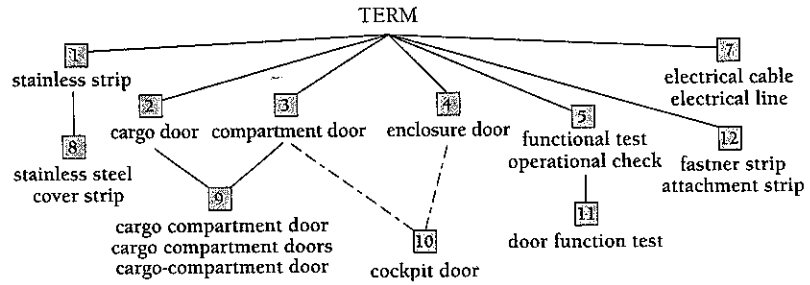


Figure 6.2. The ExtrAns Thesaurus

the cohesiveness of a multi-word unit and a criterion called *LocalMax* to select the candidate terms from the evaluated list. The list obtained by the statistical method of mutual expectation and LocalMax showed the results of recall 44% and precision 15%. For the list obtained by the stop-phrase method the recall was 66% and precision 12%. When combining the methods (the hybrid case), the recall increased to 78% and precision became 10%. After manual validation a 1 million word corpus from the AMM yielded over 13,000 terms.

6.4 Analysis

Once the terminology of a domain is available, it is necessary to detect relations among terms in order to exploit it. We have focused our attention in particular to the relations of synonymy and hyponymy, which are detected as described in this section and gathered in a thesaurus. The organizing unit is a WordNet style synset which includes strict synonymy as well as three weaker synonymy relations. These sets are further organized into an isA hierarchy based on two definitions of hyponymy.

Synonymy detection begins during tokenization to normalize terms that contain punctuation by creating a punctuation free version and recording that the two are strictly synonymous. Further processing is involved in terms containing brackets to determine if the bracketed token is an acronym or simply optional.

In the former case an acronym-free term is created and the acronym is stored as a synonym of the remaining tokens that contain it as a regular expression. Morpho-syntactic processes such as head inversion also identify strict synonymy, like "cargo compartment door" and "door of the cargo compartment." Translating WordNet's synset onto the terminology defines three weaker synonymy relations (Hamon and Nazarenko, 2001) illustrated in figure 6.2: Head synonymy (node 7), modifier synonymy (node 12) and both (node 5). For a description of the frequency and range of types of variation

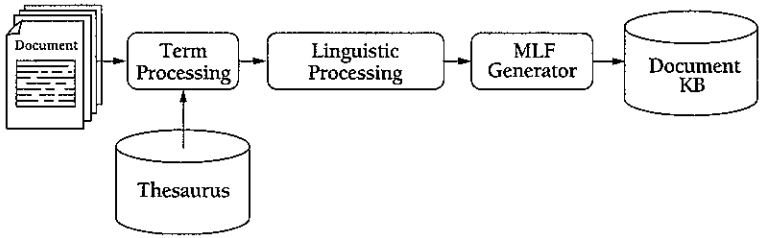


Figure 6.3. Generating the Semantic Representation (Offline Processing)

present in the AMM see Rinaldi et al. (2002).

Hyponymy is defined as two distinct types: modifier addition producing "lexical hyponymy" and "WordNet hyponymy" translated from WordNet onto the term set. As additional modifiers naturally form a more specific term, lexical hyponymy is easily determined. Term *A* is a lexical hyponym of term *B* if: *A* has more tokens than *B*; the tokens of *B* keep their order in *A* and *A* and *B* have the same head.[2] This relation is exemplified in figure 6.2 between nodes 1 and 8. It permits multiple hypernyms, as node 9 is a hyponym of both 2 and 3.

"WordNet hyponymy" is defined between terms linked through WordNet's immediate hypernym relation. The dashed branches in figure 6.2 represent links through modifier hyponymy where the terms share a common head and the modifiers are defined as immediate hypernyms in WordNet. Nodes 3 and 4 are both hypernyms of 10. Similarly, "floor covering" is a kind of "surface protection" as "surface" is an immediate hypernym of "floor" and "protection" is an immediate hypernym of "covering."

Automatically discovering these relations across 6,032 terms from the AMM produces 2,770 synsets with 1,176 lexical hyponymy links and 643 WordNet hyponymy links. Through manual validation of 500 synsets, 1.2% were determined to contain an inappropriate term. A similar examination of 500 lexical hyponymy links identified them all as valid. However, out of 500 WordNet hyponymy links more than 35% were invalid. By excluding the WordNet hyponymy relation we obtain an accurate thesaurus of synsets related through lexical hyponymy which is exploited by our QA system, ExtrAns.

6.5 ExtrAns

ExtrAns answers questions over technical domains exploiting linguistic knowledge from the documents and terminological knowledge about a specific domain. The original ExtrAns system was used to extract answers to arbitrary user queries over Unix documentation files. A set of 500+ unedited UNIX man pages has been used for this application (Mollá et al. 2000). Later, we tackled a different domain, the *Aircraft Maintenance Manuals* (AMM) of the

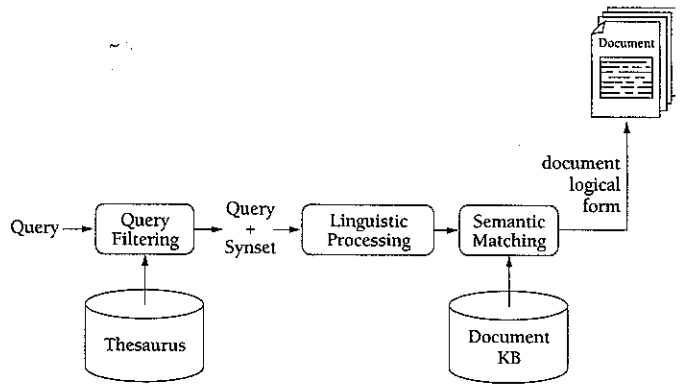


Figure 6.4. Exploiting Thesaurus and KB (Online Processing)

Airbus A320. The highly specific nature of this domain, as well as an SGML-based format and a much larger size (120MB) provided an important test-bed for the scalability and domain independence of the system. An evaluation of the question answering capabilities of ExtrAns against a baseline IR system (based on the MRR measure) is presented in (Rinaldi et al. 2002).

6.5.1 Brief Description of the System

We have chosen a computationally intensive approach to question answering. First, all the documents are analysed in an off-line stage (see figure 6.3) and a semantic representation of their contents is stored in a knowledge base (KB). In an online phase (see figure 6.4), the semantic representation which results from the analysis of the user query is matched in the KB against the stored representations, locating sentences that best answer the query (see figure. 6.5).

The solution to the parsing problem and paraphrase problem (section 6.3) takes place during tokenization of the input stream. Replacing MultiWord Terms (MWT) with their synset identifier from the thesaurus produces two results. First, a MWT is packed into a single lexical token for parsing—reducing the average number of parses per sentence by 46% (Dowdall et al. 2002). Second, any variant of the underlying concept is represented by the same synset identifier producing an implicit "terminological normalization" of the domain.

Documents (in the off-line stage) and queries (in the on-line stage) are first processed by a tokenizer and the terminology-processing module described above. Subsequent linguistic modules include a parser and grammar (an adaptation of Sleator and Temperley 1993), a partial disambiguator (inspired by Brill and Resnik 1994), and a pronominal anaphora solver (inspired by Lappin and Leass 1994). From the output of the above modules ExtrAns derives one logical form per sentence, or more if there are remaining ambiguities.

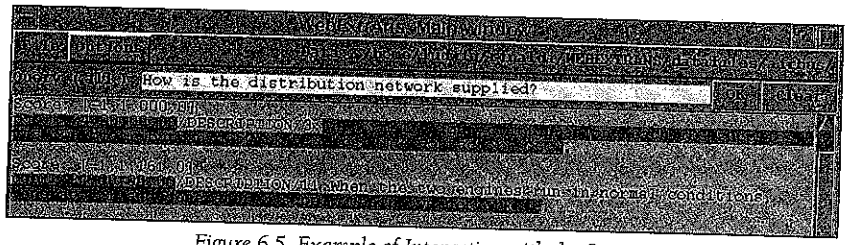


Figure 6.5. Example of Interaction with the System

6.5.2 Minimal Logical Forms

ExtrAns depends heavily on its use of logical forms. The ability of ExtrAns' logical forms to underspecify makes them good candidates for NLP applications, especially when the applications benefit from the semantic comparison of sentences (Copestake et al. 1997). In the case of ExtrAns, the logical forms only encode the dependencies between verbs and their arguments, plus modifier and adjunct relations. Information ignored is, among others, quantification, tense and aspect, temporal relations, plurality, and modality. We have argued elsewhere that overly detailed logical forms may interfere with the answer extraction mechanism and that additional information can be added incrementally (Mollá 2001). This is why ExtrAns' logical forms are called *minimal logical forms* (MLFs).

MLFs use reification to allow different kind of modifications, very much in the line of Hobbs (1985). MLFs do not reify all predicates, as opposed to Copestake et al. (1997). In the current implementation only reification of objects, eventualities (events or states), and properties is used. MLFs are expressed as conjunctions of predicates where all variables are existentially bound and have wide scope. For example, the MLF of the sentence "A coax cable connects the external antenna to the ANT connection" is:

```
holds(e1), object(coax_cable,o1,[x1]),
object(external_antenna,o2,[x2]), object(ant_connection,o3,[x3]),
evt(connect,e1,[x1,x2]), prop(to,p1,[e1,x3]).
```

In other words, ExtrAns derives from the sentence three multi-word terms and translates them into objects: $x1$ (a coax cable), $x2$ (an external antenna), and $x3$ (an ANT connection). The entity $e1$ represents an eventuality derived from the verb involving two objects, the coax cable and the external antenna. The entity e1 is used in the property derived from the prepositional phrase to assert that the eventuality happens to $x3$, the ANT connection.

An advantage of ExtrAns' MLFs is that they can be produced with minimal domain knowledge. This makes our technology easily portable to different domains. The only true impact of the domain is during the preprocessing stage of the input text and during creation of a thesaurus that reflects the specific terms used in the chosen domain, their lexical relations and their word senses.

6.5.3 Answer Extraction

User queries are processed on-line and the resulting MLFs are proved by deduction over the MLFs of document sentences stored in the KB. More specifically, the MLFs are translated into Prolog predicates, and Prolog's theorem prover is used to find the answers. For example, the following Prolog goal is generated for the question "How is the external antenna connected?"

```
?- object(external_antenna,O2,[X2]),
   evt(connect,E1,[X1,X2]), object(anon_object,O1,[X1]).
```

If a sentence in the document asserts that the external antenna is connected to or by something, the Prolog query will succeed. This something is the anonymous object in the query. If there are no answers or too few answers, ExtrAns relaxes the proof criteria. First, hyponyms are added to the query terms.

Second, the system will attempt approximate matching, in which the sentence that has the highest overlap of predicates with the query is retrieved.

6.6 Conclusion

Technical domains present an interesting opportunity for the exploration of content-based approaches to question answering. We have illustrated this point using a system (ExtrAns) which uses a combination of robust NLP technology and dedicated terminology processing to create a domain-specific Knowledge Base, containing a semantic representation for the propositional content of the documents. We have discussed why techniques that are typically used in data-intensive open domain question answering systems would not work effectively in technical domains that have less data redundancy. One of the major differences between technical documents and open domain texts is the major role played by domain terminology in the former which compares with the role of named entities in open domain question answering.

Acknowledgments

The work described here was funded by the Swiss National Science Foundation (contracts 1214-45448.95 and 1213-53704.98) and by the Gebert Rüf Foundation (contract GRS-043/98). SR Technics provided us with the Aircraft Maintenance Manual of the Airbus A320 in machine readable format.

Notes

1. In open-domain QA an important problem is that of the temporal validity of the answers which might require all information to be time stamped, see chapter 11.

2. This is simply a reflection of the compounding process involved in creating more specific (longer) terms from more generic (shorter) terms.

James Dowdall is a Research Assistant at the Institute of Computational Linguistics of

the University of Zurich. Additional information can be found at www.cl.unizh.ch/dowdall.

Michael Hess is a professor of computational linguistics at the University of Zurich. His interests include computational semantics and logic programming. Additional information can be found at www.cl.unizh.ch/hess

Diego Mollá-Aliod is a lecturer in the Department of Computing and associated with the Centre for Language Technology at Macquarie University in Sydney, Australia. His research interests include question answering and answer extraction. He can be reached at diego@ics.mq.edu.au

Fabio Rinaldi is a project manager at the Institute of Computational Linguistics, University of Zurich. Additional information can be found at www.cl.unizh.ch/rinaldi

Rolf Schwitter is a senior lecturer in the Department of Computing and associated with the Centre for Language Technology at Macquarie University in Sydney, Australia. His research interests include controlled natural languages, answer extraction, and knowledge representation. He can be reached at rolfs@ics.mq.edu.au

Section Three

Multiperspective, Temporal, and Multimedia QA

Just as the second section of this collection introduced new directions in terms of additional types of questions and answers, the chapters in this third section address questions dealing with opinions, time, and multiple media.

The first chapter by Claire Cardie, Janyce Wiebe, Theresa Wilson, and Diane Litman (the first from Cornell University, the others from the University of Pittsburgh) considers the annotation and summarization of opinions in support of multiperspective question answering. The authors describe an annotation scheme developed for the low-level representation of opinions, methods to find and organize opinions in text including the use of opinion-oriented scenario templates, and present results of interannotator agreement studies. Unlike factoid questions, opinionoid questions include examples such as: what is the general opinion from the African press toward the recent presidential election in Zimbabwe?; what was the world-wide reaction to the 2001 annual U.S. report on human rights?; and has there been any change in the official opinion from China toward the 2001 annual U.S. report on human rights since its release?

To answer such questions requires the annotation of the expression of opinion in a text along with its source (i.e., the agent expressing the opinion), its type (e.g., positive, negative, uncertain), and its strength (e.g., low, medium, high, extreme). Opinions can be articulations of private (e.g., unobservable mental or emotional) states expressed using subjective language. This can be done directly (e.g., "western countries were frustrated") or indirectly via subjective expressions (e.g., the anger expressed in lexical choice such as "daylight robbery"). Expressions can capture explicit private state (e.g. "think", "hope," "want") or a private state mixed with speech (e.g., "berate," "praise"). The authors describe a markup language and GATE annotator (http://www.cs.pitt.edu/~wiebe). To assess the quality of the annotation language, interannotator agreement was measured for a particular annotation (called "onlyfactive") that is used to indicate whether the source of the private state or speech event is indeed expressing an emotion, opinion, or other pri-