A Karaka Based Approach to Parsing of Indian Languages

Akshar Bharati Rajeev Sangal

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur

Kanpur 208 016 India

Abstract

A karaka based approach for parsing of Indian languages is described. It has been used for building a parser of Hindi for a prototype Machine Translation system.

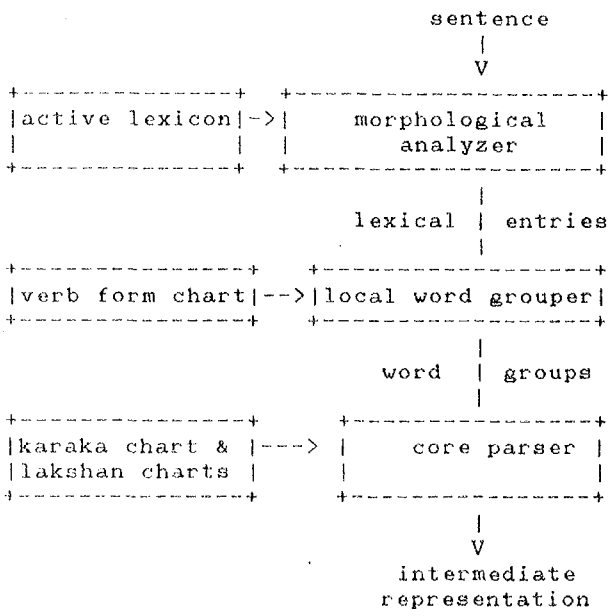
A lexicalised grammar formalism has been developed that allows constraints to be specified between 'demand' and 'source' words (e.g., between verb and its karaka roles). The parser has two important novel features: (i) It has a local word grouping phase in which word groups are formed using 'local' information only. They are formed based on finite state machine specifications thus resulting in a fast grouper. (ii) The parser is a general constraint solver. It first transforms the constraints to an integer programming problem and then solves it.

1. Introduction

Languages belonging to the Indian linguistic area share several common features. They are relatively word order free, nominals are inflected or have post position case markers (collectively called as having vibhakti), have verb complexes consisting of sequences of verbs (possibly joined together into a single word), etc. There are also commonalities in vocabulory, in senses spanned by a word in one language to those of its counterpart in another Indian language, etc.

We base our grammar on the karaka (pronounced kaarak) structure. It is necessary to mention that although karakas are thought of as similar to cases, they are fundamentally different: "The pivotal categories of the abstract syntactic representation are the karakas, the grammatical functions assigned to nominals in relation to the verbal root. They are neither semantic nor morphological categories in themselves but correspond to semantics according to rules specified in the grammar and to morphology according to other rules specified in the grammar" [Kiparsky, 82].

Before describing our grammar formalism, let us look at the parser structure:



Function of the morphological analyzer is to take each word in the input sentence and extract its root and other associated grammatical information. This information forms the input to the local word grouper (LWG).

2. Local Word Grouper (LWG)

The function of this block is to form the word groups on the basis of the 'local information' (i.e., information based on adjacent words) which will need no revision later on. This implies that whenever there is a possibility of more than one grouping for some word, they will not be grouped together by the LWG.

This block has been introduced to reduce the load on the core parser resulting in increased efficiency and simplicity of the overall system.

The following example illustrates the job done by the LWG. In the following sentence in Hindi:

ladake adhyapak ko haar pahana rahe hein
लड़के अध्यापक को हार पहना रहे हैं।
boys teacher to garland garland -ing
(Boys are garlanding the teacher.)

the output corresponding to the word 'ladake' forms one unit, words 'adhyapak' and 'ko' form the next unit, similarly 'pahana', 'rahe' and 'hein' will form the last unit.

3. Core Parser

The function of the core parser is to accept the input from LWG and produce an 'intermediate language' representation (i.e parsed structure along with the identified karaka roles) of the given source language sentence. The core parser has to perform essentially two kinds of tasks

1) karaka role assignment for verbs
2) sense disambiguation for verbs and nouns respectively.

For translating among Indian languages, assignment of karaka roles is sufficient. One need not do the semantic role assignment after the karaka assignment.

Let us now look at the grammar.

3.1 Grammar Formalism

The notion of karaka* relation is central to the model. These are semantico-syntactic relations between the verb(s) and the nominals in a sentence. The computational grammar specifies a mapping from the nominals and the verb(s) in a sentence to karaka relations between them. Similarly, other rules of grammar provide a mapping from karaka relations to (deep) semantic relations between the verb(s) and the nominals. Thus, the karaka relations by themselves do not give the semantics. They specify relations which mediate between vibhakti of nominals and verb form on one hand and semantic relations on the other [Bharati, Chaitanya, Sangal, 90].

For each verb, for one of its forms called as basic, there is a default karaka chart. The default karak chart specifies a mapping from vibhaktis to karakas when that verb-form is used in a sentence. (Karaka chart has additional information besides vibhakti pertaining to 'yogyata' of the nominals. This serves to reduce the possible parses. Yogyata gives the semantic type that must be satisfied by the word group that serves in the karaka role.)

When a verb-form other than the basic occurs in a sentence, the applicable karaka chart is obtained by taking the default karaka chart and transforming it using the verb type and its form. The new karaka chart defines the mapping from vibhakti to karaka relations for the sentence. Thus, for example, 'jotata hai' (ploughs) in A.1 has the default karaka chart which says that karta takes no parsarg (Ram). However, for 'jota' (ploughed) in A.2, or A.4, the karaka chart is transformed so that the karta takes the vibhakti 'ne' 'ko' or 'se',

A.1 Ram khet ko jotata hai.
राम खेत को जोतता है।
Ram farm ko-parsarg plough -s.
(Ram ploughs his farm.)

A.2 Ram ne khet ko jota.
राम ने खेत को जोता।
Ram ne- farm ko- ploughed.
(Ram ploughed the farm.)

A.3 Ram ko khet jotana pada.
राम को खेत जोतना पड़ा।
Ram ko- farm plough had-to.
(Ram had to plough the farm.)

*Here, we use the word 'karaka' in an extended sense which includes 'hetu', 'tadarthya' etc. in addition to actual karakas.

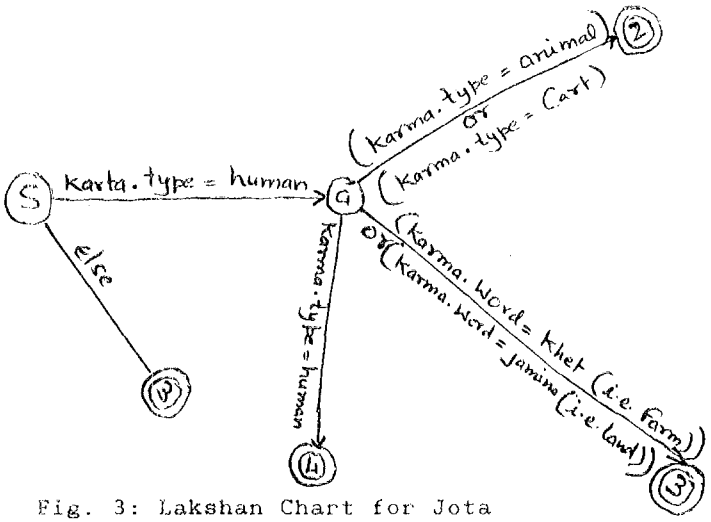


Fig. 3: Lakshan Chart for Jota

Finally, besides the merged karaka charts associated with individual verbs, there is also a global table of common karakas. It pertains to adhikarana karaka (time and place), hetu (cause), etc. and is applicable to all the verbs. It can be used to account for source word groups that remain after satisfying the mandatory karakas. In this sense, it only contains optional karakas.

3.3 Parsing

For the task of karaka assignment, the core parser uses the fundamental principle of 'akanksha' (demand unit) and 'yogyata' (qualification of the source unit).

The linguistic units which play the role of demand and source word groups can vary depending on the parse cycle. In the case of simple sentences, only one cycle is needed in which verb groups and some special noun groups (e.g. 'paas'(near), 'door'(far) etc.) play the role of demand word groups, and noun groups and predicative adjectives play the role of source word groups.

During the parsing process, each of the source word groups may be tested against each of the karaka restrictions in each of the karaka charts of the demand word groups. An appropriate data structure may be created storing the source word groups and the karak restrictions (in karaka charts of demand groups) they satisfy. We call each such entry as a candidate variable.

Typically, a number of source word groups will qualify for a particular demand. The job of the core parser is to make an appropriate assignment of the candidates, subject to certain constraints such as the following:

1) one candidate source word group cannot satisfy more than one demand of the same demand word.

2) every obligatory demand must be satisfied in some karaka chart of every demand word group.

3) every source word must have an assignment.

4) if more than one interpretation of a source word is available, then exactly one has to be selected.

The above problem is transformed to an integer programming problem. Assigning 1 to a candidate variable means that the particular karaka relation between the source word group and the demand word group holds; 0 stands for otherwise. All the various types of constraints mentioned above can be specified in a very natural manner using algebraic inequalities in integer programming. Having a set of candidate variables assigned to 1 not only identifies the karaka relations which can be used to get the deep cases, but also identifies the karaka chart which serves to identify the sense of the verb group, etc.

Moreover Integer programming also permits a linguist to express preferences among various candidates for a particular demand. A typical example of such a preference can be given. For example, for most of the verbs an animate thing is more likely to be the karta than inanimate things, and among animates human beings are more likely candidates to be karta than non-human candidates. These preferences would simply order the multiple parses if any in the absence of other information.

The parsing strategy actually adopted in the system makes use of the merged karaka chart and corresponds to Anvit-Abhidhanvad, a theory of mimamsa school of the Indian grammatical tradition. In this approach, we first determine the karaka relationships among the demand and source word groups. (These are determined

A.4 Ram se khet nahi jota gaya.
राम से खेत नहीं जोता गया।
Ram se- farm not plough could.
(Ram could not plough the farm.)

The above principle allows us to deal with active passives. The verb forms for active and passive are just two special cases of the forms a verb can take.

For example, the verb 'jota' in Hindi has four different meanings listed in the dictionary:

1) harness (e.g., Ram ne bail ko kolhu me jota, or Ram harnessed the bullock for (turning) the crusher.)
राम ने बैल को कोल्हू में जोता।
2) hitching the cart (e.g., Ram ne gaadii ko jota, or Ram hitched the cart.) राम ने गाड़ी को जोता।
3) plough (e.g., Ram ne jamindar ka khet jota, or Ram ploughed the landlord's farm.) राम ने जमींदार का खेत जोता।
4) exploit (e.g., Ram ne naukar ko kaam me jota diya, or Ram exploited his servant by putting him to (hard) work.) राम ने नौकर को काम में जोत दिया

For each of the four senses, a karaka chart can be created. A karaka chart specifies the mandatory karakas (i.e., which must be filled for the sentence to be grammatical), optional karakas, and desirable karakas. For each of the karakas, it specifies the vibhakti (i.e., inflection or post position marker), and the semantic specification (typically in the form of semantic type) to be satisfied by the source word (group). Such a specification for a karaka in a karaka chart is called a karaka restriction. Thus, the karaka chart for the 'hitching' sense of 'jota' has two mandatory karaka restrictions: one for karta karaka (pronounced kartaa kaarak) and the other for karma karaka (pronounced karm kaarak). The former karaka relation maps to agent and the latter to patient semantic relation. As shown in Fig. 1, the restriction for karta karaka says that a source word group satisfying it must be present in the sentence, its vibhakti must be 0, and its semantic type should be human.

restriction on karta karaka:
karaka: karta
mandatory: yes
vibhakti: 0
semantic expression: human

restriction on karma karaka:
karaka: karma
mandatory: yes
vibhakti: 0-or-ko
semantic expression: cart

Fig. 1: Karaka Chart for Jota (Sense 2)

3.2 Refining the Grammar Model

The actual grammar we use in the system is based on the model discussed above. However, it differs from it slightly so as to have a faster parser.

Instead of a separate karaka chart for each sense of a verb, we have a single merged karaka chart. It consists of a set of karaka restrictions where a restriction for a particular karaka relation is obtained by taking the logical-or of the necessary vibhakti and semantic types for the same karaka relation in the different karaka charts. For example, semantic type in restriction for karma karaka for the merged karaka chart is obtained by taking logical-or of semantic types in karma karaka restrictions in the different karaka charts. Fig. 2 shows the merged karaka chart for jota.

Karaka	Necessity	Vibhakti	Semantic Type
karta	m	0	animate
karma	m	0-ko	animate or instrument or land
karana	d	se-dvara	animate or instrument

Fig. 2: Merged Karaka Chart for Jota

As the separate karaka charts are no longer available for distinguishing among the senses of the main verb, separate information is needed. This information is available in the form of lakshan charts or discrimination nets. These nets can be obtained by looking at the separate karaka charts and identifying features that help us in distinguishing among the different senses. An example lakshan chart for jota is given in Fig. 3.

by testing the source word groups against karaka restrictions in the merged karaka chart, and then solving the integer programming problem.) The word meaning is determined only later using the lakshan charts on the karaka assignment.

4. Conclusions

The major features of our approach can be summarized as follows:

1) a parsing strategy based on 'akanksha' (demand) and 'yogyata' (qualification of the source unit). Note that the karaka charts expressing restrictions as above are similar to sub-categorization and selectional restrictions, but are not identical to them. Sub-categorization refers to deep cases, and selectional restrictions usually specify semantic types. Here we use karaka relations, and specify not just semantic types but also post-position markers. It should, of course, be noted that these ideas play a central role in our grammar and parser.

2) a parsing strategy that uses merged karaka chart to do karaka assignment, and only later does the sense selection for nouns and verbs using lakshan charts.

3) formulation of the core parsing problem as integer programming problem. It should be noted that integer programming is a general purpose technique making a large amount of power and flexibility available to the parser. This is at the cost of efficiency if the number of variables to be handled simultaneously is large (though our current parser runs fairly fast). We are engaged in building a special constraint solver that will use this power only when necessary [Ramesh,90].

The grammar and the parser described above are part of a machine translation system for Indian languages based on an interlingua [Sangal & Chaitnya, 87]. Generator in the system uses the same grammar. In principle, each of the stages of the parser is reversed [SenGupta, 89].

Acknowledgement

We would like to acknowledge the principal source of ideas in this paper: Dr. Vineet Chaitanya.

References

[Bharati, Chaitanya & Sangal, 90] A Computational Grammar for Indian Language Processing, A. Bharati, V. Chaitanya, and R. Sangal, Technical Report TRCS-90-96, Dept. of Computer Sc. & Engg., I.I.T. Kanpur, 1990.

[Kiparsky,82] Some Theoretical Problems in Panini's Grammar, P. Kiparsky, Bhandarkar Oriental Research Institute, Pune, 1982.

[Ramesh, 90] Constraints in Logic Programming, P.V. Ramesh, M.Tech. thesis, Dept. of Computer Sc. & Engg., I.I.T. Kanpur, Mar. 1990.

[Sangal & Chaitanya, 87] An Intermediate Language for Machine Translation: An Approach based on Sanskrit using Conceptual Graph Notation, Computer Science & Informatics, J. of Computer Society of India, 17, 1, pp. 9-21, 1987.

[Sangal, Chaitanya & Karnick, 88] An Approach to Machine Translation in Indian Languages, Proc. of Indo-US Workshop on Systems and Signal Processing, Indian Institute of Science, Bangalore, Jan. 1988.

[Sen Gupta, 89] Some Aspects of Language Generation, Rimli Sen Gupta, M.Tech. thesis, Dept. of Electrical Engg, I.I.T. Kanpur, 1989.