A Successful Case of Computer Aided Translation

Miguel Filgueiras

LIACC, Universidade do Porto
R. do Campo Alegre 823, 4100 Porto, Portugal
mig@ncc.up.pt

Abstract

This paper reports on the techniques used to successfully translate with the help of a computer a Mathematics textbook from Portuguese to English. Even if these techniques are simple and this case has some peculiarities, it seems that the same method will be useful in other cases as well, and, more importantly, that this experience is a good motivation for some research to be done on the possible refinements of the method.

1 Introduction

This paper reports on the techniques used to successfully translate with the help of a computer a Mathematics textbook from Portuguese to English. These techniques, although quite simple, proved to be surprisingly effective. In fact, for a 400 pages book, the amount of work required from the translator (the author of the book himself) was equivalent to that of typing a few 40 pages. Even if the book subject matter and the pair of languages have some peculiarities, it seems that the same method will be useful in other cases as well, and, more importantly, that this experience is a good motivation for some research to be done on the possible refinements of the method.

2 The Problem

The work I am about to describe originated in the question "Which Machine Translation system should I use to translate my book from Portuguese into English?". To which the only fair answer from anyone acquainted with the current state of the art of Machine Translation ought to be a (maybe qualified) "None!". The book in question is *Semigrupos Finitos e Álgebra Universal*, a textbook on finite semigroup theory.

The problem was then reformulated as "What may help me in avoiding to type in all the 400 pages of the book, given that it is a book on Mathematics and was prepared in LaTeX?". A book on Mathematics meant that the language used was somewhat formal and that all mathematical formulas could be preserved during the translation. That the book has been prepared in LaTeX, a text processor widely used by mathematicians and computer scientists (Lamport, 1986), meant that it would be possible to use the LaTeX commands in the text, for instance, to detect the boundaries of formulas (the same would be true of texts encoded using a mark-up language like, for instance, SGML). This far less ambitious goal of building some tools that would help the translation seemed quite attainable, even if at the time the final result was not expected to be as good as it turned out to be.

3 The Method

The basic method employed consists in having a dictionary of rewrite rules, each one being a sequence of words in Portuguese with its counterpart in English, and in applying these rules to the source text. The dictionary is looked up from the beginning so that the first rule in it whose left-hand side (lhs) matches a prefix of the source text is the one selected, irrespective of existing other rules that could be applied. This means that a dictionary rule whose lhs is a prefix of the lhs of another rule must appear after it. If no rule can be applied the first word in the source text is left unchanged. In any case the same method is used for the rest of the text.

A finer analysis of the source text was added to this basic method in order to cope with LaTeX commands, so that

- mathematical formulas, that must be left unchanged, can be detected,
- LaTeX denotations of diacritics are taken as belonging to the words they occur in,
- some commands (which were called *transparent*), such as those for section names or footnotes, have their arguments translated, while all the others are left unchanged.

Another refinement was the possibility of having

rewrite rules with parameters that stand for formulas, as in

```
de $1 sobre $2 ↦ of $1 on $2
```

Finally, in order to deal with proper nouns, a capital letter at the beginning of the text is given a special treatment. The dictionary is searched for a rule that matches the text as is. Then, the capital letter is converted into lower case and the dictionary is again searched for an appropriate rule. If both searches succeed, the rule that is nearest to the beginning of the dictionary is selected.

4 The Tools

Three small programs (amounting to a total of 9 pages) were written in Prolog to cope with different aspects of the problem at hand.

One of them scans the source text and prints the words in it, skipping formulas and irrelevant LaTeX commands. The list of words thus obtained, after being sorted and after deletion of repeated words (what can be done by using the `sort` utility in a Unix system), is very useful in preparing the dictionary.

The format adopted for the dictionary, as written by the user, is simply that of a text with a rewrite rule in each line, the left-hand side followed by a tabulation character (that will be shown as ↦ in the sequel) and the right-hand side. Parameters standing for formulas in a rule are written as `$N` where `N` is a positive integer. Each such parameter must occur once and only once in each side of the rule and not at the beginning of its lhs. Examples of rules in this format are

```
de $1 sobre $2 ↦ of $1 on $2
sejam $1, $2 e $3 ↦ let $1, $2 and $3 be
```

The second program transforms the dictionary as typed by the user into a set of Prolog clauses for use by the translation tool, our third program. These clauses can be seen as the usual translation of Definite-Clause Grammar (DCG) rules into Prolog but for the order of the arguments. For the sake of efficiency in searching the dictionary for rules that can be applied at each step, we take the "string" arguments of the translation of a DCG non-terminal to be the first arguments of the corresponding Prolog predicate. As these arguments are lists, and the indexing mechanism of the Prolog compiler we use, YAP (Damas et al., 1988), looks at the first element of lists, this results in a speed-up by a factor of 2 or 3. Other points are

- the rewrite rules are numbered in order for the translation tool to be able to decide on which rule to apply when dealing with capital letters (as seen at the end of last section),
- each parameter is replaced by a predicate call that processes a formula,
- when a lhs does not finish with a parameter, a predicate is called that checks the occurrence of a separator.

Examples of rewrite rules in the user language and in Prolog are

```
função identidade ↦ identity function

rule([102,117,110,92,99,123,99,
      125,92,126,97,111,32,105,
      100,101,110,116,105,100,97,
      100,101|L0],L0,1,
     [105,100,101,110,116,105,116,
      121,32,102,117,110,99,116,
      105,111,110|L1],L1) :- sep(L0,_).

imagem de $1 pelo ↦ image of $1 under

rule([105,109,97,103,101,109,32,
      100,101|L1],L2,2,
     [105,109,97,103,101,32,111,
      102|L3],L4) :-
        math(L3,[117,110,100,101,114|L4],
             L1,[112,101,108,111|L2]),
        sep(L2,_).
```

The third program, the translation tool, implements the method described in the previous section. It was written as a DCG in Prolog with some fragments of Prolog to deal with special situations, like the fact that the dictionary rules are implemented as above.

5 The Translation

What I am about to describe was done by Jorge Almeida, the author and translator of the book.

The strategy adopted for doing the translation was to build and tune a dictionary by focusing on the translation of a single, representative chapter of the book, then further enhance the dictionary on the basis of the translation of the other chapters, and finally to use this dictionary to translate the whole book with our translation tool. The text obtained in this way was then revised (by using a text editor) to have the final version in English of the book.

The first step was to use our first program to have a list of the words in all the book. This list was sorted out (and repeated entries deleted) by using the UNIX `sort` utility. A preliminary version of the dictionary was built on the basis of the resulting list. Inspection of the output of the translation tool (using this dictionary on the selected chapter) suggested the addition of new rules to the dictionary. After some iterations of this process an acceptable translation of the selected chapter was obtained. Some

further refinements to the dictionary were made by applying this same technique to other chapters.

Some rules used in the actual translation were

```
a operação ↦ the operation
a ↦ to
Estado de São Paulo ↦ Estado de São Paulo
variáveis distintas ↦ distinct variables
```

The first two rules are related with a lexical ambiguity problem: *a* in Portuguese is both an article and a preposition. In the absence of a syntactic analysis, the latter alternative is taken as the default (2nd rule above); this means that *a* as an article will be either manually corrected in the translated text, or translated correctly if another translation rule in which it occurs is applied. This latter case is exemplified by the first rule above. The third rule is an example of an identity rule that, along with the treatment of capital letters, is useful in coping with proper nouns — this particular rule blocks the translation of the preposition *de* by some generic rule. The last rule shows how inversions in word order can be dealt with.

Most of the effort in translating the book was spent in building and tuning the dictionary. The amount of text typed during this phase is estimated in about 20 pages. About another 20 pages were typed during the revisions made to the output of the translation program (this includes the introduction of some small updates and corrections to the original text).

I give now some statistics on the work done on building and tuning the dictionary and translating the book. The execution times below are CPU execution times of Prolog programs using the YAP compiler (Damas et al., 1988) running under EP/IX (a Control Data Corporation version of UNIX) on a CDC 4680 machine (with two MIPS 6000 RISC processors).

Selected chapter	
no. of pages (final)	77
no. of characters	205 KB
no. of words	8571
no. of different words	1168
exec. time to extract words	10 sec
exec. time to translate chapter	25 sec

Final dictionary	
no. of rules	ca. 6000
average total no. of words/rule	3.4
exec. time to process rules	52 sec

Book after translation	
no. of characters	1040 KB
total no. of typeset pages	436
estimated revision effort	80 hours

6 Conclusions

For anyone knowing well the fields of Natural Language Processing and Machine Translation, the case I have presented may come as a surprise — and surprised was I with the results obtained with so simple a method. It can be argued that the pair Portuguese-English makes the translation easy, because, for instance, there is no need for translating most of the gender agreements appearing in Portuguese; and that a book in Mathematics is written in a very restricted kind of language with a relatively small number of different constructions. This is obviously true although some remarks should be made, if one is not trying to have a 100%-correct translation:

1. gender agreements lend themselves to simple treatment in most cases; this is to say that most of the time there will be no need for complex analyses to arrive at their correct translation,
2. the pair Portuguese-English poses some difficult problems concerning word order, and this would be seldom the case, for instance, with pairs of Romance languages,
3. the need for translating scientific texts in Mathematics, Physics, Chemistry, Medicine and other fields in which formalized and restricted subsets of natural languages are used is probably big enough to make translation tools as those I have developed very interesting,
4. to the best of my knowledge, this is the very first case of a book being successfully translated mostly by computer.

The main conclusion to draw seems to be that there are particular translation problems that can be solved or partially solved using simple and efficient methods in a computer, much effort being saved this way. Nevertheless, even if one is to take the results described so far as pertaining more to ingenuity than to research (an analysis with which I fully agree, the time spent in writing down and improving the programs being quite negligible), there is an interesting set of questions put by them which I think should be brought forward to those working in this kind of problems. These questions are:

1. what kind of syntactic analysis (even superficial) is needed to improve these tools, and how to integrate it?
2. the same for semantic analysis,
3. for which language pairs would these tools be in/appropriate, and why?
4. how far can a rewrite rule dictionary as the one described be re-used?

Acknowledgements

The work described in this paper would not have started without the questions I mentioned in the beginning and that were formulated by Jorge Almeida. I would like to express my gratitude for his suggestions on improvements of the programs, for his patience with some unpleasant bugs I produced, as well as for his contributions to and comments on previous drafts of this paper.

References

Luís Damas, V. Costa, R. Azevedo, and R. Reis, 1988. *YAP Reference Manual.* Centro de Informática, Universidade do Porto.

Leslie Lamport. 1986. *LaTeX, A Document Preparation System.* Addison-Wesley.