Dealing with ambiguities in an answer extraction system

Diego Mollá
molla@ifi.unizh.ch

Michael Hess
hess@ifi.unizh.ch

Computational Linguistics Group
Dept. of Information Technology
University of Zurich, Switzerland

Abstract

We report on the treatment of ambiguity in ExtrAns, a system that performs an exhaustive linguistic analysis of UNIX manpages to do answer extraction over them. Disambiguation is performed in two stages. The first stage consists of a set of simple rules that delete some of the wrong interpretations that can be spot with purely syntactic information. The second stage extends the use of Brill and Resnik's algorithm to disambiguate several types of attachment ambiguities. Ambiguities that pass the disambiguation procedures are handled by ExtrAns by displaying the answers to the user with graded selective highlighting.

1 Introduction

The treatment of ambiguity is a pervasive problem that any Natural Language Processing (NLP) system needs to address if it is to solve practical problems. This is certainly true of ExtrAns, a system that performs an exhaustive linguistic analysis of UNIX documentation files (the so-called "manpages") to do answer extraction over them. Given a user query worded in natural language, ExtrAns returns those sentences in a set of manpages that answer the query, and it highlights the exact parts of the sentence that give the direct answer. An example of the output of ExtrAns can be seen in Figure 1.

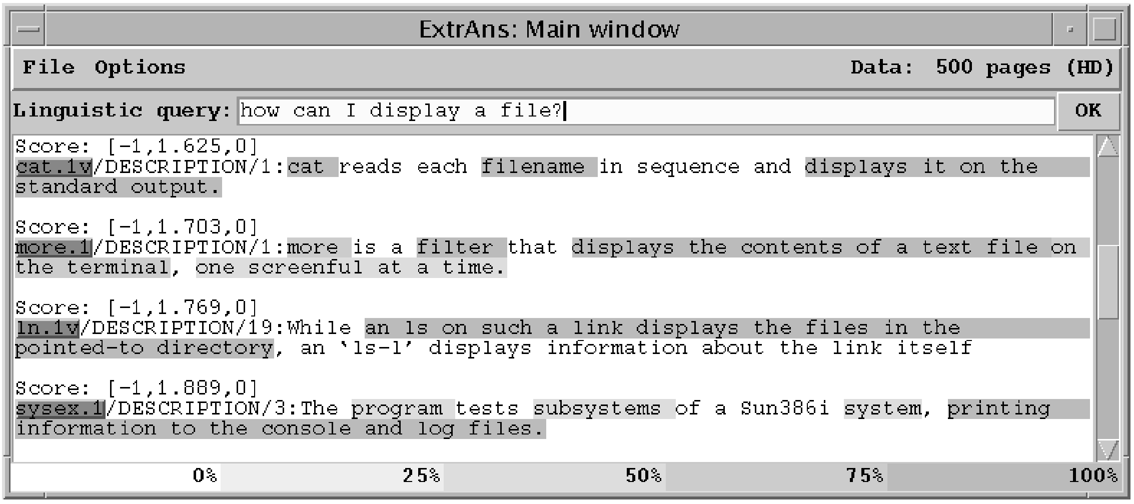

Figure 1: An example of the output of ExtrAns

ExtrAns performs a complex linguistic analysis of the manpages and the user queries by sequentially calling to specific linguistic modules. Detailed explanation of the linguistic modules in

ExtrAns can be found in [1, 5, 4]. The remaining sections report on the specific modules that perform disambiguation, and how remaining ambiguities are handled.

2 Disambiguation based on syntactic information

The sentences are parsed by a third-party parser, the Link Grammar (LG) [7]. LG is a dependency-based parser and grammar that returns the syntactic dependencies between the words in the sentence in the form of the so-called linkages [6]. LG performs syntactic analysis only, without attempting to solve ambiguities. As a consequence, LG may return several alternative linkages of a particular sentence and ExtrAns must perform disambiguation by eliminating the wrong linkages among them.

The pruner performs the first disambiguation step by applying a set of simple rules that rely on syntactic information. Some simple domain knowledge can be used in these rules too, given the technical domain of the UNIX manpages. This is so because the original grammar that comes with LG is targeted towards the parsing of newspaper-like text. Many of the sentence structures that appear in this type of text will not appear in the manpages (such as the several ways of addressing people and specifying dates, the use of contracted forms, etc). The pruner takes advantage of this and includes some simple domain knowledge related with the type of sentences found in the manpages. Typical rules are:

1. The word "regular" can be both an adjective and a noun, but it can modify a noun only as an adjective:

   (1) *the regular.a directory* versus *the regular.n directory*

2. A comma cannot separate an adjective from the head noun:

   (2) *(although they are) (expensive, houses are convenient)* versus *(although they are expensive,) houses are convenient*

3. Idioms are preferred over non-idioms:

   (3) *read_or_write_permissions* versus *read or write permissions*

Rule 1 takes advantage of the stricted use of the word *regular* in the UNIX manpages. Rule 2 complements the internal filtering rules defined by LG. Rule 3 eliminates a linkage (the one without idiom) if there is another linkage which is more convenient (the one with idiom). Some of these rules can be implemented directly in the grammar and built-in postprocessor. Still, other rules are impossible to implement in LG (such as Rule 3 above). Since the latter justify the need for the pruner, we have decided not to modify the default rules in LG in excess, and add these rules to the pruner. This way it is easier to port ExtrAns to another domain without having to modify LG again; only the rules introduced by the pruner would need modification.

3 Disambiguation based on corpus information

There are certain types of ambiguity that cannot be solved with syntactic information only. This is generally the case with attachment ambiguity. ExtrAns' disambiguator module uses Brill & Resnik's algorithm [2] to solve attachment ambiguity. Brill & Resnik's algorithm follows a corpus-based approach. In particular, the algorithm is trained with a corpus and creates a set of rules that can be used to resolve the attachment ambiguity in sentences with a transitive verb and a prepositional phrase (PP). The disambiguation rules consult four pieces of information from the sentence: the verb, the verb object, the preposition, and the prepositional object. The disambiguation rules take this information as input and decide on whether the attachment should belong to the verb or to the verb object.

The output produced by LG is a set of all the possible linkages of the sentence. For every linkage, ExtrAns recreates the data that is needed by the disambiguator and adds to it the attachment decision made in the particular case. The disambiguator is run with the data generated this way, and the attachment decision of the disambiguator (the first piece of information) is compared with the decision made in the linkage (the second piece of information, shown in boxes above). Those linkages that show a highest ratio of correct attachment decisions pass the filter, and the rest are rejected.

The original algorithm [2] can also be used to cover other types of attachment ambiguity, provided that the training corpus offers the data necessary to cover them. The following types of ambiguity have been incorporated:

V + N + particle + VP. *I ran the program by typing*
`run program by type`

V + N + particle + Wh-relative clause. *I give the book to whoever I like*
`give book to whoever`

Multiple PP attachment can also be handled by allowing a linkage to produce data that resembles single PP attachment. For example, the data produced by *cp copies the file from A to B* are:

(4) `copy A to B`

(5) `copy file from A`

(6) `copy file to B`

The introduction of such information creates data that do not always reflect what the original sentences say. In the example above, (4) corresponds with a sentence like *cp copies A to B*. This was not implied by the original sentence. To smoothen potential inaccuracies due to this, the training data also includes this type of information.

Table 1 shows the disambiguation accuracy of ExtrAns after several evaluations with different training and evaluation sets. The figures show that the accuracy is better than with the training set provided by the developers of the original algorithm (who was designed for the Treebank corpus [3]), while the number of rules produced is much smaller. This is due to the fact that the manpages belong to a technical domain, and therefore the disambiguation rules can be more specific.

	`rules` (1770 rules)	`rulesman1` (80 rules)	`rulesman2` (116 rules)
`testman1` (17 manpages)	**72.82%**	—	**76.55%**
`testman2` (17 manpages)	**65.19%**	**74.04%**	—

Table 1: Efficiency of several training sessions of the disambiguator. `rules` are the rules produced with the original training set. `rulesman1` are the rules produced with the training set `testman1`. `rulesman2` are the rules produced with the (disjoint) training set `testman2`.

4 Graded selective highlighting

Answer extraction is performed by applying a proof procedure of the minimal logical forms of the user query against the minimal logical forms of the manpage sentences [1, 5]. Every predicate in the minimal logical forms includes pointers to the words in the sentence that are used to create the predicate. If a particular predicate is used in the proof of the query, the sentence words associated to it are marked for highlighting.

However, some ambiguities survive the two disambiguation stages. There are several reasons for this:

- There are syntactic ambiguities that do not involve attachment. For example, two homographs that fall into different parts of speech may produce two valid interpretations. For that reason, the sentence *time flies like an arrow* would have several interpretations that have nothing to do with attachment ambiguity.
- Not all the attachment ambiguities are covered by the disambiguator. For example, ambiguity introduced by coordination is not covered (e.g. *(after you phoned, I went) and he came* vs. *after you phoned, (I went and he came)*).
- A syntactic structure may produce alternative logical forms. This may happen, for example, when there is lexical ambiguity or two homographs have fall into the same parts of speech (such as the bank of a river and the bank as an institution).

ExtrAns stores the minimal logical forms of all the surviving readings and they are all used by the proof procedure that finds the answer. If several interpretations of one sentence can be used independently in the proof procedure, it may happen that different words are marked for highlighting in each of the alternative interpretations of one particular sentence. The sentence is displayed once, and the words are highlighted following a colouring schema according to the ratio between the number of times the word is marked for highlighting and the total number of times the sentence includes an answer to the question (Figure 1). In other words, relevance of a word is seen here as a degree of unambiguity in the answer provided by the sentence. This way of presenting ambiguous results has two advantages. First, the overall presentation of the results is not cluttered with alternative readings. Second, the user does not need to perform any type of disambiguation to interpret the results.

References

[1] Jawad Berri, Diego Mollá, and Michael Hess. Extraction automatique de réponses: implémentation du système ExtrAns. In *Proceedings of the fifth conference TALN 1998 (Traitement Automatique des Langues Naturelles)*, pages 12–21, Paris, June 1998.

[2] Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING '94)*, volume 2, pages 998–1004, Kyoto, Japan, 1994.

[3] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2), 1993.

[4] Diego Mollá. ExtrAns: An answer extraction system for unix manpages – on-line manual. Technical report, Computational Linguistics, University of Zurich, 1998. http://www.ifi.unizh.ch/CL/.

[5] Diego Mollá, Jawad Berri, and Michael Hess. A real world implementation of answer extraction. In *Proc. of the 9th International Conference and Workshop on Database and Expert Systems. Workshop "Natural Language and Information Systems" (NLIS'98)*, pages 143–148, Vienna, August 1998.

[6] Diego Mollá, Gerold Schneider, Rolf Schwitter, and Michael Hess. Answer extraction using a dependency grammar in ExtrAns. forthcoming.

[7] Daniel D. Sleator and Davy Temperley. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, 1993.