FROM PLAIN ENGLISH TO CONTROLLED ENGLISH

Diego Mollá and Rolf Schwitter
Department of Computing
Language Technology Group
Macquarie University
NSW 2109 Australia
{diego, schwitt}@ics.mq.edu.au

Abstract

We outline an approach for transferring specifications and technical documents written in plain English into controlled English. In a first step, we analyse the sentences of the source documents and (partially) translate them into a set of logical forms. In the second step, we send the logical forms to a surface-form generation module that plans how the sentences of the controlled language should be composed and how the surface string should be realised. The result is an explicit interpretation of the source document in controlled English that can be refined in a stepwise manner by the authors until it corresponds to their intended interpretation.

Introduction

Writing specifications or technical documents in a controlled language is difficult. While the authors are describing something that is in the process of being understood or invented, they have to remember the allowed structures and rules of the controlled language. Fitting ideas straightaway into a controlled language can be harmful, since it inhibits the flow of the ideas. Plain natural language is a great tool for writing rough drafts of something that is only vaguely perceived, but in many cases these drafts need to be reworked sooner or later into precise technical descriptions. At this point controlled languages can play a key role in the document production process, since they have very nice characteristics that one would like to retain to increase the document's precision: they are unambiguous, easily readable, and ideally computer-processable like formal languages [Schwitter 1998, Fuchs et al. 1999]. In this paper we propose a method of how drafts written in plain English can be semi-automatically transferred into precise (formal) documents in controlled English. The problem is similar to that of machine translation, with the obvious advantage that the target language is also (a subset of) English.

To give the authors full freedom to develop their ideas, we let them have the power of plain English to write the first draft of a document. The document is then transferred into controlled English in two steps. In the first step, the document is syntactically analysed and (partially) translated into a set of logical forms [Schwitter et al. 1999, Mollá 2001]. In the second step, the logical forms are sent to a surface-form generation module that plans how the sentence of the controlled language should be composed and how the surface string should be realised [Reiter & Dale 2000].

Each sentence of the original text is processed independently to produce the logical form, and it will generate one or more sentences in the target controlled language.

The Logical Forms

A central point in this process is the format of the logical forms. We propose the use of flat logical forms [Schwitter et al. 1999, Mollá et al. 2000a]. Flat logical forms have a number of interesting characteristics. They allow:

- the capturing of different levels of representation, which means that in situations where a parser is not able to produce a complete result, it is still possible to produce a partial representation of the content of the text;
- the monotonic increase of the level of detail in the logical representation, appropriate in situations where a human (or another program) refines the initial flat logical form produced by the system by adding more information;
- the easy comparison of the semantic content of two sentences by computing the overlap in their flat logical forms, thus allowing the computation of sentence similarity; and
- the implementation of practical systems where the linguistic complexity of the domain is very high but only a subset of the semantic information can be extracted or used efficiently, as in the case of answer extraction systems and content-based information retrieval systems.

Flat logical forms are obtained by means of *reification*, whereby abstract objects are converted into concrete entities [Hobbs 1985]. These entities can then be used as arguments in other predicates in the logical forms.

Consider the domain of Unix manuals that we use as a case study in the answer extraction system ExtrAns [Mollá et al. 2000b]. If we reify objects, events and properties, then the representation of the sentence *cp copies files quickly* is:

```
cp(o1,[x1]),
copy(e1,[x1,x2]),
file(o2,[x2]),
quick(p3,[e1]).
```

Here, the elements in bold indicate the reification of two objects (`o1` and `o2`), an event (`e1`), and a property (`p3`). Note that this allows the use of the constant `e1` in the final term, thus avoiding the need for a nested expression like `quick(copy(x1,x2))`.

These flat logical forms are existentially closed conjunctions of terms. The advantage of this simple notation is that it allows the partial encoding of the propositional content of sentences, simply by removing unanalysable natural language expressions. This partiality can be used in a variety of ways in different applications. In the case of ExtrAns, partial representations are used to perform answer extraction.

We propose to generalise ExtrAns' flat logical forms by allowing the reification of any of the terms. We believe that (generalised) flat logical forms are useful not only

for answer extraction but also for a wide range of applications requiring the use of (possibly partial) logical information from unrestricted text [Mollá 2001], including the paraphrasing of text into controlled English. Take, for example, the sentence:

The customer should enter the VISA card with a pin code into the slot of the ATM.

The natural language sentence contains a modal verb (*should*), two compound nouns (*VISA card* and *pin code*), an abbreviation (*ATM*), and three prepositional phrases (*with the pin code*, *into the slot*, and *of the ATM*). Most of these elements lead to ambiguity problems during the syntactic analysis and to competing formal representations if the ambiguities cannot be resolved. Even worse, the modal verb (*should*) indicates that the state of affairs described in the sentence can be interpreted either from an epistemic or a deontic viewpoint (therefore modal verbs would have to be formalised with respect to the conversational backgrounds they allow) and it is not clear how to represent these readings properly in first-order logic. Here is one out of several competing partial representations for the example sentence by using flat logical forms:

```
customer(o1,[x1]),
enter(e1,[x1,x2]),
visa_card(o2,[x2]),
with(p1,[x2,x3]),
pin_code(o3,[x3]),
into(p2,[e1,x4]),
slot(o4,[x4,x5]),
automated_teller_machine(o5,[x5]).
```

The formal representation above is not a full logical form and it shows a couple of design decisions: all compound nouns have been translated into single terms, the modal verb has been completely ignored, and the *of*-preposition has been absorbed by a relational term (`slot`).

Since flat logical forms are conjunctions of terms, it is very intuitive to express partial information with them, namely by omitting some of the terms of an otherwise full logical form. This, in turn, makes it possible to produce systems that deal with very complex or ungrammatical sentences by deriving only the logical forms of the known parts of a sentence. Therefore, flat logical forms allow us to work with unrestricted, unedited text and they can be produced with the current technology [Mollá et al. 2000a]. Flat logical forms are thus a convenient internal representation of text in robust natural language understanding systems. It is theoretically possible to use a shallow parser in the process, such as the parser used in GATE [Wilks & Gaizauskas 1999], generating as a result underspecified logical forms. The degree of accuracy in the translation would obviously depend on the thoroughness and accuracy of the parser's output and the level of detail in the logical forms, but this procedure would ensure that we get some sort of result. The task of the controlled natural languages is now to paraphrase the interpretation of the logical forms and make them easily readable.

Generating a Paraphrase in Controlled English

The logical representation above is sent to the microplanner of the generation module that uses internal rules in order to plan the controlled English paraphrase of the original sentence. Example rules are [Schwitter 1998]:

1. Prepositional phrases in adjunct position always modify the verb.
2. Relative sentences modify the immediately preceding noun.
3. Only *of*-constructions are allowed as post-nominal modifiers.
4. Abbreviations are resolved to their full form.

The microplanner detects a problem between the logical form of the source sentence and these rules since the translation of the prepositional phrase *with the pin code* resulted in a post-nominal modifier of the compound noun *Visa card*. The micro-planner tries to repair this difference and finally suggests using a relative sentence (*that has a pin code*) instead of the prepositional phrase.

By using meta-programming techniques, the microplanner takes the logical form of the source sentence as input:

```
visa_card(o2,[x2]),
with(p1,[x2,x3]),
pin_code(o3,[x3]),
```

and employs a rewriting rule of the form:

```
N(O2,[X2]), noun(N), with(P1,[X2,X3]) =>
     N(O2,[X2]), have(E2,[X2,X3]]).
```

This rule specifies that, if a term can be expressed by a noun and if it is modified by the term ‘with’, then convert this term into a ‘have’ term. The test `noun(N)` can be performed by a dictionary lookup. The rule will produce the following representation:

```
visa_card(o2,[x2]),
have(e1,[x2,x3]),
pin_code(o3,[x3]),
```

There are cases where the microplanner needs additional syntactic information to produce the correct transformation. Therefore, local syntactic structures of the input sentence are employed as supplementary constraints in the rewriting rules (e.g. the logical forms currently ignore number and verb tenses. Since we have to re-create these in the controlled English languague, we will need an appropriate data structure that encodes these features).

The surface realisation component takes the output produced by the microplanner and converts it into a surface string that corresponds to the rules of the controlled language. For the transformed part of our example, the string

Visa card that has a pin code

is generated and embedded into the final result in controlled English:

The customer [enters the [VISA card that has a pin code] into the slot of the Automated Teller Machine].

The square brackets in the controlled language version above make the interpretation explicit showing that the prepositional phrase *into the slot* modifies the verb *enters* and the relative sentence *that has a pin code* modifies the compound noun *Visa card*.

In addition to this paraphrase in controlled language an explanation is produced that shows the users which transformations have been applied:

T1: Visa card with => Visa card that has.

In this way, there is no hidden information, the user is always informed about the applied transformations and will be able to learn something about the structure of the controlled language simply by observing the feedback information. The left-hand side of the explanation will be an extract of the source text (the extract can be traced from pointers attached to the terms in the logical forms, see below), and the right-hand side of the explanation will be automatically generated from the instantiation of the right-hand side of the transformation rule.

Since the input to the microplanner could be a partial logical form, the microplanner may not have enough information to produce a complete paraphrase. The semantic interpreter that produces the logical forms will keep track of the parts of the original sentence that have been used. The system will display the parts that have been used for the translation by highlighting them and give the user useful feedback about those parts of the input text that still require rewording. The basic methodology to keep track of the used parts of the input text and the highlighting mechanism need not be very different from earlier work on selective highlighting done in ExtrAns [Schwitter et al. 1999]. Suppose, for example, that the parser is unable to parse a sentence with noun-modifying prepositional phrases. Many parses (especially shallow parsers) often have this type of restrictions, but still they try to output the syntactic structure of the parts that they can process. The parser used in ExtrAns, the Link Grammar [Sleator & Temperley, 1993], is not a shallow parser but it can ignore words in the sentence until a full parse is found. ExtrAns would still be able to produce partial logical forms of the output of the parser. The terms in the partial logical forms contain pointers to the relevant words in the sentence:

The customer should enter the VISA card with a pin code into the slot of the ATM.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

```
customer(o1,[x1])/[1,2],
enter(e1,[x1,x2])/[4],
visa_card(o2,[x2])/[5,6,7],
into(p2,[e1,x4])/[12],
slot(o4,[x4,x5])/[13,14,15],
automated_teller_machine(o5,[x5])/[16,17].
```

After the paraphrase has been produced and displayed in one window, the original text can be displayed in another window with those parts highlighted that have been successfully processed by the system:

The customer** should **enter the VISA card** with a pin code **into the slot of the ATM.

More serious is the case when the partial logical form cannot generate a complete sentence in controlled English. In that case, the paraphraser can only generate phrases (not full sentences) that correspond with the partial logical forms that have been created by the logical form generator. For example, let's assume that the parser cannot process the entire prepositional phrase *with a pin code* but is able to recognise the compound noun *pin code*. On the basis of this partial information the logical form generator would produce the following representation:

```
customer(o1,[x1])/[1,2],
enter(e1,[x1,x2])/[4],
visa_card(o2,[x2])/[5,6,7],
into(p2,[e1,x4])/[12],
slot(o4,[x4,x5])/[13,14,15],
automated_teller_machine(o5,[x5])/[16,17],
pin_code(o6,[x3])/[10,11].
```

The term `pin_code(o6,[x3])/[10,11]` is not related to the propositional content since `x3` is not used anywhere else in the logical form. This interpretation is directly reflected in controlled language where a string of the form:

The customer [enters the VISA card into the slot of the Automated Teller Machine].

is generated together with unrelated text:

{pin code}

It is now the task of the author to reformulate the sentence and use this text in accordance with the rules of the controlled language.

Conclusions

This proposed research bridges the gap between the use of unrestricted text in natural language and a controlled language with formal properties. By automatic converting a text into the equivalent text in a controlled language, the author does not need to learn the rules that govern the controlled language. We are aware that the methodology introduced here would not always produce 100% accuracy in the first translation step. However, since the controlled language is a subset of English, the author can easily check if the current translation is correct and act accordingly by modifying the original text, modifying the current translation, or starting from scratch. The system proposed in this paper is not an error-free complete translator that translates unrestricted text into controlled English. Rather, the system is a tool for the author who needs to produce a specification or a technical document in controlled English. Further work would be needed to develop human-computer interaction techniques that would provide supplementary aid for the author. Apart from generating paraphrases in

a controlled language, flat logical forms can be used for other real-world applications where high flexibility for dealing with unrestricted text is a prerequisite.

References

[Fuchs et al. 1999] N. E. Fuchs, U. Schwertel, R. Schwitter, Attempto Controlled English – Not Just Another Logic Specification Language, Lecture Notes in Computer Science 1559, Springer, 1999.

[Hobbs 1985] J. R. Hobbs, Ontological Promiscuity, Proceedings, 23rd Annual Meeting of the ACL, University of Chicago, Illinois, pp 61-69, 1985.

[Mollá et al. 2000a] D. Mollá, G. Schneider, R. Schwitter, and M. Hess, Answer extraction using a dependency grammar in ExtrAns, Traitement Automatique de Langues, 41(1):127-156, 2000.

[Mollá et al. 2000b] D. Mollá, R. Schwitter, M. Hess, and R. Fournier, ExtrAns, an Answer Extraction System, Traitement Automatique de Langues, 41(2):495-519, 2000.

[Mollá 2001] D. Mollá, Ontologically promiscuous flat logical forms for NLP, Fourth International Workshop on Computational Semantics (IWCS-4), Tilburg, The Netherlands (forthcoming).

[Reiter & Dale 2000] E. Reiter, R. Dale, Building Natural-Language Generation Systems, Cambridge University Press, 2000.

[Schwitter 1998] R. Schwitter Kontrolliertes Englisch für Anforderungsspezifikationen, Dissertation, Institut für Informatik, Universität Zürich, 1998.

[Schwitter et al. 1999] R. Schwitter, D. Mollá, M. Hess, ExtrAns – Answer Extraction from Technical Documents by Minimal Logical Forms and Selective Highlighting, The Third International Tbilisi Symposium on Language, Logic and Computation, Batumi, Georgia, September 12-16, 1999.

[Slator & Temperley 1999] D. Slator, D. Temperley, Parsing English with a Link Grammar, Third International Workshop on Parsing Technologies, August 1993.

[Wilks & Gaizauskas 1999] Y. Wilks, R. Gaizauskas, LaSIE Jumps the GATE, in: T. Strzalkowsky, Natural Language Information Retrieval, Dordrecht, Kluwer, pp. 197-214, 1999.