NLP for Answer Extraction in Technical Domains

Diego Mollá
Rolf Schwitter
Centre for Language Technology
Macquarie University
{diego,rolfs}
@ics.mq.edu.au

Fabio Rinaldi **James Dowdall**
Michael Hess
Institute of Computational Linguistics
University of Zurich
{rinaldi,dowdall,hess}
@cl.unizh.ch

Abstract

In this paper we argue that question-answering (QA) over technical domains is distinctly different from TREC-based QA or Web-based QA and it cannot benefit from data-intensive approaches. Technical questions arise in situations where concrete problems require specific answers and explanations. Finding a justification of the answer in the context of the document is essential if we have to solve a real-world problem. We show that NLP techniques can be used successfully in technical domains for high-precision access to information stored in documents. We present ExtrAns, an answer extraction system over technical domains, its architecture, its use of logical forms for answer extractions and how terminology extraction becomes an important part of the system.

1 Introduction

Early question-answering (QA) systems were complex AI-based systems that converted a natural language query into a knowledge base query, searched in the knowledge base for an answer, and returned the results in natural language. Constructing and maintaining these knowledge bases became a true bottleneck and the resulting systems were brittle in nature and non-scalable. Well-known examples are the SHRDLU system (Winograd, 1972) and the LUNAR system (Woods, 1977).

Recently there has been an increase of research on text-based QA, triggered by the Text REtrieval Conference (TREC) Question Answering Track (Voorhees, 2001). In these modern approaches the knowledge base is replaced by collections of documents (mainly large corpora of newspaper articles) thereby eliminating the major problem of early QA systems.

The TREC Question Answering Track demonstrated from an early stage the deficiency of traditional IR approaches when applied to extracting answers from documents. This inadequacy of IR techniques is most visible when answers have to be found within a small window of text (50 bytes). It turned out that systems that used some form of deeper linguistic knowledge did a good job when the answer had to be localised within a small snippet of text.

Some sort of convergence appears to be emerging towards a common base architecture for text-based QA which is centred around four core components (Voorhees, 2001; Hirschman and Gaizauskas, 2001): A **Passage Retrieval** module is used to identify paragraphs (or text windows) that show similarity to the question (according to some system specific metric), a **Question Classification** module is used to detect possible answer types, an **Entity Extraction** module analyses the passages and extracts all the entities that are potential answers and finally a **Scoring** module ranks these entities against the question type, thus leading to the selection of the answer(s).

Recent QA systems use the Web as a resource. Several contributions to the QA track of TREC used the Web as a means to obtain data redundancy and avoid the need for complex linguistic analysis (Clarke et al., 2001; Brill et al., 2001). The rationale is, provided that we have enough data, there will always be some passage that explicitly shows the answer to the question using a simple pattern. The Web becomes a knowledge resource that can be accessed by crawlers and search engines and used for question answering.

In this paper we will argue that QA over technical domains cannot benefit in the same way from data-intensive approaches. Instead, the formal writing style used in these documents make them a good target object for intensive NLP techniques. The remainder of this paper is structured as follows: In Section 2, we motivate why we believe that technical texts are a good application domain for NLP-intensive approaches. In Section 3, we present ExtrAns, an answer extraction system that finds and displays answers to questions in technical domains. In Section 4, we show how we get a grip on terminology. In Section 5, we discuss how we represent the propositional content of sentences as minimal logical forms. In Section 6, we compare ExtrAns with a traditional information retrieval system. Finally, in Section 7, we conclude and summarize our experiences with NLP for answer extraction.

2 Technical Domains and Terminology

There will always be a need for technical documentation, and there will always be a need for tools that help people find the information they want from technical documentations. A Linux user may want to know how to set a symbolic link to a file or a directory. A user of Photoshop may want to know how to improve the tonal range of an image. A member of an Airbus technical maintenance crew may want to know the location of the Electronic Centralised Aircraft Monitor contactor. These technical documentations are not large when compared with the data used in the TREC Question Answering Track, and the user is unlikely to find the answer to some of these technical questions on the Web.

Approaches that rely on data redundancy do not work well in these domains for two reasons. First of all, the amount of text is not large enough and therefore problems of sparse data are likely to occur. Second, authors of technical manuals typically try to avoid redundancy, they do not want to explain the same concept more than once or twice. Trying to use data redundancy approaches in non-redundant data is a self-defeating task.

On the other hand, technical manuals become good source documents on which to apply NLP-intensive approaches. The formal writing in these texts makes it possible to write a grammar that will cover these texts. In fact, in a parser evaluation up to 90% of the sentences in a software manual were parsed by the publicly-available Link Grammar parsing system after incorporating a specific lexicon, and the evaluation was done more than 5 years ago (Sutcliffe and McElligott, 1996). Current parsing systems have improved since. It is currently possible to build the logical form of a sentence and use it in the question answering process, as the system described in Section 3 shows.

Given the non-redundant nature of technical texts, an approach that attempts to find the meaning of the text and use it for question answering is preferred to an approach that uses bags of words or collections of sentence patterns. In other words, technical texts allow and require the use of NLP-intensive approaches.

Technical domains typically use technical terms that are not defined in standard lexicons. In fact, in any technical domain the most important concepts are represented using terminology. These terms need to be properly detected and managed in order to be leveraged upon in a functioning QA system. For example in the Aircraft Maintenance Manual (AMM) different materials, parts of the aircraft, technician's tools and units of measure are so abundant that without proper identification of technical terms any NLP system would perform very poorly (Dowdall et al., 2002; Rinaldi et al., 2002).

3 ExtrAns, an Answer Extraction System

To deal with real-word problems in technical domains, we have developed and implemented ExtrAns, an answer extraction system that finds and displays precise answers in technical documents.

In contrast to other modern QA systems that operate over large collections of documents and use relatively little linguistic information, 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 in the domain of Unix documentation files. An on-line demo of this early version of ExtrAns is available at the project web page.[1] More recently, we tackled a different domain, the Aircraft Maintenance Manual (AMM) of the Airbus A320 to prove the scalability of our approach.[2] The highly technical nature of this domain as well as an SGML-based format and a much larger size (120MB) than the Unix documentation, provide an important test-bed for the scalability and domain independence of the system. Currently we are integrating the HOWTOs from the Linux domain. These are documents that describe in detail certain aspects of configuring or using the GNU/Linux operating system.

The architecture of the ExtrAns system consists of several modules some of which are adaptations of third-party systems (Figure 1). The entire document collection is processed in an off-line stage and user queries are processed on-line. The same linguistic analysis is applied in both stages, transforming the input into a semantic representation called Minimal Logical Forms (MLFs).

The documents are first processed by the terminology extraction tool FASTR (Jacquemin, 2001) so that linguistic variations of terms can be taken into account. The linguistic analysis is done by Link Grammar (LG), a robust dependency-based parser (Sleator and Temperley, 1993). Multi-word terms are parsed as single syntactic units. This reduces the complexity of parsing (in terms of processing time and memory requirements) of the source text by as much as 50% since there is no need to compute the internal structure of such terms. Different forms of attachment ambiguities (prepositional phases, gerunds, infinitives, and *wh*-relative clauses) are resolved by an extension of Brill and Resnik's approach (Brill and Resnik, 1994). Sentence-internal pronouns are dealt with using the anaphora resolution algorithm (Lappin and Leass, 1994). From these partially disambiguated dependency structures ExtrAns derives one or more MLFs as semantic representation for the core meaning of each sentence (Mollá et al., 2000). If ExtrAns detects that a term belongs to a set of synonyms (= synset) in the terminological knowledge base, then the term is replaced by a synset identifier in the MLF. This results in a canonical form, where the synset identifier denotes the concept named by the terms in the synset (Rinaldi et al., 2003).

Unlike sentences in documents, user queries are processed on-line and the resulting MLFs are proved by deduction over MLFs of document sentences. When no direct answer for a user query can be found, the system is able to relax the proof criteria in a stepwise manner. First, synonyms are considered, then hyponyms, in a next step an overlap of logical forms is calculated, and as a last resort a bag of words approach is used (Mollá et al., 2000).

The MLFs contain pointers to the original text which allow ExtrAns to identify and highlight those words in the retrieved sentence that contribute most to a particular answer (Mollá et al., 2000). An example of the output of ExtrAns can be seen in Figure 2. When the user clicks on one of the answers provided, the corresponding document will be displayed with the relevant passages highlighted. This allows the user to check the answer in the context of the document and to verify the justification of the answer. This is especially important in the case of procedural questions where an explicit solution to a problem is required.

4 Terminology

As Answer Extraction involves a high degree of linguistic processing, terminology quickly becomes a major thorn in the side of computational efficiency. And worse, unstandardised terminology can effectively stop Answer Extraction in its tracks.

To produce a syntactic representation for each sentence the terms need to be identified as a phrasal unit. As only the word *compartment* in the term *overhead stowage compartment* inter-

[1] `http://www.cl.unizh.ch/extrans/`
[2] `http://www.cl.unizh.ch/webextrans/`

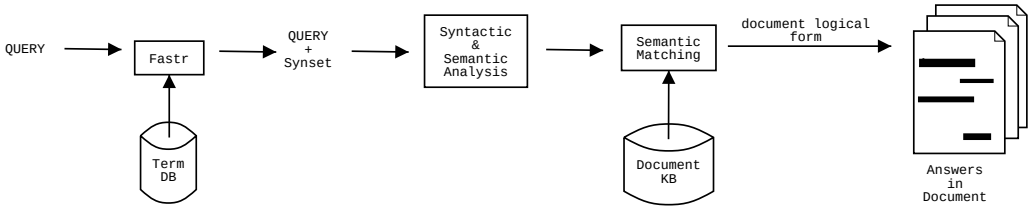

Figure 1: Schematic architecture of the ExtrAns system

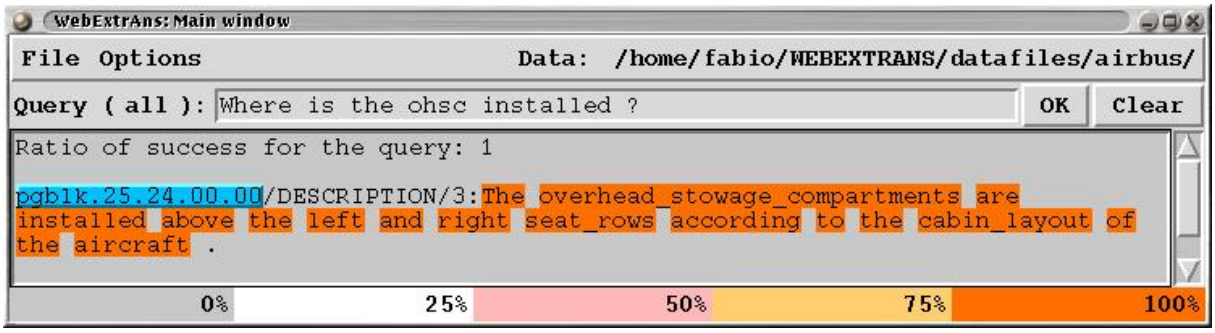

Figure 2: An example of the output of ExtrAns - query window

acts with other sentence words, *overhead stowage* should be effectively ignored to produce the sentence parse. This is an advisable strategy as the *overhead stowage* may be plausibly combined with the wrong sentence words producing multiple possibilities for a given sentence, leaving the correct parse still to be chosen. The parser also wastes effort in assigning a structure to the term itself. Unfortunately, the internal syntactic structure of terms are notoriously idiosyncratic and resist standard parsing techniques.

To address this problem, we pre-process the document collection, extracting all the terminology. The extracted term list is organised into a hierarchy of subtypes to determine which terms are more specific kinds of other terms, and all of the different ways a single concept is refered to in the domain are gathered into synsets.

The subtype relations across the term set can easily be automatically determined. A pleasing characteristic of technical terminology is that more specific terms are formed by adding words to a more generic term. So a simple algorithm can determine that the *overhead stowage compartment* is a subtype of *stowage compartment* and the *crew member seat* is a subtype of *seat*. This hyponymy relation is comparable to the insertion variations defined by (Daille et al., 1996).

In order to find synonymous terms we have adapted the terminology extraction tool FASTR (Jacquemin, 2001). Using phrase structure rules in combination with a morphological database and WordNet, linguistic variations between two terms can be identified. This can detect simple head inversion (*water flow* $\longrightarrow$ *flow of water*), morphological variations (*electric connector* $\longrightarrow$ *electrical connector*) and complex morphosyntactic variations (*electrical generation equipment* $\longrightarrow$ *equipment for generating electricity*).

Exploiting the WordNet synsets allows weaker synonymy relations to be discovered. Terms with synonymous heads (*bulk cargo* $\longrightarrow$ *bulk load*), synonymous modifiers (*upright position* $\longrightarrow$ *vertical position*) or both (*functional test* $\longrightarrow$ *operational check*) are all detected.

We organise the terminology of the domain (6032 terms) in an internal structure that we call the Terminological Knowledge Base containing 2770 synsets with 1176 hyponymy links. In plainer terms, we could describe it simply as a computational thesaurus for the domain, organised around synonymy and hyponymy and stored in a database.

5 Minimal Logical Forms

The success of ExtrAns depends heavily on its use of logical forms. ExtrAns' logical forms are designed so that they are easy to build and to use, yet expressive enough for the task at hand. Not least importantly, the logical forms and associated semantic interpretation method are designed to cope with problematic sentences. This includes very long sentences, even sentences with spelling mistakes, and structures that are not recognised by the syntactic analyser.

To this end, ExtrAns uses *Minimal* Logical Forms (MLFs). The ability of these MLFs to underspecify makes them good candidates for NLP applications, specially when the applications benefit from the semantic comparison of sentences (Copestake et al., 1997; Mollá, 2001). In the case of ExtrAns, the logical forms only encode the dependencies between verbs and their arguments, plus modifier and adjunct relations. Ignored information includes complex quantification, tense and aspect, temporal relations, plurality, and modality. We have argued elsewhere that too detailed logical forms may interfere with the answer extraction mechanism and that additional information can be added incrementally (Mollá et al., 2000).

The MLFs of ExtrAns use *reification* to allow different kind of modifications, very much in the line of (Davidson, 1967; Hobbs, 1985; Copestake et al., 1997). The MLFs do not reify all predicates, as opposed to (Hobbs, 1985; Copestake et al., 1997; Mollá, 2001). In the current implementation only reification of objects, eventualities (events or states), and properties is carried out. The 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 three multiword terms using the Terminological Knowledge Base 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`.

The entities `o1`, `o2`, `o3`, and `p1` are not used in the MLF above, but other more complex sentences may need to refer to the reification of objects (required for non-intersective adjectives such as *the former pilot*) or properties (required for adjective-modifying adverbs such as *very safe*).

Reification can also be used to encode the existence of concepts. The predicate `holds(e1)` expresses that the connecting event `e1` holds in the given context.

In general, MLFs are monotonically extensible and allow us to add new constraints over entities as soon as new information becomes available without destructively rewriting the original expression. For example, the adverb in the sentence *A coax cable* **directly** *connects the external antenna to the ANT connection* changes nothing in the original MLF, but additionally asserts that `e1` is *directly*:

```
prop(direct,p2,e1)
```

Answer extraction is performed by finding those sentences whose MLFs form a superset of the MLFs of the question. To make this happen, the MLFs are translated into Prolog predicates and Prolog's theorem prover is used to find the answers. For example, the following Prolog call is generated for the question *How is the external antenna connected?*

```
?- object(external_antenna,O2,[X2]),
evt(connect,E1,[X1,X2]),
object(Anonymous_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 as described in Section 3.

Given that the MLFs are simplified logical forms converted into flat structures, ExtrAns may

find sentences that are not exact answers but are still related to the user's question. Thus, given the question above, ExtrAns may also find sentences such as

- *The external antenna must not be directly connected to the control panel.*
- *Do not connect the external antenna before it is grounded.*
- *The external antenna is connected, with a coax cable, to the ANT connection on the ELT transmitter*.

An additional 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 the creation of a terminological knowledge base that reflects the specific terms used in the chosen domain, their lexical relations and their word senses.

6 Evaluation

An interesting evaluation framework is to compare the performance of ExtrAns against a traditional IR system – SMART (Salton, 1989). However, the traditional measures of *precision* and *recall* are deceptive in such a comparison due to the contrasting aims of the two systems. For the IR system these measures are of equal importance in an attempt to identify all possibly relevant documents, whereas for the AE system an increased focus on *precision* requires finding at least one answer to the question. With this in mind, a more informative measure is the Mean Reciprocal Rank (MRR) as used in the QA track of TREC (Voorhees and Tice, 1999) The rank of a given result is the position in which the first correct answer is found in the output list of the system. Over a given set of answers the MRR is calculated as the mean of the reciprocals of the ranks of all the answers.

The domain of the evaluation was the 120MB of the AMM (Rinaldi et al., 2002). By manual investigation 100 questions were devised with "known" answers in the document collection. For practical considerations an arbitrary result threshold was set at 10. If no answer was identified in the first 10 results, the case was classified as "Not Found".

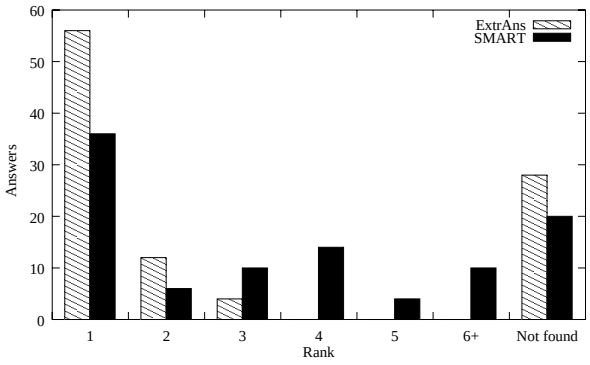

Figure 3: Answers at different ranks

Figure 3 displays the number of answers found at each rank (with 6 to 10 together). Even with the cut off point at 10, SMART clearly finds more answers than ExtrAns. However, in the majority of cases when ExtrAns does find an answer it is placed in first position. Further, in some cases ExtrAns finds more than one valid answer for the same question (possibly in the same document).

The MRR for ExtrAns was 0.63 whereas SMART achieved 0.46. As expected, ExtrAns provides far higher precision than the generic IR system, at the price of smaller recall.

7 Conclusions

We have introduced ExtrAns, an answer extraction system that uses intensive NLP techniques. Decisive factors for the success of ExtrAns include: (*i*) the explicit handling of domain-specific **terminology**; (*ii*) the integration of a full parser and semantic interpreter that include **robust technology** to cope with complex or ungrammatical sentences; (*iii*) The use of a **logical notation** that is flat and encodes the minimal semantic information required for the task at hand; and (*iv*) the integration of **displaying techniques** that help the user to find the answer in context.

Our experience with ExtrAns shows that answer extraction over technical domains is feasible and practical for real-world applications, and benefits from NLP techniques.

References

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. COLING '94*, volume 2, pages 998–1004, Kyoto, Japan.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2001. Web reinforced question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Ann Copestake, Dan Flickinger, and Ivan A. Sag. 1997. Minimal recursion semantics: an introduction. Technical report, CSLI, Stanford University, Stanford, CA.

B. Daille, B. Habert, C. Jacquemin, and J. Royaut. 1996. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–258.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. Univ. of Pittsburgh Press.

James Dowdall, Michael Hess, Neeme Kahusk, Kaarel Kaljurand, Mare Koit, Fabio Rinaldi, and Kadri Vider. 2002. Technical terminology as a critical resource. In *International Conference on Language Resources and Evaluations (LREC-2002), Las Palmas*, 29–31 May.

Lynette Hirschman and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proc. ACL'85*, pages 61–69. University of Chicago, Association for Computational Linguistics.

Christian Jacquemin. 2001. *Spotting and Discovering Terms through Natural Language Processing*. MIT Press.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *Traitement Automatique des Langues*, 41(2):495–522.

Diego Mollá. 2001. Ontologically promiscuous flat logical forms for NLP. In Harry Bunt, Ielka van der Sluis, and Elias Thijsse, editors, *Proceedings of IWCS-4*, pages 249–265. Tilburg University.

Fabio Rinaldi, James Dowdall, Michael Hess, Diego Mollá, and Rolf Schwitter. 2002. Towards Answer Extraction: an application to Technical Domains. In *ECAI2002, European Conference on Artificial Intelligence, Lyon*, 21–26 July.

Fabio Rinaldi, James Dowdall, Michael Hess, Kaarel Kaljurand, and Magnus Karlsson. 2003. The role of technical Terminology in Question Answering. In *accepted for publication at TIA 2003, Terminologie et Intelligence Artificielle, March 31 - April 1, 2003*, Strasbourg, 31 March – 1 April.

Gerard Salton. 1989. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer*. Addison Wesley, New York.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

Richard F. E. Sutcliffe and Annette McElligott. 1996. Using the link parser of Sleator and Temperley to analyse a software manual corpus. In Richard F. E. Sutcliffe, Heinz-Detlev Koch, and Annette McElligott, editors, *Industrial Parsing of Software Manuals*, chapter 6, pages 89–102. Rodopi, Amsterdam.

Ellen M. Voorhees and Donna K. Harman, editors. 2001. *The Tenth Text REtrieval Conference (TREC-10)*, number 500-250 in NIST Special Publication. NIST.

Ellen M. Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track evaluation. In *The Eighth Text REtrieval Conference (TREC-8), Gaithersburg, Maryland*, November 17-19.

Ellen M. Voorhees. 2001. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press.

W.A. Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In A. Zampolli, editor, *Linguistic Structures Processing*, volume 5 of *Fundamental Studies in Computer Science*, pages 521–569. North Holland.