LMT at Tivoli Gardens

Arendse Bernth and Michael McCord
IBM T. J. Watson Research Center
P.O.B. 704, Yorktown Heights, NY 10598
arendse@watson.ibm.com mccord@watson.ibm.com

Abstract

This paper reports on certain aspects of English→Danish machine translation, using a newly redesigned version of LMT that is implemented in C and is capable of translating Web pages. We use the translation of a Web page about Tivoli Gardens as a springboard for discussion of LMT's treatment of some phenomena of English→Danish translation, with emphasis on the contributions of a new transformational system for LMT.

1 Introduction

We would like to take you to Tivoli Gardens, even though it is midwinter. Our group at IBM Research has been working on a new version of the MT system LMT (McCord, 1989a, 1989b; Bernth and McCord, 1991), and we are applying it currently to the translation of Web pages. We will illustrate a new English→Danish LMT prototype on the translation of a nice Web page about Tivoli Gardens.

For several years, the LMT system was developed in Prolog. The Personal Translator English↔German LMT product (Lehmann, 1995) is implemented largely in Prolog. However, three years ago our group at IBM Research decided to redo our natural language software in C. This move was originally prompted by a desire for increased speed and greater portability; but it turned out to be a great excuse for making various design improvements, even though the overall structure and theory remain the same.

First the Slot Grammar system (McCord, 1980, 1990, 1993) and our grammar checker EasyEnglish (Bernth, 1997) were rewritten, and then the group (including Sue Medeiros) turned to redoing the LMT shell. We have concentrated so far mainly on the English→German version (with Claudia Gdaniec as our German expert), but have also done a partial porting of the existing Prolog English→Danish version. (The Prolog version was begun by Bernth and developed more fully by Hanne Jensen.) The group will soon turn to the development of other language pairs in the new framework, since by now the core engine is rather well developed.

In this paper, we will describe some central aspects of the new LMT system through the English→Danish version (EDLMT). As opposed to some of the other LMT language pairs, e.g. English→German, EDLMT may still be considered a "toy" system; for instance the transfer lexicon is fairly small, about 8000 lemmas. Nevertheless, we can give a good illustration of several technical features of the new LMT in the context of English→Danish.

LMT is a transfer system comprising four major steps: Source analysis, lexical transfer, transformations, and target morphological generation. We will not attempt, in this short paper, to describe the whole system. The transformational step is the most interesting for present concerns, and we will emphasize it in our examples.
The transformational phase uses a newly designed transformational formalism. It is a language in its own right, without any "escapes" to an underlying programming language. It has a Lisp-like syntax, and is interpreted by our C programs in the LMT shell. The transformational formalism is described in detail in (McCord and Bernth, 1998); here we shall only give a brief description in the context of our examples for EDLMT.

2 Web Page Translation

The new C version of LMT can process complete HTML files (as well as documents in other mark-up systems), identifying the natural language segments and replacing them by their translations. The treatment of segment-internal HTML tags and punctuation ("mark-up") is non-trivial, since (1) the target language may have different constituent order and the mark-up in the target output must be placed appropriately, and (2) the meaning of segment-internal mark-up (e.g. a quote) may depend on segment-external mark-up.

Together with W. Rubin, we have developed "WebLMT", which allows users of a Web browser to get on-the-fly translations of Web pages using a proxy server on which LMT is running. The user can select the target language (so far, English is the only source language). WebLMT shows both the source and target pages in a 2-frame browser window. All of this can be done without any additional installation of software on the user's machine.

Figure 1 shows the WebLMT translation for English→Danish of our Tivoli Web page. The URL is:

http://www.danbbs.dk/~ais/copenhagen/tivoli.html

This Web page was created by Hans-Henrik T. Ohlsen, and is used here with his permission.

For the remainder of this paper, we will use the text of our example as a means of illustrating various features of EDLMT. We organize the discussion in terms of the linguistic phenomena that are treated by EDLMT—which comprise some of the main problems in English→Danish translation. The point is not to break any new linguistic ground, but to show how these phenomena are treated by LMT, with emphasis on the transformational component.

3 Treatment of Definite Noun Phrases

Our first problem is the treatment of Danish definite noun phrases—corresponding roughly to English noun phrases with a definite article determiner ("the"). The Danish NP may have an overt definite article determiner or a clitic on the head noun of the noun phrase. We will not deal with the differences between English and Danish in the usage of definiteness, but demonstrate our treatment of definites once definiteness has been established. Definiteness in the Danish NP may be introduced because of a definite article in the English, in which case it is introduced by lexical transfer; or it may be introduced for other reasons, by a transformation.

Whether the Danish NP uses a clitic or not depends on a number of factors. The basic case is clitic and appears when the NP has no adjectival premodifiers (with a few exceptions such as "hel" ("whole")) and no restrictive relative clause postmodifiers; otherwise an overt definite article is used in Danish. See e.g. (Diderichsen 1987; Koefoed 1987; Jones and Gade, 1985).

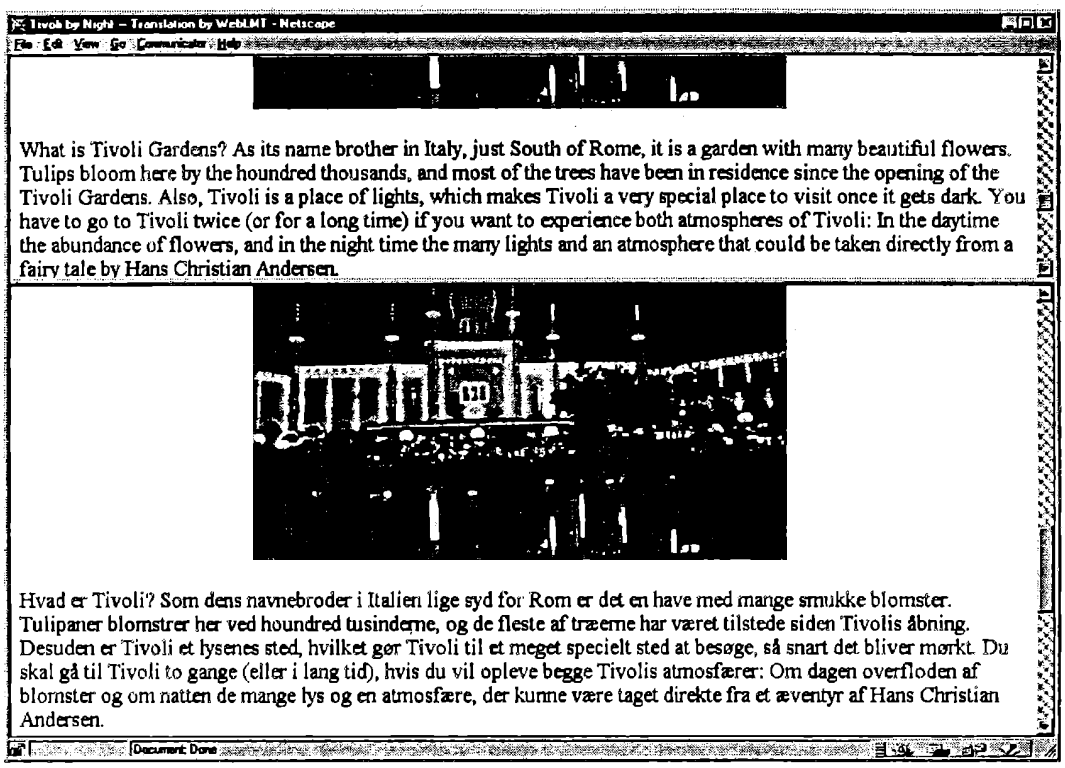

Figure 1. Web Page Translation by WebLMT

The issue is exemplified in just about every sentence of the Web page. Here is one noun phrase that shows both the clitic and the non-clitic cases:

"the abundance of flowers, and in the night time the many lights" ⇒

"overfloden af blomster og om natten de mange lys".

The factors determining whether to use the clitic or the non-clitic determiner are structural, and specific to the target language (Danish). In LMT, such issues are handled by the transformations, which can explore, test, and change the tree structure as it is converted from the source language form to the target language form. The trees operated on by the transformations are expressed in terms of target language words and features (resulting from lexical transfer), but they begin with source language tree structure.

One of the features shared across a noun phrase (but not across coordination) is the feature "Definite". During lexical transfer, "Definite" is set to true (meaning a clitic will be used) as the default, and the actual definite determiner is zeroed out (by default) by setting one of its features, the feature "IsZero", to true. If necessary, the transformational system will later change these defaults, using a transformation called **noncliticdet**.

We will show here a somewhat simplified version of **noncliticdet**. It has two parts: one dealing with the case of a premodifying adjective, and one for the restrictive relative clause. These two rules could be combined into one rule with a disjunction, but it is easier to read separate clauses, which will be tried in order of appearance until one succeeds, or there are no more clauses for the rule (as in Prolog clauses with cuts).

Here is the clause for the case of a premodifying adjectival, to be explained below:

```
noncliticdet
   (tf ((POS . noun)))
   (tm ( *v1
         (v2 (th d))
         (v4 (ts nadj))
         *v5
         head
        *v6 ) )
   (cf1 v2 (IsZero . F))
   (cf1 (Definite . F))
```

The transformational algorithm will attempt to apply this transformation to every node. As it is tried for a given node n, we call n the *node in focus*.

This transformation has four top-level operations, with operators tf, tm, cf1, and cf1. In the name of an operation, a beginning "t" indicates a *test*, and a beginning "c" indicates a *change*.

After that (in the operation name), we use "f" for features, "m" for modifiers, "h" for head word (citation form), and "s" for slot (filled by the node in question). A variable name is a "v" followed by an integer, and the prefix "*" on a variable signals that the variable is a *sublist variable*, which can match any sublist of a modifier list.

The first operation, with operator tf, tests that the node in focus has part of speech (POS) noun. Generally, tf can take any list of feature-value pairs, and can not only test, but also assign values to transformation variables.

The second operation, with operator tm ("test modifiers"), gets to the heart of the matter. Its argument is a pattern that is matched against the modifier list of the NP. The pattern has six pattern elements, displayed vertically here.

The first pattern element, *v1, matches any predeterminers (a sublist variable may match nil).

The second pattern element, (v2 (th d)), illustrates a general feature of the pattern language. A pattern element may be of the form

(Var Op1 Op2 ...)

where the variable Var matches a modifier node and the operations Opi act as tests (and must succeed), as if the matched node were the node in focus. These operations can recursively invoke the full power of the transformation language. The variable Var may also be a sublist variable, in which case the operations are tried on every node in the matched sublist. In the present case, there is one operation, (th d), which tests that the node matched has head word "d",

which happens to be the form that the transfer lexicon gives to the Danish definite article before inflection. So this pattern element just matches the definite article.

The third pattern element, (v4 (ts nadj)), requires v4 to match a node which fills the nadj slot. The nadj slot can be filled by adjectives, verb participles, and numbers. (There are additional possibilities, which we will leave out here for the sake of simplicity.) So this pattern element finds a key thing for the transformation, an adjectival premodifier.

After this, we have any modifiers *v5, followed by the head word, followed by any postmodifiers *v6.

If this modifier pattern succeeds, then we continue and the first cf1 operation will apply. This operation, (cf1 v2 (IsZero . F)), looks at the determiner node v2 and changes its IsZero feature to F (false). (Recall from above that the default setting (before transformations) is that IsZero = T).

Then we do the next change, (cf1 (Definite . F)). Note that this application of cf1 has only one argument, the feature setting. Generally, when the node argument of an operation is omitted, it is understood to be the node in focus, which in this case is our NP. So this cf1 operation sets the Definite feature of the NP to F.

So if an adjectival premodifier exists, our transformation sees to it that the definite article is overt (IsZero = F) and the head noun will *not* get the (default) clitic (Definite = F).

If this transformation fails on a node, then the next rule for **noncliticdet** is tried on that node. This is the rule that tests for a restrictive relative clause postmodifier.

```
noncliticdet
      (tf ((POS . noun))
      (~ (tsourcepunc ,))

      (tm ( *v1
            v2 (th d))
            *v5
            head
            (v6 (| (ts nrel) (ts nrela)))
            *v7 ) )
      (cf1 v2 (IsZero . F))
      (cf1 (Definite . F))
```

In this version of the rule, we test that the relative clause is *not* preceded by a comma in the English source (via the tsourcepunc operator); this fairly reliably tests that the (English) relative clause is restrictive. The disjunctive test (| (ts nrel) (ts nrela)) on the modifier node v6 tests that v6 is a relative clause.

Note: The relative clause rule for **noncliticdet** is ordered after other transformations that convert certain English NP postmodifiers (like participial clauses) to Danish relative clauses, so that we are really testing for the occurrence of Danish relative clauses, even though the English may have used some equivalent postmodifier.

4 Verb-Second Rule

Danish is a verb-second language—meaning that in an independent finite clause, the finite verb is the second constituent. Since this is a structural issue, it is a good candidate for a transformation.

The **verb-second** transformation is exemplified by

"Som dens navnebroder i Italien lige syd for Rom det er en have med mange smukke blomster." ⇒

"Som dens navnebroder i Italien lige syd for Rom er det en have med mange smukke blomster."

"Desuden Tivoli er et lysenes sted" ⇒ "Desuden er Tivoli et lysenes sted"

Figure 2 shows the output of the main steps in translating the last example: the source analysis tree, the lexical transfer tree, the tree resulting from application of all the transformations, and the target string resulting from Danish morphological generation.

The **verb-second** transformation interchanges the verb "være" (infinitive for "er") and the subject "Tivoli", in order to make "være" second, since there is an initial adverb. Other transformations are involved in this translation.

```
Syntactic analysis no. 1   Evaluation = 2.321000...
──────────────────────────────────────────────────────────────────
┌─── vadv        also1(1)     adv
┌─── subj(n)     Tivoli1(2)   noun propn sg
o─── top         be1(3,2,5)   verb vfin vpres sg vsubj
│┌── ndet        a1(4)        det sg indef
└┼── pred(n)     place1(5,u)  noun cn sg
 │┌─ nprep       of1(6,7)     prep
 └┴─ objprep(n)  light2(7)    noun cn pl
──────────────────────────────────────────────────────────────────

Transfer tree...

──────────────────────────────────────────────────────────────────
┌─── vadv    også    ((POS . adv) (PCS . posi) (HasWh . F) (Case . nom)
┌─── subj    Tivoli  ((POS . noun) (NType . propn) (NClass) (CClass) (Pe
o─── top     være    ((POS . verb) (VInfl . vfin) (Tense . vpres) (Mood
│┌── ndet    en      ((POS . det) (DType . indef) (Person . pers3) (Numb
└┼── pred    sted    ((POS . noun) (NType . cn) (NClass r) (CClass) (Per
 │┌─ nprep   af      ((POS . prep) (Case . dat) (SSlot . nprep))
 └┴─ objprep lys     ((POS . noun) (NType . cn) (NClass o) (CClass) (Per
──────────────────────────────────────────────────────────────────

Restructured tree...

──────────────────────────────────────────────────────────────────
┌── vadv    desuden ((POS . adv) (PCS . posi) (HasWh . F) (Case . nom) (
o── top     være    ((POS . verb) (VInfl . vfin) (Tense . vpres) (Mood .
└── subj    Tivoli  ((POS . noun) (NType . propn) (NClass) (CClass) (Per
│┌─ ndet    en      ((POS . det) (DType . indef) (Person . pers3) (Numbe
└┼─ objprep lys     ((POS . noun) (NType . cn) (NClass o) (CClass) (Pers
 └─ pred    sted    ((POS . noun) (NType . cn) (NClass r) (CClass) (Pers
──────────────────────────────────────────────────────────────────

Desuden er Tivoli et lysenes sted.
```

Figure 2. Steps of LMT Translation

5 Intrasentential Punctuation

Intrasentential punctuation is a highly language-specific issue. For Danish, this is determined by sentence structure (at least in "traditional" Danish punctuation). This is also an ideal candidate

for transformations. The punctuation transformations should be applied at the end of the transformational process, since they need to work on the final structure of the sentence.

The punctuation transformations apply to many of the sentences in the Web page example; even where the English punctuation happens to coincide with the Danish, the punctuation is actually added by a transformation.

One example of the punctuation transformations is shown in:

"Du skal gå til Tivoli to gange (eller i lang tid), hvis du vil opleve begge Tivolis atmosfærer:"

Here a comma is inserted between the main and dependent clauses.

6 Compound Nouns

In Danish, compound common nouns are mostly written as one word with a potential connecting sound (usually "e","s", "er", "en", or "t"). Morphological rules determine the combining forms of nouns, as illustrated by "name brother" → "navnebroder".

A simplified version of the particular heuristic employed here is:

If the noun ends in two consonants, the last of which is not "d", then the combining form is noun+"e".

Of course there are many exceptions to the general rules, and it is not possible to get *all* combining forms right based on general rules alone; however, in the absence of suitable lexical information, rules provide a noticeable improvement in translation quality.

7 Target Word Selection

The LMT transfer lexicon can specify tests that determine the preferred transfer of a given source word (sense), based on context in the source analysis tree. The most common tests are on the *complements* of the source word sense, and these tests may be on semantic types (using a type hierarchy), on morphosyntactic conditions, or on specific words or phrases. However, target selection tests can examine any part of the source tree, and a general mechanism for getting to any part of the tree and making a test is the *path test*.

A path test consists of a "path" (describing a way to get to some part of the tree from the node in question), plus a test to be performed on the destination node. For EDLMT, a path test takes care of translating the relative pronoun "that" into "der" if it is the subject (and not a right conjunct), and into "som" otherwise. This gives a nice variation even though "som" also could be used for the subject.

A simplified version of the entry for "that" shows this. (For the sake of simplicity, all other analyses of this word have been omitted.)

```
that
    < pron sg def
     > if (pth (sf subj) --u (sf nrel)) der
```

```
if (pth --u (sf nrel)) som
else den (png pers3 sg nt)
```

This entry first gives the head word "that", then the source element (preceded by "<"), and then the corresponding target element (preceded by ">"). In the target element, a path test (signaled by "pth") tests whether the word fills the subject slot, and whether its mother ("u" for "up one level") fills a relative clause slot. (The latter is a test that the pronoun is a relative pronoun.) If the test succeeds, the transfer is "der". If it does not succeed, the next test is tried. This just checks whether it is indeed a relative pronoun (but not a subject), in which case the transfer is "som". If it is a pronoun, but not a *relative* pronoun, the translation is "den".

An example is:

"an atmosphere that could be taken" ⇒

"en atmosfære, der kunne være taget"

The Prolog version of LMT made use of a special anaphora resolution module (Lappin and McCord, 1990a, 1990b; Lappin and Leass, 1994); this is not implemented for the C version yet.

8 Unknown Words

Words not found in the lexicon are assumed to be proper nouns and are not translated. Of course, this also applies to spelling errors. The Web page has a spelling error: "by the houndred thousands", which is the source of a bad translation: "ved houndred tusinderne". The translation should have been "i hundredetusindvis".

9 Conclusion

Our trip to Tivoli Gardens has illustrated a number of features of EDLMT directly: Treatment of the definite article by way of a transformation; treatment of difference in word order exemplified by the verb-second rule; intrasentential punctuation; combining forms of nouns; and transfer choices in the lexicon.

In addition to these, a number of other features have been illustrated indirectly. The most notable are the following: Our treatment of HTML files, which allows us to hook up LMT to do on-the-fly translation of Web pages; the high quality of the source analysis by English Slot Grammar, which allows accurate transfer on both a lexical and a structural level; and the Danish generation morphology. These features comprise both language-specific parts and language-independent parts.

Our focus has been on the parts that are specific to the language pair English→Danish, but with emphasis on language-independent methods (transformations and lexicons).

All of these features and methods contribute to the quality and usefulness of LMT.

References

Bernth, A. (1997). "EasyEnglish: A Tool for Improving Document Quality," *Proc. Fifth Conference on Applied Natural Language Processing*, pp. 159-165. Association for Computational Linguistics.

Bernth, A. and McCord, M. C. (1991). "LMT for Danish-English Machine Translation," In C.G. Brown and G. Koch (Eds), *Natural Language Understanding and Logic Programming, III*, pp. 179-194, North-Holland.

Diderichsen, P. (1987). *Elementær Dansk Grammatik*, Gyldendal.

Jones, W. Glyn and Gade, K. (1985). *Danish—A Grammar*, Gyldendal.

Koefoed, H. A. (1987), *Teach Yourself Danish*, Hodder and Stoughton.

Lappin, S. and McCord, M. C. (1990a). "A Syntactic Filter on Pronominal Anaphora for Slot Grammar," *Proceedings of the 28th Annual Meeting of the ACL*, pp. 135-142.

Lappin, S. and McCord, M. C. (1990b). "Anaphora Resolution in Slot Grammar," *Computational Linguistics*, vol. 16, pp. 197-212.

Lappin, S. and Leass, H. (1994). "An Algorithm for Pronominal Anaphora Resolution," *Computational Linguistics*, vol. 20, pp. 535-561.

Lehmann, H. (1995). "Machine Translation for Home and Business Users," *Proc. MT Summit V*, Luxembourg, July 10-13.

McCord, M. C. (1980). "Slot Grammars," *Computational Linguistics*, vol. 6, pp. 31-43.

McCord, M. C. (1989a). "Design of LMT: A Prolog-based Machine Translation System," *Computational Linguistics*, vol. 15, pp. 33-52.

McCord, M. C. (1989b). "LMT," *Proceedings of MT Summit II*, pp. 94-99, Deutsche Gesellschaft für Dokumentation, Frankfurt.

McCord, M. C. (1990). "Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars," in R. Studer (Ed.), *Natural Language and Logic: International Scientific Symposium*, Lecture Notes in Computer Science, Springer Verlag, Berlin, pp. 118-145.