MAT: A Machine Assisted Translation System

K. Narayana Murthy
Department of Computer and Information Sciences,
University of Hyderabad,
Hyderabad, 500 046,
email: knmcs@uohyd.ernet.in

Abstract

This paper describes MAT - a machine assisted translation system that was developed by the author for the Government of Karnataka, a southern state in India.[1] MAT is used for translating English texts into Kannada, the official language of Karnataka. MAT is a parser based translation aid, ideally suited for translating between positional languages like English and Indian languages, which are characterized by a relatively free word order and a very rich system of morphology. MAT is based on the UCSG (Universal Clause Structure Grammar) theory of syntactic analysis developed by the author. This paper starts with a very brief introduction to parsing in UCSG and then goes on to describe the structure of the MAT system in detail. Current performance of the MAT system, transcripts from the system and plans for further work are given in conclusion.

Keywords: Machine Translation, Translation Aids

1 Introduction

It is now fairly clear that fully automatic high quality translation is difficult to realise in practice. Either we lose out on quality or we will have to involve the human translator in the process somewhere. MAT is a machine assisted translation system that provides for a full spectrum of possibilities - from fully automatic generation of raw translations suitable for manual post editing, through semi-automatic translation to almost fully manual translation using the facilities provided by the system. The basic idea is to make the best of both the human and machine capabilities to achieve good translations with minimum time and effort. Apart from a very powerful post editing tool, MAT also comes with dictionaries, a thesaurus, morphological analyser/generator and several other useful tools.

MAT is a parser based translation system. Each sentence in the source text is parsed syntactically before translating. This makes MAT suitable for translating between languages that show a significant variation in sentence structures. MAT was developed especially for translating between English and Indian languages but it can be applied to other languages too (Murthy1996b; Murthy1997c).

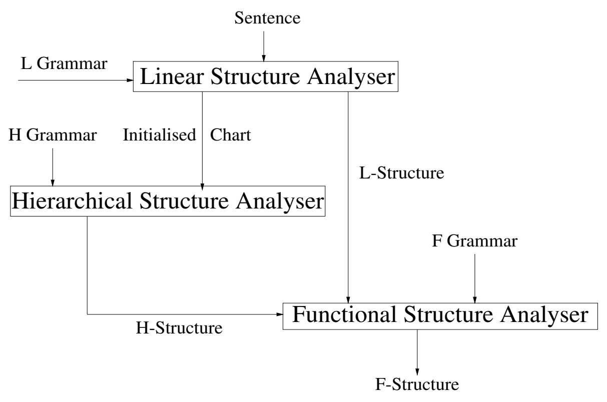
The next section describes briefly the UCSG parsing system that is used in MAT. The following section deals with the translation module. Then we describe the post editing tool. We then go on to briefly sketch the other tools in MAT. We end with conclusions, giving indications of current performance and plans for further work. Sample transcripts from the MAT system are included.

2 Parsing in UCSG

This section describes briefly the UCSG parsing system that is used in MAT. For further details and theoretical issues, the reader may refer to (Murthy1996a; Murthy1997a; Murthy1997b). In UCSG we divide the task of parsing into three modules and we apply the least powerful kind of grammar for each subtask, making the parsing process highly efficient on the whole. Also, this modularisation in UCSG makes it easy to write grammars and to port the system to other languages. Apart from the English parser, experimental parsers for Kannada and Telugu languages have also been built.

[1]The research and development work reported in this paper was supported by the Government of Karnataka.

Grammars can be viewed as sets of constraints on the structure of sentences. There are constraints on the linear position of constituents, hierarchical nesting of constituents one inside the other and functional dependencies between different constituents in a sentence. In UCSG we find that these three primary kinds of structure inherent in human languages namely, linear, hierarchical and functional structures, lend themselves naturally for analysis by three independent modules. UCSG proposes three levels of representation called *L-Structure, H-Structure* and *F-Structure* along with the corresponding components of the grammar and parser. In UCSG we *divide and conquer*. The division of labour into these three separate modules, especially the introduction of an independent H-Structure is the highlight of UCSG. We have shown that this modularization makes UCSG efficient and robust for analysing both positional and free word order languages on an equal footing. The following block diagram gives the overall architecture of UCSG:


Sentence
L Grammar
Linear Structure Analyser
H Grammar
Initialised
Chart
L-Structure
Hierarchical Structure Analyser
F Grammar
Functional Structure Analyser
H-Structure
F-Structure

The L-Module picks up one sentence at a time, breaks it up into words and looks up each word in the dictionary, calling the morphological analyser if and where required. It outputs all potential word groups (or 'phrases') in the sentence. The L grammar is a Nondeterministic Finite State Automaton. It has been shown that all potential word groups in a sentence can be recognized in a single scan of the sentence in linear time - that is, it takes no more than a constant times 'n' amount of time to recognize all the word groups in a sentence with 'n' words (Murthy1996a). All subsequent levels of processing look only at whole word groups, never the individual words. Thus the effective length of the sentence is reduced and the parsing becomes so much more efficient.

UCSG claims that strong constraints exist on the sequences of verbs and certain clause boundary markers called sentinels. The hierarchical nesting of clauses in multi- clause sentences is dictated mainly by these constraints. Noun groups and all the constraints of functional structure play only a secondary and very weak role in determining clause structure. Therefore, in UCSG, we analyse the structure of clauses solely based on verb groups and sentinels. We have shown (Murthy1996a) that we can recognize the various types of clause and understand their inter-relationships before and without applying any of the functional structure constraints. In fact we can determine the clause boundaries also, albeit only partially.

Knowing the clause boundaries has great significance to parsing since the functional structure of a clause is essentially local to that clause. Thus we can *work from whole to part* rather than from left to right. In UCSG, the clause structure is determined in the H-module. This module takes the sequence of verb groups and sentinels (only) as input and produces a parse tree - the clause structure tree using a very small number of *Context Free Grammar* (CFG) rules. UCSG employs an *active chart parsing algorithm* for the H-level. Although the complexity of parsing with general CFGs is $O(n^3)$, here 'n', the input length is much smaller than the length of the sentence. Further, the number of grammar rules required is also very small. Thus the H-module itself is very efficient.

Then the F-module assigns functional roles such as 'subject' and 'object' to various word groups in each clause in the sentence. To do this, we bring to bear simultaneously the top-down constraints in the form of expectations of the verb groups and the bottom-up constraints in the form of available word groups and their grammatical features. All possible functional role assignments that satisfy all the constraints are noted. There is a rating system that rates the role assignments based on a number of hard and/or soft constraints.

After making all possible role assignments, a *best-first search* algorithm picks up the best possible set of role assignments. This scheme has several merits. Firstly, we can hope to get the best parses first without compromising on the ability of the system to generate all the potential parses. If the first parse, or one of the top few parses is found acceptable, the others are not even generated. This is much better than first generating all the parses and then ranking them in some way. Also, a variety of statistical rating schemes can be incorporated easily for better ranking. Finally, this technique gives robustness to the parsing system - even ungrammatical or incomplete sentences can be parsed, albeit with a penalty.

The F-module takes up each clause, starting from the matrix clause and working inwards, down the clause structure tree, passing on expectations and information about the inter-clause dependencies if any. This makes the functional structure analysis of a clause essentially independent of all other clauses in the sentence, thereby reducing the computational complexity of parsing significantly. The H-module has made it possible to assign r roles to r word groups c times for a sentence with c clauses rather than attempting to assign $c * r$ roles to $c * r$ word groups all at once. The additional effort in analysing clause structure is much more than compensated for by this. Parsing in UCSG is thus highly efficient, making it practicable for many applications including machine assisted translation. UCSG has also been used in other areas such as interpreting metaphors (Varma1996). Transcripts given at the end illustrate the whole process.

3 The Translator

In MAT, the parser and translator can be run in one of the three modes called *non-interactive*, *interactive* and *custom* modes. In the non- interactive mode, the system neither asks the user for any help nor does it pause to display intermediate results. This is the simplest and the fastest mode - a full 100 page text can be processed in less than 10 minutes on a personal computer. In the interactive mode, the system stops to ask the user for help in dealing with unknown words, unresolved ambiguities, etc. It also stops to show the parse structure of each sentence. The user can select the correct parse if there are several of them, and translation proceeds after getting confirmation from the user. This mode is very useful for researchers and system developers. We can also expect better quality of translation. In the custom mode, the user can customise the system interface by specifying the kinds the questions the system may ask the user, the intermediate results that should be displayed and/or sent to log file etc. This mode is ideal for testing and development as also for getting a feel for how exactly the system works. A time limit can be specified to instruct the system to skip a sentence if it is taking too much time. Performance of the system is displayed continuously and a summary and a histogram are displayed at the end.

In MAT, like the parser, the translator also works from whole to part, rather than from left to right. After a sentence has been parsed by the UCSG parser, we would know the number, type and inter-relationships amongst the various clauses in the sentence and the word groups that take on various functional roles in each of these clauses. Keeping this structure of the sentence in mind, a suitable structure for the equivalent sentence in the target language is first developed. Where it is not feasible to make more or less direct transfer of structure, sentence transformation rules can be called to restructure the source sentence so that it becomes amenable for translation. This makes it possible to deal with languages with vastly different sentence structures.

After having fixed the overall structure of the target language sentence, the individual clauses are mapped. Finally, the various word groups are mapped. In each word group, the head and the modifiers are identified and transfered, keeping in mind the L-grammar of the target language.

For each word, a suitable target language equivalent is obtained from the bilingual dictionary. The MAT system provides for incorporating syntactic and some simple kinds of semantic constraints in the bilingual lexicon for word sense disambiguation. For example, the word 'rise' is mapped differently into Kannada in the following three sentences:

I *rose*	na:nu *eddenu*
The moon *rose*	caMdra *huTTitu*
The prices are *rising*	belegaLu *heccuttive*

Kannada, the target language in the current experiments, is morphologically very rich. Words in Kannada often take as many as 6 suffixes. In many cases, a whole word group in English translates into a single highly inflected word in Kannada. The MAT system includes a morphological analyzer/generator for Kannada (Murthy1999; Sridhar1990). Using the Kannada morphological generator, appropriate word forms are generated in each case.

Finally, the target language sentence is generated by placing the clauses and the word groups in appropriate linear order, according to the constraints of the target language grammar. For example, in Kannada the subject has to agree with the verb not only in number but also in gender. So gender feature is extracted from the subject of the English source sentence and used in the morphological generation of the Kannada verb so that, for example, 'she came' becomes 'avaLu baMdaLu' and 'he came' becomes 'avanu baMdanu' in Kannada. See transcripts given at the end for more examples.

In case a complete parse with satisfactory rating was not obtained for whatever reason, the word groups with or without the functional roles assigned to them are available for semi-automatic translation. The user picks up parts of the sentence, interactively calls the translator module to translate the part and finally assists the machine in assembling the target language sentence.

4 The Post Editing Tool

The post-editing tool displays the source text and the corresponding translated target language text one sentence at a time. Using the tool, the translator can move around pieces of text easily. He can also delete, insert and edit words at will. More significantly, he can call the thesaurus on line and substitute selected words by their equivalents. *Morphological analysis and generation are done on the fly* so that the correct word forms are substituted automatically. Substitution can be called on both the target and source language words. One can also specifically call the morphological analyser, look at and modify the feature list and then re-generate a new word form. Further, it is also possible to call the translator to translate selected parts of a text. These unique advanced features make it possible to translate full sized English texts with a minimum of effort. As a last resort the user can manually retype the correct translation. High quality translations can be obtained irrespective of the complexity of sentence structures employed and the inherent limitations of the parsing technology. Overall, the post editing tool makes the life of the translator so much easier and gives significant time savings too.

5 Other Tools in MAT

There are separate monolingual dictionaries for English and Kannada as also a bilingual English-Kannada dictionary. A unique feature of this system is that the bilingual dictionary can be used as a kind of thesaurus too. You can get all *related words* for a given word.

There is also a morphological analyser cum generator for Kannada. Kannada words can be analyzed for their internal structure. Specific word forms can also be generated from a given root word. Further, it is possible to get a full paradigm, that is, a systematic listing of all the forms of a given word. Both noun and verb morphology are included. The current version is limited to inflectional morphology - there is neither any derivational component nor rules for inter-word saMdhi nor for compounding.

Internally, Kannada text is represented in Roman notation. However, Kannada input and output can be in ISCII (Indian Standard Code for Information Interchange) format. Inter-conversion tools are provided for use in GIST or Leap Office environments. Through Leap Office you will be able to take the Kannada text into MS-Word also for further formatting and printing. (GIST, Leap Office and MS-Word are Trade Marks of the respective companies).

A 'Pre-Scan' utility is provided to get a list of words from the given text whose entries are not found in the dictionary. All the required entries will be indicated in the appropriate structure in a file called 'custom.dat'. The 'Custom.dat' file is read each time the MAT system is run. So all changes and enhancements made in this file will be effective in all future runs. However, in case you wish to incorporate the entries here directly into the main dictionary, an 'Update-Dictionary' tool is provided.

6 Conclusions

MAT version 1.0 has been completed, delivered and found to be usable. It runs on any IBM or compatible Personal Computer running DOS, Windows 3.1, Windows 95 or Windows NT with at least 16 MB RAM and at least 10 MB free disk space. A simple copy protection scheme is provided.

The MAT system version 1.0 has been tested on several budget speech texts, each of which runs into 100 to 150 pages. MAT system 1.0 can parse and translate about 40 to 60 percent of sentences fully automatically. The translations so produced by the machine are often in more or less acceptable form. Little or no post editing may be required in such cases. Changes required are mostly stylistic in nature. Primary meaning is preserved. However, Some sentences may need substantial editing and in a few rare cases, the outputs may have to be rewritten completely. Where automatic translation fails, semi-automatic translation is possible - user selects parts of the sentence, calls the translator to produce translations for the parts, and finally assembles the parts into the complete target language sentence. In all cases, high quality translations can be obtained quickly using the post editing tool. Overall, the system can bring in substantial time savings.

Further work is in progress to enhance the coverage, robustness, and efficiency of parsing and translation as also to improve the user interfaces.

References

K Narayana Murthy. *Universal Clause Structure Grammar.* PhD thesis, Dept. of Computer and Information Sciences, University of Hyderabad, 1996a.

K Narayana Murthy. Parsing Telugu in the UCSG Formalism. In *Proceedings of the Indian Congress on Knowledge and Language*, volume 2, pages 1–16, 1996b.

K Narayana Murthy. Universal Clause Structure Grammar and the Syntax of Relatively Free Word Order Languages. *South Asian Language Review*, VII(1):47–64, 1997b.

K Narayana Murthy. UCSG and the Machine Aided Translation from English to Kannada. In *Indo-French Symposium on Natural Language Processing, University of Hyderabad*, 1997c.

K Narayana Murthy. A Network and Process Model for Morphological Analysis/Generation. In *Proceedings of the Second International Conference on South Asian Languages*, 1999.

K Narayana Murthy and A Sivasankara Reddy. Universal Clause Structure Grammar. *Computer Science and Informatics*, 27(1):26–38, 1997a.

S N Sridhar. *Kannada.* Routledge, 1990.

Vasudev Varma and A Sivasankara Reddy. Knowledge based metaphor interpretation. *Knowledge-Based Systems*, 9:339–342, 1996.

7 Transcripts from the MAT system

Here are some transcripts from the MAT system 1.0. Clause structure in parse output is indicated through indentation. The first parse produced by the parser has been shown here. Translations shown here are raw translations - no post editing has been done. Note how the order of clauses have been changed during translation as required by the traget language grammar. Gloss in English has been added by the author.

```
1 :  Efforts are being continued to obtain more funds for providing
 relief to the victims. : 14

F_Structure 1 :

Sent 1 : Parse No. 1 : Clause 1 : Rating 0 : Passive : efforts are being
continued to obtain more funds for providing relief to the victims

   2 : subj : empty :: EMPTY :: [] :: []
      3 : subj : empty :: EMPTY :: [] :: []
1 : obj : 2 :: 2 :: [] :: []
   2 : cause :  3 :: 3 :: [] :: []
1 : subj : np :: efforts :: [ng,[n,pl,(rt,effort)]] :: [pl]
1 : vg : vg :: are being continued :: [vgf,[v,aux,be,(rt,be)],[v,aux],[v|_2ADC]] :: [durative,perfective,passive]
   2 : vg : vg :: to obtain :: [vgf,[v]] :: [[(nf2,infinitive)],[(nf2,empty)]]
   2 : obj : np :: more funds :: [ng,[a,cmp,(rt,much)],[n,pl,(rt,fund)]] :: [pl]
      3 : vg : vg :: for providing :: [vgf,[v]] :: [[(nf2,infinitive)],[(nf2,empty)]]
      3 : obj : np :: relief to the victims :: [ng,[n],[prep,(semf,[space,time])],[det],[n,pl,(rt,victim)]] :: [pl]


KANNADA:
toMdarege      oLaga:davarige             pariha:ravannu odagisalu    heccina
difficulty-dat be-subjected-to-people-dat relief-acc     provide-inf  more

nidhigaLannu paDeyalu   prayatnagaLu muMduvarisala:guttive.
fund-pl-acc  obtain-inf effort-pl    continue-passive2-pres-p3-pl

2 : The Government knows that the people of Karnataka will welcome
 this. : 11

F_Structure 1 :

Sent 2 : Parse No. 1 : Clause 1 : Rating 0 : the government knows
that the people of karnataka will welcome this

1 : obj : 2 :: 2 :: [] :: []
1 : subj : np :: the government :: [ng,[det],[n]] :: []
1 : vg : vg :: knows :: [vgf,[v,p3,sl,(rt,know)]] :: []
   2 : cls_link : sentinel :: that :: [] :: []
   2 : subj : np :: the people of karnataka :: [ng,[det],[n,pl],[prep],[n,prp]] :: []
   2 : vg : vg :: will welcome :: [vgf,[v,aux,modal],[v]] :: []
   2 : obj : np :: this :: [ng,[pr,sl,nom,acc,dem]] :: [nominative]


KANNADA:
karna:Takada janaru idannu sva:gatisutta:re       eMdu  sarka:ra tiLiyuttade.
Karnataka-of people it-acc welcome-non-past-p3-pl that  govt.    know-pres-p3-sl
```