The Challenge of Technical Text

Michael Hess, James Dowdall, Fabio Rinaldi

University of Zürich, Institute of Computational Linguistics
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
{hess,dowdall,rinaldi}@ifi.unizh.ch

Abstract

When evaluating and comparing Answer Extraction and Question Answering systems one can distinguish between scenarios for different information needs such as the “Fact Finding”, the “Problem Solving”, and the “Generic Information” scenarios. For each scenario, specific types of questions and specific types of texts have to be taken into account, each one causing specific problems. We argue that comparative evaluations of such systems should not be limited to a single type of information need and one specific text type. We use the example of technical manuals and a working Answer Extraction system, “ExtrAns”, to show that other, and important, problems will be encountered in the other cases. We also argue that the quality of the individual answers could be determined automatically through the parameters of correctness and succinctness, i.e. measures for recall and precision on the level of unifying predicates, against a (hand-crafted) gold standard of “ideal answers”.

1. Introduction

The classical type of *information need* satisfied by existing IR systems can be described with the scenario of “Essay Writing”: If you have to write an essay on a given topic you need to locate as much backup material dealing with this topic as possible, i.e. preferably whole documents[1].

Increasingly, more specific types of information needs become important. *First*, one need not catered for by the “Essay Writing” scenario is a determination to locate factual knowledge about individually identifiable entities, concerning their location in time or space, their properties, or their identity with other entities. This could be called the “Fact Finding” scenario, and it is the situation assumed by the QA Track of TREC. The questions are factual questions (“where is/who is XYZ”). One source of such information is, of course, news items but also includes encyclopedias, text books, and fact sheets.

A *second*, equally important, information need beyond the “Essay Writing” scenario arises in situations where concrete problems require explicit solution(s) from a collection of documents. This could be called a “Problem Solving” scenario, and the questions asked are procedural (“how do I do XYZ”). A typical, real world, example is that of an airplane maintenance technician who needs to repair a defective component. He must locate in the massive maintenance manual of the aircraft the exact description of the specific repair procedure. Other text types that contain procedural information are “case data bases” used for trouble shooting purposes, operational handbooks, and some types of scientific articles (e.g. diagnostic and therapeutic reports in medicine).

Third is the situation where you need to find information about principles and regulations, i.e. what one might call the “Generic Information” scenario. The typical questions are definitional (“what is”), and the typical texts consulted in this situation are on-line encyclopedias, but also technical standards publications. Many technical manuals also contain numerous definitions of concepts or devices. It can also be argued that deontic texts (laws etc.) also fall under this heading, and they are extremely important in society.

What users need in the “Fact Finding”, “Problem Solving”, and “Generic Information” scenarios are systems capable of finding those exact (parts of) sentences in document collections that constitute the answer to their question. Depending on the type of question (“where is/who is”, “how do I”, “what is”) different problems will be prominent to different degrees. Thus, named entities are important for answering factual questions but less so for problem solving and definitional questions. There is also evidence that for the latter two types of questions a deeper (syntactic and semantic) analysis of questions is needed than for the factual ones. In order to define standards for comparative evaluations that are not biased towards one particular type of information need, examples of queries and texts of different types should be used from the very beginning.

In the present position statement we will briefly describe ongoing research in the related fields of Question Answering (QA) and Answer Extraction (AE), primarily in the dual context of the TREC QA track (section 2.) and of our own work on the first text type mentioned above, i.e. technical manuals (section 3.). Later we will present some of the problems that are specific to different text types (section 4.), briefly consider the difficulties of evaluating AE systems (section 5.), and finally mention the resources used in our work (section 6.). As relative ’outsiders’ we explicitly aim at providing a critical and, in some respects, dissenting voice, giving the view of somebody approaching Question Answering from a perspective different from that defined (and circumscribed) by the TREC QA track.

2. Results from TREC

Results from the two first TREC Question Answering Tracks (Voorhees, 2000; Voorhees and Harman, 2001) seemed to show that standard, keyword based, IR techniques are not sufficient for satisfactory Answer Extraction. When the answer is restricted to a very small window of text (50 bytes), systems that relied only on those techniques fared significantly worse for the kind of questions used in

[1]It has been often observed that Information Retrieval should rather be called “Document Retrieval”.

the QA track than systems that employed some kind of language processing.

More successful approaches employ special treatment for some terms (Ferrett et al., 2001) and named entity recognition (Humphreys et al., 2001), or a taxonomy of questions (Hovy et al., 2001). Interestingly, some sort of convergence appears to be emerging towards a common base architecture which is centered around four core components (Abney et al., 2000; Pasca and Harabagiu, 2001). Passage Retrieval (Clarke et al., 2001) is used to identify paragraphs (or text windows) that show some general similarity to the question (according to some system specific metric), a Question Classification module is used to detect possible answer types (Hermjakob, 2001), an Entity Extraction module analyzes the passages and extracts all the entities that are potential answers, and finally a Scoring module (Breck et al., 2001) ranks these entities against the question type, thus leading to the selection of the answer(s).

The results of this general design are promising for the kind of factual questions that make sense in the context of news messages. Since such questions ask mostly about properties of individually identifiable entities, good named entity recognition can go a long way towards finding informative text passages. However, for other types of questions (procedural and definitional) we need to be able to analyze other types of constructions, and pinpoint answers more precisely. This means that the choice of a single type of text for the purpose of comparative evaluation creates the risk of "over-fitting" in that all competitors converge on the techniques used by the most successful system for this particular type of text. This effect tends to stifle innovation rather than foster it, and we think that a wider range of texts should be used in comparative evaluation from the beginning to counteract this danger.

It appears that, partly, the problem has already begun to emerge in the latest TREC QA track (TREC10). On one hand, many systems are converging towards the 'generic AE system design' described above, on the other hand, the system that did best (Soubbotin and Soubbotin, 2001) made massive use of heuristics and patterns, that might have limited portability to other domains and other types of applications.

3. ExtrAns

Over the past few years our research group has developed an Answer Extraction system (ExtrAns) (Rinaldi et al., 2002; Mollá et al., 2000) that is mainly geared towards procedural and definitional questions over technical texts.

Two real world applications have so far been implemented with the same underlying technology. The original ExtrAns system is used to extract answers to arbitrary user queries over the Unix documentation files ("man pages"). A set of 500+ unedited man pages has been used for this application. An on-line demo of ExtrAns can be found at the project web page.[2]

More recently we tackled a different domain, the Airplane Maintenance Manuals (AMM) of the Airbus A320. The combined challenges of an SGML-based format and

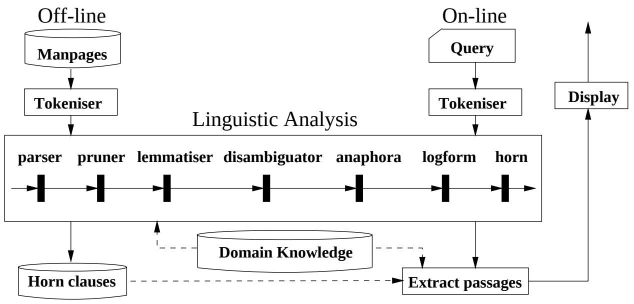


Figure 1: Architecture of the ExtrAns system

the more technical nature of the text and a larger size (120MB)[3] have been met using the original basic architecture (Fig.1), plus a specialized XML based tokenizer and a new CSS-based display utility.

Essentially, ExtrAns extracts answers from documents by semantically comparing queries against document sentences. This is achieved by deriving, from documents and queries, the basic semantic relationships of each sentence and representing them as Minimal Logical Forms (MLF). These are representations that use selected reification and underspecification to keep them open to dynamic, incremental and non-destructive extension, depending on requirements. Answers are derived from these logical forms by deductive proof. This representation is both expressive enough to allow non-trivial comparison and computationally "light" enough for real world applications. True, this approach requires expensive deep linguistic analysis of questions and documents, involving syntax, semantics and consideration of lexical alternations (synonyms and hyponyms) but it returns, in exchange, the exact answer sentences (ideally) and often manages to even determine the individual parts of sentences constituting the exact answer(s) to user questions.

The general design of the system is fairly standard. A (very powerful) tokenizer identifies word and sentence boundaries as well as domain specific multi-word terms. Once tokenized, sentences are parsed using Link Grammar (LG) (Sleator and Temperley, 1993). Link Grammar's ability to predict the syntactic requirements of unknown words ensures that an analysis of all sentences is returned. So ExtrAns always produces MLFs, possibly extended with special predicates that mark any unprocessed tokens as "keywords". Multi-word terms (to be extracted independently and beforehand) are parsed as single syntactic units. Relieving LG of the need to compute the internal structure of such terms reduces the time and space involved for parsing technical text by almost 50%.

A corpus-based approach (Brill and Resnik, 1994) then disambiguates prepositional phrase attachments as well as gerund and infinitive constructions. An anaphora resolution algorithm (Lappin and Leass, 1994) resolves sentence-internal pronouns. The same algorithm can also be applied

[2]http://www.ifi.unizh.ch/cl/ExtrAns/

[3]Still considerably smaller than the size of the document collections used for TREC.

to sentence-external pronouns but this is not (yet) done in ExtrAns.

From the resulting disambiguated linkage, semantic relations between verbs and arguments as well as modifiers and adjuncts are expressed as a MLF. Strict underspecification ensures this only involves objects, eventualities and properties. These predicates are conjoined, and all variables are existentially bound with maximal scope. By way of an example, (1) represents the sentence, *"A coax cable connects the external antenna to the ANT connection"*:

(1)
```
holds(o1),
object(coax_cable,o2,[v3]),
object(external_antenna,o3,[v4]),
object(ANT_connection,o4,[v5]),
evt(connect,o1,[v3,v4]),
prop(to,p1,[o1,v5]).
```

ExtrAns identifies three multi-word terms, translated into (1) as the objects: `v3`, a coax_cable, `v4` an external_antenna and `v5` an ANT_connection. The entity `o1` represents the fact of a 'connect' eventuality involving two objects, the coax_cable and the external_antenna. This reified argument, `o1`, is used again in the final clause to assert the eventuality happens '*to*' v5 (the ANT_connection).

The utility of reification, yielding the additional arguments `o1`, `o2`, `o3` and `o4` as hooks to the abstract entities they denote is that the expression (1) can now be modified by monotonically adding constraints over these entities without destructively rewriting the original expression (Schneider et al., 1999). So the sentence *"A coax cable* ***securely*** *connects the external antenna to the ANT connection"* changes nothing in the original MLF, but additionally asserts (2) that `o1` (i.e. the fact that the coax cable and the external antenna are connected) is *secure*:

(2)
```
prop(secure,p8,o1).
```

This MLF only needs to refer to the reification of an **eventuality** for further modification but other, more complex, sentences will need to refer to the reifications of **objects** (e.g. for non-intersective adjectives) or of **properties** (e.g. for adjective modifying adverbs).

ExtrAns extracts the answers to questions by forming the MLF of the question and running Prolog's theorem prover to find the MLFs from which the question can be derived. So,

"How is the external antenna connected ?'

becomes:

(3)
```
holds(V1),
object(external_antenna,O2,[V5]),
evt(connect,V1,[V4,V5]),
object(anonymous_object,V3,[V4]).
```

If a sentence in the text used as a knowledge base asserts that the *external antenna* is connected to or by *something*, the query will succeed. This *something* is the anonymous object of the query. If there are no answers (or too few) ExtrAns relaxes the proof criteria by introducing hyponymy related tokens as part of the MLF. Additionally, a sentence identifier indicates from which tokens the predicate is derived (not shown in the example above). This information is used to highlight the (relevant parts of the) answer in the context of the document (see Fig. 2).

This kind of very parsimonious representation could appear too "semantically weak" for general QA. This may be true but it is optimized for the task at hand (AE) and can be extended, at will, for more demanding tasks (such as full QA). The MLFs can also be used to ensure that sentences are retrieved that are, in strictly logical terms, not correct answers, but they are useful nevertheless. Thus (4i-ii) are useful (albeit not logically correct) answers, in addition to the correct answers (4iii-iv).

(4) i. The external antenna must not be directly connected to the control panel.

ii. Do not connect the external antenna before it is grounded.

iii. The external antenna is connected, with a coax cable, to the ANT connection on the ELT transmitter.

iv. To connect the external antenna use a coax cable.

4. Text Types, Question Types, and Problem Types

At present, discussions in the TREC community around the further development of Answer Extraction and Question Answering (e.g. in the "Roadmap" document (Burger et al., 2001)) address a very large number of problem and question types, many of them very thorny. However, they do so almost exclusively against the background of *one specific document type*, viz. newspaper texts.

We feel, on the basis of six years' of development and experimentation with Answer Extraction systems, that this exclusive focus on a single, very specific, type of document is not ideal, and that other document types should be considered from the beginning. There are three reasons for this:

1. Processing Strategies developed for newspaper texts become less relevant to users accessing increasing volumes of technical data.

2. Some important problems of AE/QA hardly occur in newspaper texts.

3. Some of the problems that are quite fundamental to any kind of AE/QA can be found in a more isolated, "pure", form in other types of text.

Concerning the *first* point, it is our experience that better access to archived newspaper texts and similar documents is low on the list of priorities for most potential users of QA/AE-Systems in industry, administration, and academia. One exception may be intelligence agencies with interests in monitoring news streams. However, systems that allow high-precision access to the information stored in texts covering narrower, more technical, domains would be welcomed by many organisations in business, administration, and research. Cases in point are (among others):

- Technical manuals of complex systems (any large technical system comes with massive manuals, most often in machine-readable form)
- On-line help systems (for software or other complicated products, such as some financial products)
- Customer queries (systems that process and answer e-mails and/or Web inquiries)
- Access to abstracts and full texts of scientific articles (such as Medline).

Concerning the *second* point, there are some important problems *not* given sufficient weight in the Roadmap document, due to the fairly specific characteristics of newspaper texts:

- **Domain specific terminology:** It is generally recognized that the compilation and use of terminologies is a top priority for the automatic processing of texts in technical applications. The *use* of a (reliable) terminology for a given domain makes the processing of texts vastly simpler, faster, and more useful than without (the quality of Machine Translation systems, for instance, remains dismal without terminology). However, the automatic *compilation* of terminologies ("term extraction") is basically an unsolved problem (none of the available methods produce really useful results). More work is needed in this field but the problem is very peripheral in the Roadmap document.
- **Procedural Questions:** In many of the applications mentioned above (apart from natural language interfaces to technical manuals also on-line help systems and customer e-mail processing systems) the procedural questions of the type *"How do I do X?"* ("How do I convert Apple files to UNIX text format?", "How can I move funds from checking to savings?") are of paramount importance. However, this type of question makes little sense in the framework of newspaper texts, and is therefore given too little attention in the Roadmap document.
- **Generic Questions:** In the documents used for the above-mentioned types of applications (but also in on-line encyclopedias etc.) many sentences are *generic* (timeless rules). Typical questions directed at such texts are *"How do you stop a Diesel engine?"* or *"What is a typhoon?"*. These, too, are relatively rare in newspaper texts (which normally describe individual, time-bound facts), and they are consequently not mentioned in the Roadmap document [4]. Although generic sentences are admittedly a thorny problem they must not be ignored, due to their general importance.

[4]A small number of definitional questions were included in TREC9. In TREC10 their number was significantly higher, due to the different source of the questions. It has however been observed that a corpus of newspaper articles is not the best place to search for answers to that type of questions (Voorhees, 2001).

Concerning the *third* point, there is a number of problems that are fundamental to any kind of AE/QA system, and that do occur in newspapers texts, but which are "drowned" by the numerous other difficulties resulting from the characteristics of newspaper texts. Among them are:

- **Intensional constructions**: Contrary to (almost) common belief, intensional constructions are fairly common in perfectly normal language, and not treating them properly results in wrong answers. Cases in point are "higher order verbs" (as in "pack **attempts** to store the specified files in a packed form" - it may not succeed) and intensional uses of adjectives (as in "Only the super-user can allocate **new** files" - they don't exist yet).
- **Anaphoric references:** Although it has been argued that anaphoric reference (by means of pronouns or definite noun phrases) is irrelevant for document retrieval purposes (or even damaging) the situation is definitely different for AE/QA. Crucial information is often contained in sentences that refer to entities *only* by anaphoric references. Moreover, information is often given in technical manuals just once, so even one missed pronominal reference may seriously impair retrieval performance. Even for the relatively simple task of named entity recognition we must often have recourse to some of the techniques needed for reference resolution ("Bill Gates of Microsoft" &... "Gates" ... "the Gates company" etc.).
- **Pluralities**: Reference to groups of objects (be it through plurals ["dogs"] or through conjunctions ["Fido and Rover"]) is a well-known headache, in particular due to the different possible readings of plural noun phrases (collective/distributive/cumulative: "Fido and Rover fought/barked/ate up the food"). While in many cases it is possible to leave underspecified the exact number of objects introduced by pluralities this is no option when we want to get exact numbers from textual documents (e.g. via "how many"-questions).

The specific characteristics of newspaper texts that somehow overshadow these problems are:

1. **Range of topics:** Due to the vast range of topics covered by newspapers the topic of *sense ambiguity* becomes a top priority problem (cf. "Where is the Taj Mahal?"). In more restricted domains we can usually get away with little or no sense disambiguation (and if we have to perform it, it is much simpler than in open domains). Since sense disambiguation is a very thorny problem, domains where it is not of primary importance would be most useful.

   The wide range of topics also creates the rather ill-understood problem of the type "original vs. copy" ("What is the height of the Statue of Liberty?" - only the original, no models thereof).

2. **Time-dependence of information:** The things described by newspapers are mostly time-dependent

("When was Yemen reunified?" or *"Who is the president of Ghana?").* Keeping track of stages (i.e. the changes that the world is undergoing) is difficult (not least as we can, of course, refer to past states of affairs, and would therefore be able to process the various ways in which natural language encodes such information [the whole tense system!]).

3. **Volume of information:** The sheer volume of information in newspapers archives puts such a heavy burden on processing systems that a strong bias towards shallow analysis is created. One case in point is SRI's TACITUS which was replaced by FASTUS for the MUC competitions, for reasons of speed alone, although TACITUS is a much more powerful system.

Naturally, all these problems will have to be solved sooner or later but, in our opinion, the far more fundamental problems mentioned above could be approached best when kept somewhat sheltered from these minefields.

We certainly do not argue against the use of very large, TREC-like, collections of newspaper texts in the development and evaluation of AE/QA systems but argue for the early inclusion of more moderate volumes of technical texts representative of other, very important, types of documents.

5. Evaluation of AE/QA Systems

As experience gained in the past QA tracks has shown the question of how AE and QA systems shold be evaluated consists of at least two components:

1. What should the answer sets look like?
2. How should the quality of an answer be determined?

The *first* question concerns, among other things, the question of the size of the answer string and, connected with it, that of answer justifications. There is agreement that a fixed-length string that happens to contain the correct answer but in a wrong document context should not be counted as correct (e.g. the answer string "Bush" taken from a document written when George Bush was president but dealing exclusively with shrubs). However, this requirement forces assessors to consult the original document and determine whether the answer string is justified. Clearly a considerable element of uncertainty is entered into the evaluation that way (Is the justification allowed to be implicit in, and/or distributed over, the document? When is an answer justified?)[5].

For a pure AE system, i.e. one *retrieving* explicit answers rather than *computing* answers from possibly distributed, possibly implicit, information (as done by true QA systems) this problem can be contained somewhat by requiring systems to retrieve not fixed-length strings but (not necessarily contiguous) fragments of sentences of potentially unlimited length that, when concatenated, constitute the complete answer, ideally as a well-formed sentence, as seen in Fig. 2. That this is a sensible requirement becomes particularly obvious in technical domains. Consider, for instance, the question:

> **Do I need write permissions to remove a symbolic link?**

A 50-byte answer window may retrieve from the Unix manual, among others, the string:

```
" need write permission to remove a
symbolic link, "
```

Checking the document sentence will reveal that this string is a completely wrong answer as the sentence from which it was taken is:

> **Users do not need write permission to remove a symbolic link, provided they have write permissions in the directory.**

The arbitrary limit of 50 bytes just happened to cut off the crucial negation. However, requiring the AE system to return a complete, ideally well-formed, sentence will result in the justification to be part of the answer itself (in this case, the entire document sentence should be returned).

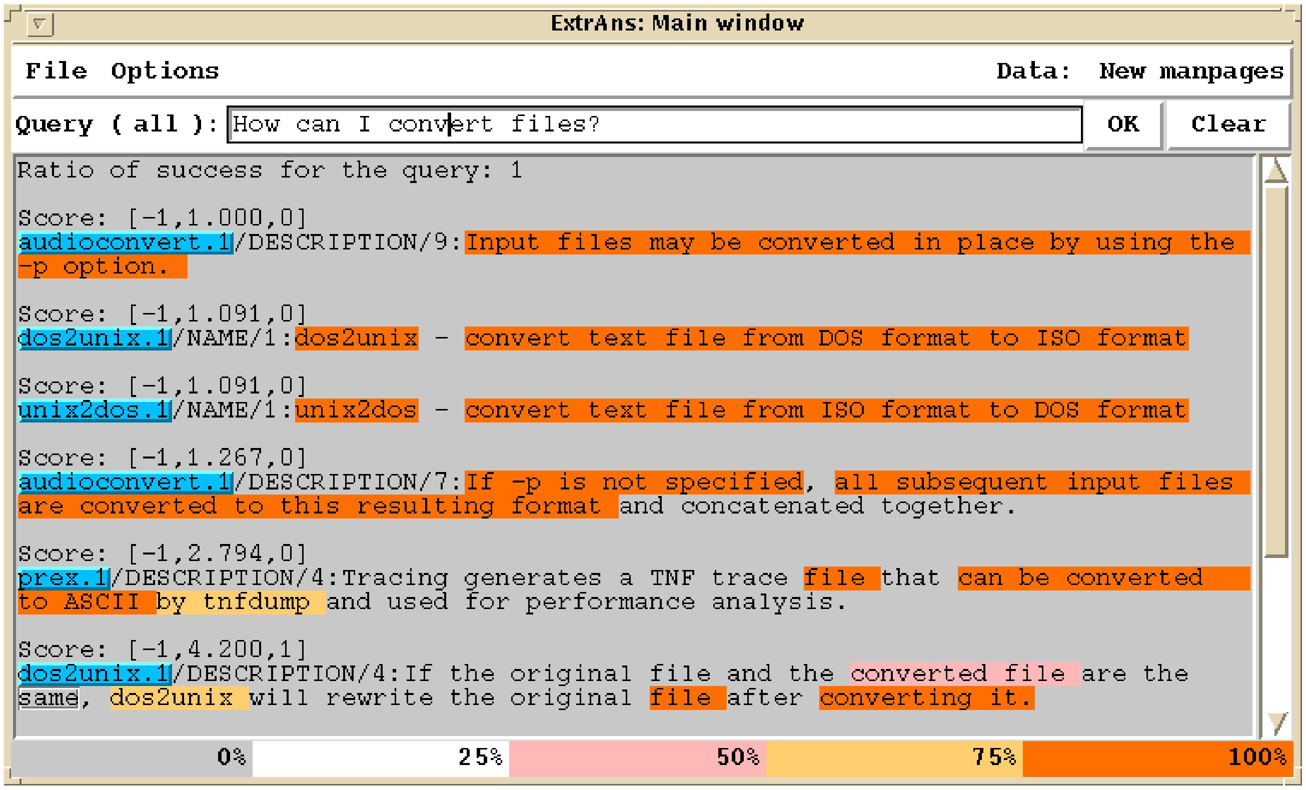
Another aspect of the first question concerns the *test queries*. Clearly, it is always better to use real world queries than queries that were artificially constructed to match a portion of text. By using, as we suggest, manuals of real world systems, it is possible to tap the interaction of real users with this system as a source of real questions (we do this by logging the questions submitted to our system over the Web). Another way of finding queries is to consult the FAQ lists concerning a given system available on the Web. By combining those two sources we compiled a list of 524 questions about the Unix domain. However, a large proportion of them is problematic as they have no answers in the document collection or are clearly beyond the scope of an automatic system (for example, if the inferences needed to answer a query are too complex even for a human judge). Nevertheless they are a useful starting point for a set of test queries in this domain.

Concerning the *second* issue, that of answer quality, the standard measures of Precision and Recall are not ideal for an Answer Extraction system, when applied to individual answer sentences. It can, in particular, be argued that Recall is significantly less important than Precision, as the aim of such a system is to provide (at least) one correct answer, rather than all the possible answers in a given collection. The user needs to find one good answer to a question and they are not interested in repeatedly finding the same answer.

In the Question Answering track of TREC a measure of precision is therefore used that takes this into account, viz. the Mean Reciprocal Rank (MRR). 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 computed as the mean of the reciprocals of the ranks for all the answers.

The problem with this approach is that the underlying assumption, that an answer returned by an AE system is either completely correct or completely wrong, is not entirely realistic. Quite often we get a series of answers

[5]for the latter see:
http://www.isi.edu/natural-language/projects/webclopedia/controv-trec10-eval.html



```
How can I convert files?

Ratio of success for the query: 1

Score: [-1,1.000,0]
audioconvert.1/DESCRIPTION/9:Input files may be converted in place by using the
-p option.

Score: [-1,1.091,0]
dos2unix.1/NAME/1:dos2unix - convert text file from DOS format to ISO format

Score: [-1,1.091,0]
unix2dos.1/NAME/1:unix2dos - convert text file from ISO format to DOS format

Score: [-1,1.267,0]
audioconvert.1/DESCRIPTION/7:If -p is not specified, all subsequent input files
are converted to this resulting format and concatenated together.

Score: [-1,2.794,0]
prex.1/DESCRIPTION/4:Tracing generates a TNF trace file that can be converted
to ASCII by tnfdump and used for performance analysis.

Score: [-1,4.200,1]
dos2unix.1/DESCRIPTION/4:If the original file and the converted file are the
same, dos2unix will rewrite the original file after converting it.
```

Figure 2: Identifying Relevant Parts of Sentences.

which are all correct to some degree but not entirely correct. We need some kind of weighting, exactly as in document retrieval, but again on the sentence level. The way this weighting should be performed is, however, less clear. One approach might be to find a representative set of correct answers by making a person write the ideal answers to a number of questions (labour-intensive but feasible), and then to find the sentences in the documents that are “semantically close” to these ideal answers automatically.

Semantic closeness between a sentence and the ideal answer, i.e. the weight of an answer sentence, could be computed by combining the two measures that one might call *“succinctness”* and *“correctness”*. Both measures compare a potential answer sentence with the ideal answer. Succinctness and correctness are the counterparts of precision and recall, respectively, but now on the sub-sentential level. These measures can be computed by checking the overlap of words between the sentence and the ideal answer (Hirschman et al., 1999), but we suggest a more content-based approach. Our proposal is to compare not words in a sentence, but their logical forms. Of course, this comparison can be done only if it is possible to agree on how logical forms should look like, to compute them, and to perform comparisons between them. The second and third conditions can be fulfilled if the logical forms are simple conjunctions of predicates that contain some minimal semantic information. In this paper we will use a simplification of the minimal logical forms used by ExtrAns (Schwitter et al., 1999). Below are two sentences with their logical forms:

(5) *rm removes one or more files.*
`remove(x,y), rm(x), file(y)`

(6) *csplit prints the character counts for each file created, and removes any files it creates if an error occurs.*
`print(x,y), csplit(x),` `character-count(y),` **`remove(x,z),`** **`file(z)`**`, create(x,z), occur(e),` `error(e)`

As an example of how to compute succinctness and correctness, take the following question:

Which command removes files?

The ideal answer is a full sentence that contains the information given by the question and the information requested. Since *rm* is the command used to remove files, the ideal answer is:

(7) *rm removes files.*
`remove(x,y), rm(x), file(y)`

Instead of computing the overlap of *words*, succinctness and correctness of a sentence could now be determined by computing the overlap of *unifying predicates*. The overlap of the unifying predicates (“overlap” henceforth) of two sentences is the maximum set of predicates that can be used as part of the logical form in both sentences. The predicates in boldface in the two examples above indicate the overlap with the ideal answer: 3 for (5), and 2 for (6).

Correctness of a sentence with respect to an ideal answer (recall on the predicate level) is the ratio between the overlap and the number of predicates in the ideal answer. In the examples above, correctness is 3/3=1 for (5) and 2/3=0.66 for (6). This means that (5) is completely correct in that it returns all the relevant predicates while (6) is only partially correct in that it describes the removal of files by a command but that this command is not the "ideal command" (the removal is, in fact, merely a side-effect of a command whose primary purpose has nothing to do with file removal).

Succinctness of a sentence with respect to an ideal answer (precision on the predicate level) is the ratio between the overlap and the total number of predicates in the sentence. Succinctness is, therefore, 3/3=1 for (5), and 2/8=0.25 for (6). This means that (5) returns only relevant predicates while (6) contains some extraneous material.

Finally, a combined measure of succinctness and correctness could be used to determine the semantic closeness of the sentences to the ideal answer. By establishing a threshold to the semantic closeness, one can find the sentences in the documents that are listed as answers to the user's query.

The advantage of using overlap of unifying predicates against overlap of words is that the (semantically highly relevant) *relations between the words* also affect the measure for succinctness and correctness. We can see this in the following artificial example. Let us suppose that the ideal answer to a query is:

(8) *Madrid defeated Barcelona.*
`defeat(x,y), madrid(x), barcelona(y)`

The following candidate sentence produces the same predicates:

(9) *Barcelona defeated Madrid.*
`defeat(x,y),` **`madrid(y)`**`,` **`barcelona(x)`**

However, at most two predicates can be chosen at the same time (in boldface), because of the restrictions of the arguments. In the ideal answer, the first argument of "defeat" is Madrid and the second argument is Barcelona. In the candidate sentence, however, the arguments are reversed. The overlap is, therefore, 2. Succinctness and correctness are 2/3=0.66 and 2/3=0.66, respectively.

While these ideas have not been implemented yet they may be useful as a contribution to the question of how answers in AE systems should be weighted according to their quality. While the "gold standard" (the ideal answers) would have to be compiled by hand, comparisons against this standard could be done in a wholly automatic fashion.

6. Resources

Some of the resources that we used in our work are:

a The Aircraft Maintenance Manual (AMM) for the Airbus A320. The original SGML markup has been converted into XML for simpler processing (in English, 120 MB total, 45 MB excluding markup).

b The Aircraft Troubleshooting Manual (ATM) for the Airbus A320. Original SGML converted into XML (in English, 62 MB total).

c The on-line manual of Unix (Solaris) in English.

d A list of 524 real user questions about Unix.

e A terminology database (semi-automatically extracted) for the aircraft manuals (approx. 3000 terms).

f Terminology Visualization Tools.
Additional XML markup that denotes the extracted terms is automatically inserted into the manual. The new markup tags can be tied to presentational information (given e.g. by CSS stylesheets), so that when the manual is browsed the terms are highlighted and differentiated from the rest of the text. Most modern web browsers are capable of handling such specification of the information.

Of these resources all the manuals are copyrighted but the lists (questions, terms) are not.

7. References

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer Extraction. In Sergei Nirenburg, editor, *6th Applied Natural Language Processing Conference*, pages 296–301, Seattle, WA.

Eric Breck, John Burger, Lisa Ferro, Warren Greiff, Marc Light, Inderjeet Mani, and Jason Rennie. 2001. Another system called qanda. In *(Voorhees and Harman, 2001)*.

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

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Rilo, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weischedel. 2001. Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). http://www-nlpir.nist.gov/projects/pub/roadmapping.html.

C.L.A. Clarke, G.V. Cormack, D.I.E. Kisman, and T.R. Lynam. 2001. Question Answering by Passage Selection (MultiText experiments for TREC-9). In *(Voorhees and Harman, 2001)*.

Olivier Ferrett, Brigitte Grau, Martine Hurault-Plantet, and Gabriel Illouz. 2001. Qualc - the question-answering system of limsi-cnrs. In *(Voorhees and Harman, 2001)*.

Ulf Hermjakob. 2001. Parsing and Question Classification for Question Answering. In *ACL'01 workshop "Open-Domain Question Answering"*, pages 17–22.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Red: A reading comprehension system. In *Proceedings of ACL'99*, University of Maryland.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2001. Question answering in webclopedia. In *(Voorhees and Harman, 2001)*.

Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. 2001. University of Sheffield TREC-8 Q&A System. In *(Voorhees and Harman, 2000)*.

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

Diego Mollá, Gerold Schneider, Rolf Schwitter, and Michael Hess. 2000. Answer extraction using a dependency grammar. *Traitement Automatique de Langues (T.A.L.), Special Issue on Dependency Grammar*, 41(1):127–156.

Marius Pasca and Sanda Harabagiu. 2001. Answer mining from on-line documents. In *ACL'01 workshop "Open-Domain Question Answering"*, pages 38–45.

Fabio Rinaldi, Michael Hess, Diego Mollá, Rolf Schwitter, James Dowdall, Gerold Schneider, and Rachel Fournier. 2002. Answer extraction in technical domains. In *Proceedings of the CICLING02 Conference*, pages 360–369, February.

Gerold Schneider, Diego Mollá Aliod, and Michael Hess. 1999. Inkrementelle minimale logische formen für die antwortextraktion. In *Proceedings of 4th Linguistic Colloquium*, University of Mainz, September 7-10. FASK.

Rolf Schwitter, Diego Mollá, and Michael Hess. 1999. Extrans - Answer Extraction from Technical Documents by Minimal Logical Forms and Selective Highlighting. In *Proceedings of the Third International Tbilisi Symposium on Language, Logic and Computation*, Batumi, Georgia.

Daniel D. Sleator and Davy Temperley. 1993. Parsing english with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 227–292.

M. M. Soubbotin and S. M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answers. To appear in Proceedings of TREC-10.

Ellen M. Voorhees and Donna Harman, editors. 2000. *Eighth Text REtrieval Conference (TREC-8)*. NIST.

Ellen M. Voorhees and Donna Harman, editors. 2001. *Ninth Text REtrieval Conference (TREC-9)*, Gaithersburg, Maryland, November 13-16.

Ellen M. Voorhees. 2000. The TREC-8 Question Answering Track Report. In *(Voorhees and Harman, 2000)*.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. To appear in Proceedings of TREC-10.