Knowledge-Based Question Answering

Fabio Rinaldi[1], James Dowdall[1], Michael Hess[1],
Diego Mollá[2], Rolf Schwitter[2], and Kaarel Kaljurand[1]

[1] Institute of Computational Linguistics, University of Zürich, Switzerland
{rinaldi,dowdall,hess,kalju}@cl.unizh.ch
[2] Centre for Language Technology, Macquarie University, Sydney, Australia,
{diego,rolfs}@ics.mq.edu.au

Abstract. Large amounts of technical documentation are available in machine readable form, however there is a lack of effective ways to access them. In this paper we propose an approach based on linguistic techniques, geared towards the creation of a domain-specific Knowledge Base, starting from the available technical documentation.
We then discuss an effective way to access the information encoded in the Knowledge Base. Given a user question phrased in natural language the system is capable of retrieving the encoded semantic information that most closely matches the user input, and present it by highlighting the textual elements that were used to deduct it.

1 Introduction

In this article, we present a real-world Knowledge-Based Question Answering[1] system (ExtrAns), specifically designed for technical domains. ExtrAns uses a combination of robust natural language processing technology and dedicated terminology processing to create a domain-specific Knowledge Base, containing a semantic representation for the propositional content of the documents. Knowing what forms the terminology of a domain and understanding the relation between the terms is vital for the answer extraction task.

Specific research in the areas of Question Answering has been promoted in the past couple of years in particular by the Question Answering track of the Text REtrieval Conference (TREC-QA) competitions [17]. As these competitions are based on large volumes of text, the competing systems cannot afford to perform resource-consuming tasks and therefore they need to resort to a relatively shallow analysis of the text. Very few systems tried to do more than skim the surface of the text, and in fact many authors have observed the tendency of the TREC systems to converge to a sort of common architecture [1]. The TREC-QA competitions focus on open-domain systems, i.e. systems that can (potentially) answer any generic question. In contrast a question answering system working on a technical domain can take advantage of the formatting and style conventions in the text and can make use of the specific domain-dependent terminology, besides it does not need to handle very large volumes of text.[2] We found

[1] The term Answer Extraction is used as equivalent to Question Answering in this paper.
[2] The magnitude of the domains we have dealt with (hundreds of megabytes) compares with the size of the TREC collections (a few gigabytes).

that terminology plays a pivotal role in technical domains and that complex multi-word terms quickly become a thorn in the side of computational accuracy and efficiency if not treated in an adequate way.

A domain where extensive work has been done using approaches comparable to those that we presented is the medical domain. The Unified Medical Language System (UMLS)[3] makes use of hyponymy and lexical synonymy to organize the terms. It collects terminologies from differing sub-domains in a metathesaurus of concepts. The PubMed[4] system uses the UMLS to relate metathesaurus concepts against a controlled vocabulary used to index the abstracts. This allows efficient retrieval of abstracts from medical journals, but it requires a complex, predefined semantic network of primitive types and their relations. However, [2] criticizes the UMLS because of the inconsistencies and subjective bias imposed on the relations by manually discovering such links.

Our research group has been working in the area of Question Answering for a few years. The domain selected as initial target of our activity was that of Unix man pages [11]. Later, we targeted different domains and larger document sets. In particular we focused on the Aircraft Maintenance Manual (AMM) of the Airbus A320 [15]. However, the size of the SGML-based AMM (120Mb) is still much smaller than the corpus used in TREC-QA and makes the use of sophisticated NLP techniques possible. Recently, we have decided to embark upon a new experiment in Answer Extraction, using the Linux HOWTOs as a new target domain. As the documents are open-source, it will be possible to make our results widely available, using a web interface similar to that created for the Unix manpages.[5]

The remainder of this paper is organized around section 2, which describes the operations adopted for structuring the terminology and section 3, which describes the role of a Knowledge Base in our Question Answering system.

2 Creating a Terminological Knowledge Base

Ideally, terminology should avoid lexical ambiguity, denoting a single object or concept with a unique term. More often than not, the level of standardization needed to achieve this ideal is impractical. With the authors of complex technical documentation spread across borders and languages, regulating terms becomes increasingly difficult as innovation and change expand the domain with its associated terminology. This type of fluidity of the terminology not only increases the number of terms but also results in multiple ways of referring to the same domain object. Terminological variation has been well investigated for expanding existing term sets or producing domain representations [3,6].

Before the domain terminology can structured, the terms need to be extracted from the documents, details of this process can be found in [4,14]. We then organize the terminology of the domain in an internal component called the Terminological Knowledge Base (TermKB) [13]. In plainer terms, we could describe it simply as a computational thesaurus for the domain, organized around synonymy and hyponymy and stored in

[3] `http://www.nlm.nih.gov/research/umls/`
[4] `http://www.ncbi.nlm.nih.gov/pubmed/`
[5] `http://www.cl.unizh.ch/extrans/`

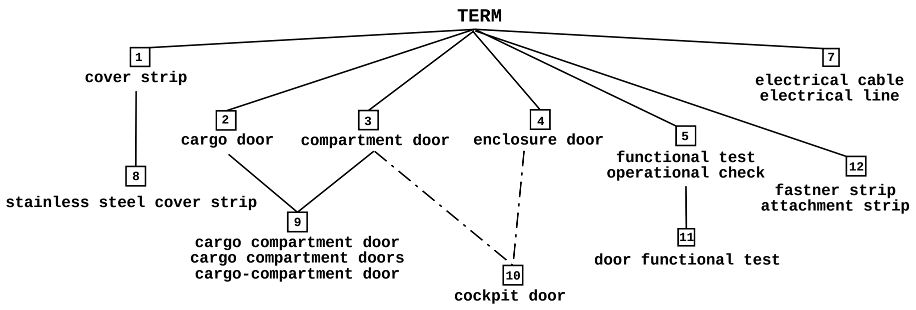


Fig. 1. A sample of the AMM Terminological Knowledge Base

a database. The ExtrAns TermKB identifies strict synonymy as well as three weaker synonymy relations [5] and exploits the endocentric nature of the terms to construct a hyponymy hierarchy, an example of which can be seen in figure (1).

We make use of simple pattern matching techniques to determine lexical hyponymy and some strict synonymy, more complex processing is used to map the immediate hyperonymy and synonymy relations in WordNet onto the terminology. To this end we have adapted the terminology extraction tool FASTR [7]. Using a (PATRII) phrase structure formalism in conjunction with the CELEX morphological database and WordNet semantic relations, variations between two terms are identified.

Hyponymy Two types of hyponymy are defined, 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`; `A` and `B` have the same head.[6] The head of a term is the rightmost non-symbol token (i.e. a word) which can be determined from the part-of-speech tags. This relation is exemplified in figure (1) between nodes $\boxed{1}$ and $\boxed{8}$. It permits multiple hyperonyms as $\boxed{9}$ is a hyponym of both $\boxed{2}$ and $\boxed{3}$.

WordNet hyponymy is defined between terms linked through the immediate hyperonym relation in WordNet. The dashed branches in figure (1) represent a link through modifier hyponymy where the terms share a common head and the modifiers are related as immediate hyperonyms in WordNet. Nodes $\boxed{3}$ and $\boxed{4}$ are both hyperonyms of $\boxed{10}$. Similarly, "`floor covering`" is a kind of "`surface protection`" as "`surface`" is an immediate hyperonym of "`floor`" and "`protection`" is an immediate hyperonym of "`covering'`". Mapping a hierarchical relation onto terms in this manner is fine when the hyponymy relation exists in the same direction, i.e. from the modifiers of `t1` to the modifiers of `t2` and the head of `t1` to the head of `t2`. Unfortunately, it is less clear what the relation between `t1`, "`signal tone`" and `t2`, "`warning sound`" can be characterized as. This type of uncertainty makes the exploitation of such links difficult.

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

Synonymy Four relations make up synsets, the organizing unit of the TermKB. These are gradiated from strict synonymy to the weakest useful relation. Simplistic variations in punctuation $\boxed{9}$, acronym use or orthography produce strict synonymy. Morpho-syntactic processes such as head inversion also identify this relation, `cargo compartment doors` $\longrightarrow$ `door of the cargo compartment`.

Translating WordNets synset onto the terminology defines the three remaining synonymy relations. Head synonymy $\boxed{7}$, modifier synonymy $\boxed{12}$ and both $\boxed{5}$.

Automatically discovering these relations across 6032 terms from the AMM produces 2770 synsets with 1176 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.[7] However, out of 500 WordNet hyponymy links more than 35% were invalid. By excluding the WordNet hyponymy relation we obtain an accurate knowledge base of terms (TermKB), the organizing element of which is the synset and which are also related through lexical hyponymy.

3 Question Answering in Technical Domains

In this section we briefly describe the linguistic processing performed in the ExtrAns systems, extended details can be found in [15]. An initial phase of syntactic analysis, based on the Link Grammar parser [16] is followed by a transformation of the dependency-based syntactic structures generated by the parser into a semantic representation based on Minimal Logical Forms, or MLFs [11]. As the name suggests, the MLF of a sentence does not attempt to encode the full semantics of the sentence. Currently the MLFs encode the semantic dependencies between the open-class words of the sentences (nouns, verbs, adjectives, and adverbs) plus prepositional phrases. The notation used has been designed to incrementally incorporate additional information if needed. Thus, other modules of the NLP system can add new information without having to remove old information. This has been achieved by using flat expressions and using underspecification whenever necessary [10]. An added value of introducing flat logical forms is that it is possible to find approximate answers when no exact answers are found, as we will see below.

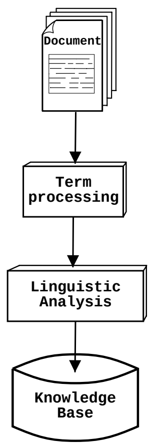


Fig. 2. Creating the KB (offline)

We have chosen a computationally intensive approach, which allows a deeper linguistic analysis to be performed, at the cost of higher processing time. Such costs are negligible in the case of a single sentence (like a user query) but become rapidly impractical in the case of the analysis of a large document set. The approach we take is to analyse all the documents in an off-line stage (see figure 2) and store a representation of their contents

[7] This result might look surprising, but it is probably due to the fact that all terminology has been previously manually verified, see also footnote 6.

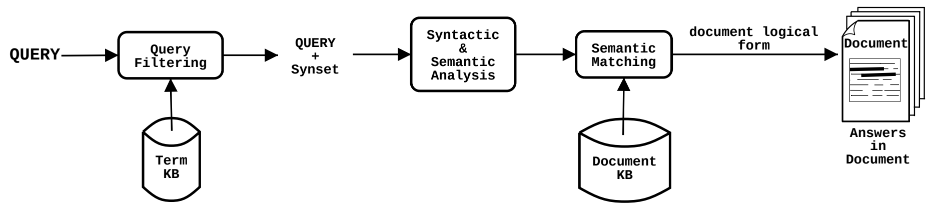


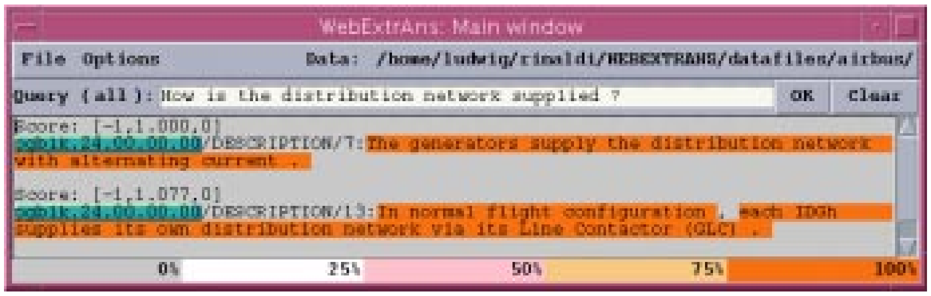
Fig. 3. Exploiting the Terminological KB and Document KB (online)

(the MLFs) in a Knowledge Base. In an on-line phase (see figure 3), the MLF which results from the analysis of the user query is matched in the KB against the stored representations, locating those MLFs that best answer the query. At this point the systems can locate in the original documents the sentences from which the MLFs where generated.

One of the most serious problems that we have encountered in processing technical documentation is the syntactic ambiguity generated by multi-word units, in particular technical terms. Any generic parser, unless developed specifically for the domain at hand, will have serious problems dealing with those multi-words. On the one hand, it is likely that they contain tokens that do not correspond to any word in the parser's lexicon, on the other, their syntactic structure is highly ambiguous (alternative internal structures, as well as possible undesired combinations with neighbouring tokens). In fact, it is possible to show that, when all the terminology of the domain is available, a much more efficient approach is to pack the multi-word units into single lexical tokens prior to syntactical analysis [4]. In our case, such an approach brings a reduction in the complexity of parsing of almost 50%.

During the analysis of documents and queries, if a term belonging to a synset is identified, it is replaced by its synset identifier, which then allows retrieval using any other term in the same synset. This amounts to an implicit 'terminological normalization' for the domain, where the synset identifier can be taken as a reference to the 'concept' that each of the terms in the synset describe [8]. In this way any term contained in a user query is automatically mapped to all its variants.

When an answer cannot be located with the approach described so far, the system is capable of 'relaxing' the query, gradually expanding the set of acceptable answers. A first step consists of including hyponyms and hyperonyms of terms in the query. If the query extended with this ontological information fails to find an exact answer, the system returns the sentence (or set of sentences) whose MLF is semantically closest with the MLF of the question. Semantically closeness is measured here in terms of overlap of logical forms; the use of flat expressions for the MLFs allow for a quick computation of this overlap after unifying the variables of the question with those of the answer candidate. The current algorithm for approximate matching compares pairs of MLF predicates and returns 0 or 1 on the basis of whether the predicates unify or not. An alternative that is worth exploring is the use of ontological information to compute a measure based on the ontological distance between words, i.e. by exploring its shared information content [12].



```
Data: /home/ludwig/rinaldi/WEBEXTRANS/datafiles/airbus/
Query (all): How is the distribution network supplied ?
Score: [-1,1.000,0]
sgbik.24.00.00.00/DESCRIPTION/7:The generators supply the distribution network
with alternating current .

Score: [-1,1.077,0]
sgbik.24.00.00.00/DESCRIPTION/13:In normal flight configuration , each IDGh
supplies its own distribution network via its Line Contactor (GLC) .
```

Fig. 4. Example of interaction with the system

The expressivity of the MLF can further be expanded through the use of meaning postulates of the type: "*If x* **is included** *in y, then x* **is** *in y*". This ensures that the query *"Where is the temperature bulb?"* will still find the answer *"A temperature bulb is included in the auxiliary generator"*. It should be clear that this approach towards inferences has so far only the scope of a small experiment, a large-scale extension of this approach would mean dealing with problems such as domain-specific inferences, contradictory knowledge, inference cycles and the more general problem of knowledge acquisition. In fact such an approach would require a domain Ontology, or even more general World-Knowledge.[8]

While the approach described so far is capable of dealing with all variations in terminology previously identified in the offline stage, the user might come up with a new variant of an existing term, not previously seen. The approach that we take to solve this problem is to filter queries (using FASTR, see figure 3) for these specific term variations. In this way the need for a query to contain a known term is removed. For example, the subject of the query "*Where is the equipment for generating electricity?"* is related through synonymy to the synset of **electrical generation equipment**, providing the vital link into the TermKB.

4 Conclusion

In this paper we have described the automatic creation and exploitation of a knowledge base in a Question Answering system. Traditional techniques from Natural Language Processing have been combined with novel ways of exploiting the structure inherent in the terminology of a given domain. The resulting Knowledge Based can be used to ease the information access bottleneck of technical manuals.

[8] Unfortunately, a comprehensive and easy to use repository of World Knowledge is still not available, despite some commendable efforts in that direction [9].

References

1. Steven Abney, Michael Collins, and Amit Singhal. Answer extraction. In Sergei Nirenburg, editor, *Proc. 6th Applied Natural Language Processing Conference*, pages 296–301, Seattle, WA, 2000. Morgan Kaufmann.
2. J. J. Cimino. Knowledge-based terminology management in medicine. In Didier Bourigault, Christian Jacquemin, and Marie-Claude L'Homme, editors, *Recent Advances in Computational Terminology*, pages 111–126. John Benjamins Publishing Company, 2001.
3. B. Daille, B. Habert, C. Jacquemin, and J. Royaut. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–258, 1996.
4. James Dowdall, Michael Hess, Neeme Kahusk, Kaarel Kaljurand, Mare Koit, Fabio Rinaldi, and Kadri Vider. Technical terminology as a critical resource. In *International Conference on Language Resources and Evaluations (LREC-2002), Las Palmas*, 29–31 May 2002. [8]
5. Thierry Hamon and Adeline Nazarenko. Detection of synonymy links between terms: Experiment and results. In Didier Bourigault, Christian Jacquemin, and Marie-Claude L'Homme, editors, *Recent Advances in Computational Terminology*, pages 185–208. John Benjamins Publishing Company, 2001.
6. Fidelia Ibekwe-SanJuan and Cyrille Dubois. Can Syntactic Variations Highlight Semantic Links Between Domain Topics? In *Proceedings of the 6th International Conference on Terminology and Knowledge Engineering (TKE02)*, pages 57–64, Nancy, August 2002.
7. Christian Jacquemin. *Spotting and Discovering Terms through Natural Language Processing*. MIT Press, 2001.
8. Kyo Kageura. *The Dynamics of Terminology, A descriptive theory of term formation and terminological growth*. Terminology and Lexicography Research and Practice. John Benjamins Publishing, 2002.
9. D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 11, 1995.
10. Diego Mollá. 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, 2001.
11. Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. Extrans, an answer extraction system. *T.A.L. special issue on Information Retrieval oriented Natural Language Processing*, 2000. [8]
12. Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1998.
13. Fabio Rinaldi, James Dowdall, Michael Hess, Kaarel Kaljurand, and Magnus Karlsson. The Role of Technical Terminology in Question Answering. In *Proceedings of TIA-2003, Terminologie et Intelligence Artificielle*, pages 156–165, Strasbourg, April 2003. [8]
14. Fabio Rinaldi, James Dowdall, Michael Hess, Kaarel Kaljurand, Mare Koit, Kadri Vider, and Neeme Kahusk. Terminology as Knowledge in Answer Extraction. In *Proceedings of the 6th International Conference on Terminology and Knowledge Engineering (TKE02)*, pages 107–113, Nancy, 28–30 August 2002. [8]
15. Fabio Rinaldi, James Dowdall, Michael Hess, Diego Mollá, and Rolf Schwitter. Towards Answer Extraction: an application to Technical Domains. In *ECAI2002, European Conference on Artificial Intelligence, Lyon*, 21–26 July 2002. [8]
16. Daniel D. Sleator and Davy Temperley. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292, 1993.
17. Ellen M. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378, 2001.

[8] Available at `http://www.cl.unizh.ch/CLpublications.html`