Knowledge Extraction from Hindi Text

Shachi Dave Pushpak Bhattacharyya[1]
Department of Computer Science and Engineering,
Indian Institute of Technology,
Bombay.
pb@cse.iitb.ernet.in

In this paper, we present a unique approach to Knowledge Extraction from Hindi text which *preserves the predicate till the end*. This approach helps in highly accurate analysis of sentences. The analysis produces a semantic net like structure expressed by means of Universal Networking Language (UNL)- a recently proposed interlingua. Extremely varied and complex phenomena of Hindi language have been tackled as the case studies reveal.

Keywords: Semantic Web, Knowledge Extraction, Universal Networking Language, Natural Language Parsing, Predicate Preserving Parser, Semantic Net.

1 Introduction

The internet today has to face the complexity of dealing with *multilinguality*. People speak different languages and the number of natural languages along with their dialects is estimated to be close to 4000. Of the top 100 languages in the world, Hindi occupies the fifth position with the number of speakers being close to 200 million. The information need of this large section of humanity will place its unique demand on the web calling for knowledge processing of Hindi documents on the web.

The Universal Networking Language[1] has been introduced as a digital metalanguage for describing, summarizing, refining, storing and disseminating information in a machine-independent and human-language-neutral form. The UNL represents information, i.e., meaning, sentence by sentence. Sentence information is represented as a hyper-graph having concepts as nodes and relations as arcs. This hyper-graph is also represented as a set of directed binary relations, each between two of the concepts present in the sentence. Concepts are represented as character-strings called *Universal Words (UWs)*. UNL vocabulary consists of:

1. **Universal Words (UWs):** Labels that represent word meaning.
2. **Relation Labels:** Tags that represent the relationship between Universal Words.
3. **Attribute Labels:** Express additional information about the Universal Words that appear in a sentence.

An UNL Expression can be seen as a UNL graph. For example,

jhaona jaao kMpnal ka AQyaxa hO]sanao Apnao inavaasasqaana maoM ek AiQavaoSana Aayaaoijat ikyaa hO.

John jo company mein adhyaksh hai usane apane nivaasasthaan mein ek adhiveshan aayojit kiyaa hai.
John, who is the chairman of the company, has arranged a meeting at his residence.

UNL for the sentence is,
;========== UNL ==========
; John jo company mein adhyaksh hai usane apane nivaasasthaan mein ek adhiveshan aayojit kiyaa hai.

[1] Author for Correspondence

[S]
mod(chairman(icl>post):01.@present.@def,company(icl>institution):02.@def)
aoj(chairman(icl>post):01.@present.@def, John(icl>person):00)
agt(arrange(icl>do):03.@entry.@present.@complete.@pred,John(icl>person):00)
pos(residence(icl>shelter):04, John(icl>person):00)
obj(arrange(icl>do):03.@entry.@present.@complete.@pred,meeting(icl>conference):05.@indef)
plc(arrange(icl>do):03.@entry.@present.@complete.@pred,residence(icl>shelter):04)
[/S]
;==

The UNL graph for the sentence is given in figure 1.

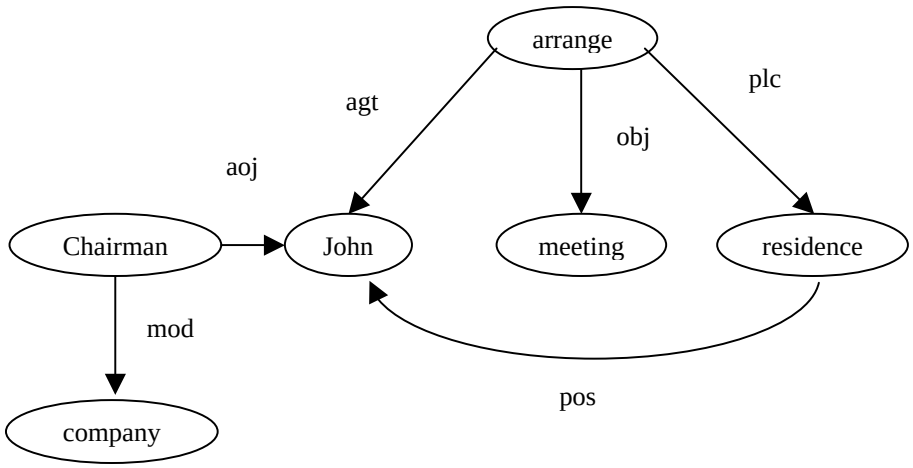

Figure 1: UNL graph

In the above, *agt* means the *agent*, *obj* the *object*, *plc* the *place*, *aoj* the *attributed object* and *mod* the *modifier*. The detailed list of such relations can be found in the reference cited above. Also the *icl* construct helps restrict the meaning of the word to single meaning.

It may be understood from the disussion above that the UNL could be a very effective vehicle for developing multilingual web based applications. The UNL expressions provide the *meaning content* of the text and search could be carried out on this meaning base instead of the text. This of course means developing a novel kind of search engine technology. In fact such a system is already being developed for United Nations' various organizational information (private communication). The merit of such a system is that the information in one language need not be stored in multiple languages. An analysis system for a particular language would extract the meaning content and a generator for another language would produce the information in that language.

Many efforts have been made towards extracting knowledge and representing it in machine understandable forms. Hahn and Schnattinger[2] have proposed a methodology for automating the maintenance of domain-specific taxonomies based on natural language text understanding. A given ontology is incrementally updated as new concepts are acquired from real-world texts. Agichtein and Gravano[3] propose another text knowledge extraction system called *Snowball*. The Snowball system tries to retrieve useful relations from the text. It is based on the DIPRE[4] system which aims for similar tuple extraction from text. It concentrates on generating patterns and extracting exact tuple definitions from the text.

We should mention here the Indian effort at language analysis and generation- especially of Hindi. Many systems have been developed for translation to and from Indian languages. The Anusaaraka system is based on Paninian Grammar[5] and renders text from one Indian language into another. Anusaaraka analyses the source language text and presents the information in a language, which has a flavour of the source language. It tries to preserve information from the input to the output text. For this task, grammaticality is relaxed and special purpose notation is devised wherever necessary. The aim of this system is to allow language access and not machine translation.

IIT Kanpur is involved in designing translation support systems called *Anglabharati* and *Anubharati*[6] from English to Indian languages and vice-versa and among Indian languages. The approach is based on word expert model utilizing *Karaka* theory, pattern directed rule base and hybrid example base.

A project called MaTra[7] , a human-aided translation system for English to Hindi is going on at NCST, Mumbai. The focus is on the innovative use of man-machine synergy. The system breaks an English sentence into chunks, analyses the structure and displays it using an intuitive browser-like representation, which the user can verify and correct, after which the system generates the Hindi. Ambiguities are resolved with user help as are the words not present in the dictionary.

Structured semantic knowledge can be used for a variety of applications. Clark, Thompson, Holmback and Duncan[8] have implemented a system for Boeing which is based on word relationships and uses semantic nets to increase precision of search engine query results. Senniappan and Bhattacharyya[9] have proposed the use of UNL in automatic hyperlink generation. Green[10] has also demonstrated automatic hyperlink generation using semantic similarity.

2 The Analyser System: EnConverter

Knowledge Extraction is first of all a parsing problem[11]. We have used the The EnConverter[12] system (henceforth called *EnCo)* in the UNL Project, Institute for Advanced Studies, United Nations University, Tokyo for our task. EnCo is a language independent parser which provides a framework for morphological, syntactic and semantic analysis synchronously. It analyses sentences by accessing a knowledge rich **lexicon** and interpreting the **Analysis Rules**. The process of formulating the rules is in fact programming a sophisticated symbol processing machine**. Thus Knowledge extraction from Hindi sentences involves constructing analysis rules and building a knowledge rich lexicon linking Hindi words with UWs covering the extremely varied language phenomena and concepts.**

EnCo operates on the nodes of the Node-list through its windows. There are two types of windows, namely *Analysis Window* and *Condition Window*. Two current focused windows called *Analysis Windows (AW)* are circumscribed by the windows called *Condition Windows (CW)*. Figure 2 shows the structure of EnCo.

EnCo uses the Condition Windows for checking the neighbouring nodes on both sides of the Analysis Windows in order to judge whether the neighbouring nodes satisfy the conditions for applying an Analysis Rule or not. The Analysis Windows are used to check two adjacent nodes in order to apply one of the Analysis Rules. If there is an applicable rule, EnCo adds Lexical attributes to or deletes Lexical attributes from these nodes, and/or creates a partial syntactic tree and/or UNL network according to the type of the rule.

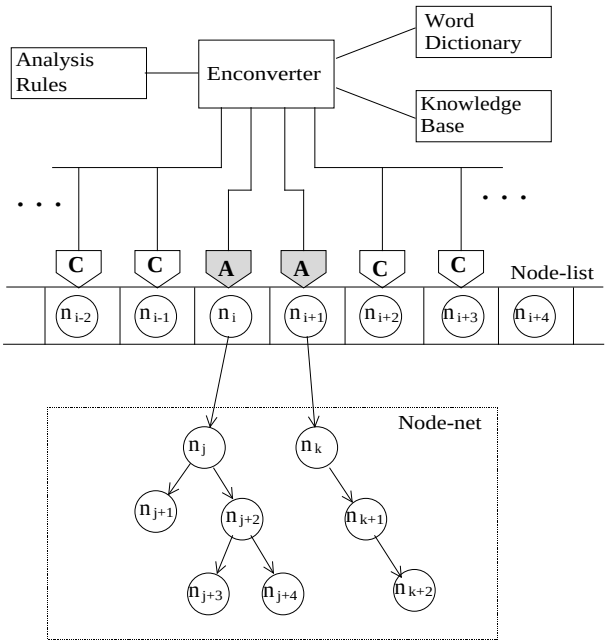

Figure 2: Structure of EnConverter
"A" indicates an Analysis Window, "C" indicates a Condition Window, and "n_n" indicates an Analysis Node

The analysis rule has the following syntax (EnConverter 2000) :

<TYPE> (<PRE>)… {<LNODE>} {<RNODE>} (<SUF_1>) (<SUF_2>) (<SUF_3>)… P<PRI>;

where,
<LNODE>:="{" [<COND1>] ":" [<ACTION1>] ":" [<RELATION1>] "}"
<LNODE>:="{" [<COND2>] ":" [<ACTION2>] ":" [<RELATION2>] "}"

For a given analysis rule, it is possible only to insert or delete only one node into or from the Node-list. Here, LNODE stands for the node under the Left Analysis Window and RNODE for that under the Right Analysis Window.

The meaning of a rule is as follows:

In the rules, under the Left Analysis Window there is a node that satisfies <COND1> attributes, and under the Right Analysis Window there is a node that satisfies <COND2> attributes. When there are nodes that fulfil the conditions in <PRE>, <MID>[2] and <SUF> in the order of left, middle and right sides of Analysis Windows respectively, the Lexical attributes in Analysis Windows are rewritten according to the <ACTION1> and <ACTION2>, respectively. And the operations are done on the Node-list depending on the type of the rule shown in field <TYPE>.
<COND1> and <COND2> stand for the list of lexical attributes whose presence (ATTR) or absence (indicated by "^ATTR") cause a rule to be fired. <ACTION> windows are used to add or delete lexical attributes from the nodes under the Analysis Windows dynamically. The <RELATION> fields are used when some semantic analysis needs to be done by producing an UNL relation between the nodes under the Analysis Windows.

[2] <MID> Condition Windows can be assigned only when either one of the nodes on Analysis Windows is described as the insertion of a node.

<PRI> indicates the priority value of the rules, which is in the range of 0-255. The larger number indicates higher priority. The rules without priority values are deemed to have 0 priority.

3 Lexicon

The lexicon used for the system consists of mapping of Hindi words to Universal Words and lexical-semantic attributes describing the words.

Any entry in the Dictionary is put in the following format (EnConverter 2000):

[HW] {ID} "UW" (ATTRIB1, ATTRIB2, ...) <FLG,FRE,PRI>;

where,

```
HW = Head Word
ID = Identification of Head Word (Omitable)
UW = Universal Word
ATTRIB = Attribute
FLG = Language Flag (e.g., E for English)
FRE = Frequency of Headword
PRI = Priority of Headword
```

Some examples of Dictionary entries are given below:

[rAma] {} "Ram" (N,P,MALE,ANI,Na,3SG) <H,0,0>;
[KZUbasurawa]{} "beautiful(icl>state)"(ADJ) <H,0,0>;

The attributes in the lexicon are collectively called *Lexical Attributes* (both semantic and syntactic attributes). The syntactic attributes include the word category- noun, verb, adjectives, *etc.* and attributes like *person* and *number* for nouns and *tense* information for verbs. The semantic attributes are derived from an Ontology DAG.

4 Predicate Preserving Parser

As EnCo scans the input sentence from left to right the following actions are taken at every step: *Morphological Analysis and Decision Making.*

Morphology is concerned with the ways in which words are formed from basic sequences of morphemes. It acts as the crossroads between phonology, lexicon, syntax, semantics, and context. Two types are distinguished[13]:

- Inflectional Morphology
- Derivational Morphology

Inflectional Morphology defines possible variations on a root form, which in traditional grammars were given as *paradigms*, for example, calaa [calaa](walked) , cala rha hO [cal rahaa hai] (is walking), calaogaa [calegaa] (will walk), *etc.* Our approach for implementing Inflectional Morphology is different from the conventional approach. The UWs defined in the lexicon have word roots specified as Universal Words followed by appropriate restrictions while the headword has the *Longest Common Lexical Unit* (LCLU) of all the possible transformations of the word. For example, the dictionary entry for the verb calanaa [calanaa](to walk) is:

[cal]{}"walk(icl>do)"(List of Semantic and Syntactic Attribute)<H,0,0>;

Where, *cal* is the LCLU for *calaa, calegee, calakara, etc.* Word suffixes for nouns, verbs and adjectives are kept in the lexicon. For example,

[aa]{}"aa"(VMOR)<H,0,0>;

EnCo selects the longest matched entry from the lexicon, starting from the first character. Thus, when EnCo consults the dictionary for a particular morpheme say *calaa*, it will be able to retrieve the LCLU (*cal* here) present in the dictionary. Then, the rules, which look ahead and signal that there is a verb suffix ahead, take control and complete the morphological analysis.

In case of decision making, according to the lexical attributes of the nodes under the two Analysis Windows, the parser decides whether they are to be combined into a single headword or a relation is to be set up between them or an UNL attribute is to be generated. While combining or modifying the two nodes, one of the nodes is deleted from the node-list. Rules ensure that the predicate or the entry node of the sentence is never deleted until the end of the analysis.

As in any parsing situation, the major decision is whether to *Shift* or to *Reduce* [3]. Here, *Reduce* refers to deletion of a node from the Node-list after it is no longer required. Like other parsers, PPP also gives greater priority to *Shift* rules. The rule types performing the *Shift* process are **L** and **R,** while the rules performing *Reduce* are the combination (+ or -) and modification rules (< or >). There are several *Reduce* decisions:

- Generation of UNL Attributes
- Generation of Dynamic Lexical Attributes
- Generation of Relations

Generation of relations is the most important decision. According to the Lexical Attributes (both Static and Dynamic) of the nodes under Analysis Windows, the semantic relation between the nodes is decided. For example, if there is a noun under the LAW with semantic attributes- PLACE (as above) and verb under the RAW and, we decide that the relation **plt** needs to be resolved:

```
lnode:[dillee ]{}"Delhi(icl>place)"(N,P,PLACE,CASEADD,PLT)<H,0,0>;
rnode:[jaa rahaa hai ]{}  "go(icl>event)" (V,MORADD,BLKINSERT) <H,0,0>;
```

The rule is:

```
>{N,PLACE,PLT::plt:}{V,^plt:plt::}P20;
```

The rule deletes the noun, as the verb is the entry node of the sentence.

5 Case Study

In this section, we discuss the complexities that arise due to peculiarities in the structure and semantics of Hindi Language and our approach to resolve them in knowledge extraction.

5.1 Assertive Simple Sentences

Assertive simple sentences have only one main clause. The analysis of such sentences is explained below with an example:

kanapur maoM [na idnaaoM bahut zMD hO.

kaanapur mein in dino bahut thand hai.

Kanpur-in these days very cold-is

It is very cold in Kanpur these days.

The Node-list is shown within "<<" and ">>". The Analysis Windows are denoted within "[" and "]". Steps related to morphological analysis are not shown here. The nodes delimited by "/" are those explored and fixed by the system.

/<</[kaanapur]/[mein]/"in dino bahut thand hai"/>>/

The noun of type *PLACE* and case marker *mein* are combined and an attribute *PLC* is added to the noun to indicate that *plc* relation can be formed between the main predicate of the sentence and this noun.

/<</[kaanapur mein]/[in]/"dino bahut thand hai"/>>/

Here we right shift to put the demonstrative pronoun *in* with the succeeding noun. This is based on the observation that a noun always succeeds a demonstrative pronoun.

/<</kaanapur mein /[in]/[dino]/"bahut thand hai"/>>/

At this step, the *mod* relation is resolved between the demonstrative pronoun *in* and the noun d*ino.*

/<</kaanapur mein /[dino]/[bahut]/"thand hai"/>>/

The system right shifts here because there is no combination or modification rule between a noun and an adverb.

/<</kaanapur mein /dino /[bahut]/[thand hai]/>>/

The system recognises *thand hai* as the predicate of the sentence because of the combination with *hai*. So it generates *man* relation between the adverb *bahut* and the predicate *thand hai.*

/<</kaanapur mein /[dino]/[thand hai]/>>/

By looking at the *TIME* attribute of the noun *dino,* the system resolves *tim* relation between *dino* and the predicate.

/<</[kaanapur mein]/[thand hai]/>>/

Similarly by using the *PLACE* attribute of the noun *kaanapur,* the system generates *plc* relation between the noun *kaanapur* and the predicate.

/<</thand hai/[>>]/

A right shift at this point brings the Sentence Tail (STAIL) under the LAW and thus signals the end of analysis. This right shift rule also attaches the attribute *@entry* to the last word left in the Node-list and thus the predicate *thand hai* is preserved till the end. The output of the system is as under:

;============================ UNL ============================
; kaanapur mein in dino bahut thand hai
[S]
mod(day(icl>period):0H.@pl, these:0D)
man(cold(icl>state):0U.@entry.@present, very(icl>intensifier):0N)
tim(cold(icl>state):0U.@entry.@present, day(icl>period):0H.@pl)
plc(cold(icl>state):0U.@entry.@present, Kanpur(icl>place):00)
[/S]
;==

5.2 Interrogative Sentences

Interrogative sentences are detected by the presence of Wh-words like @yaa [kyaa](what),khaÐ [kahaan](where), @yaaoM [kyon](why), kOsao [kaise](how), iksao [kise](whom) *etc.* and a question mark at the end of the sentence.

These sentences are divided into two categories:

1. Wh-Questions

2. Yes-No Questions

The first type of interrogative sentences like tuma khaÐjaa rho hao? [tum kahaan jaa rahe ho?] (where are you going?) are structurally very similar to simple assertive sentences. If we replace khaÐ [kahaan](where) in the above sentence with a place, say idllal [dillee](Delhi), and remove the question mark then it becomes an assertive sentence. Thus they are analysed using the strategy for simple sentences. In this type of sentences, the Wh-words (*where* in this case) also appear in the final UNL expression. So they are treated like normal nouns or pronouns having a particular attribute (*PLACE* in case of *kahaan*) which helps in deciding its relation with the main predicate of the sentence.

Hindi being a free word-ordered language, the same sentence can be written in more than one way by changing the order of words. For example,

(A) tuma khaÐ jaa rho hao?

tum kahaan jaa rahe ho?

You where going are

(B) khaÐ tuma jaa rho haø

kahaan tum jaa rahe ho?

where you going-are

(C) khaÐ jaa rho hao tuma?

kahaan jaa rahe ho tum?

where going-are you

Where are you going?

The system is equipped with rules that handle all the above cases correctly. All the above sentences produce the following output:

[S]
plc(go(icl>do):07.@entry.@interrogative.@pred.@present.@progress, where(icl>place):00)
agt(go(icl>do):07.@entry.@interrogative.@pred.@present.@progress, you(icl>male):0I)
[/S]

The analysis of second type of interrogative sentences like:

@yaa tuma]sa AaOrt kao jaanato hao ?

kyaa tum oos aurat ko jaanate ho?

what you that woman-to know-is?

Do you know that woman?

is a bit tricky. For its analysis, the valency of the main verb of the sentence is considered. If its valency is satisfied with other nouns or pronouns in the sentence, then the Wh-word is deleted without relating it with any other node. For example, in the above sentence, the valency of the verb jaanato hao [jaanate ho](know) is 1 and it is satisfied with the noun AaOrt kao [aurat ko](woman-to). So the Wh-word @yaa [kyaa](what) is deleted from the sentence. The rest of the sentence then becomes a simple sentence followed by a question mark which can be easily analysed.

5.3 Complex Sentences

The most critical issue in the analysis of complex sentences is to find the words acting as clause delimiters especially in case of nested and chained clauses. The next important issue is to relate the clause to the correct word in the sentence using the appropriate relation. For example, an adverb clause will be related with the verb of another clause while an adjective clause to the noun. We divide complex sentences into two categories:

1. Complex Sentences with Explicit Relative Pronouns
2. Complex Sentences with Implicit Relative Pronouns

5.3.1 Complex Sentences with Explicit Relative Pronouns

The beginning of a clause in such sentences is marked by the presence of relative pronouns like jaao [jo](who), jahaÐ [jahaan](where), ijasao [jise](whom), *etc*., or by the presence of ik [ki](that) for narrative sentences. We will explain the steps involved in the analysis of noun, adverb and adjective clauses below:

(i) Noun clauses:

maOMnao doKa ik saltaa sabjal, KrId rhI hO.

mai ne dekhaa ki seetaa sabjee khareed rahee hai

I saw that sita vegetable buying-is

I saw that Sita is buying vegetables

The system right shifts till it reaches a predicate, d*ekhaa* in this case. It resolves all the relations of the previous nodes with this predicate. Here the **agt** relation is resolved between *mai ne* and d*ekhaa*. The nodelist at this point of analysis is

```
/[<<]/[dekhaa ]/"ki seetaa sabjee khareed rahee hai."/>>/
```

The system then sees a relative pronoun ahead and right shifts. It combines the relative pronoun with the predicate *dekhaa* and adds a dynamic attribute *kiADD* to it using the following rule.

```
+{V:kiADD::}{PRON,KI:::}P30;
```

The clause following *ki* is resolved using the strategy for simple sentences. At the end of its analysis, its main predicate is retained which in this case is *khareed rahee hai*. Thus the nodelist now becomes

```
/<</[dekhaa ki ]/[khareed rahee hai.]/>>/
```

The **obj** relation is generated between these two verbs when the following rule is fired.

```
<{V,kiADD:::}{V:&@entry:obj:}(STAIL)P40;
```

The attribute *kiADD* in the verb *dekhaa* prevents it to be related to any other word than the main predicate of the noun clause.

The final UNL expression generated is:

;========================= UNL ========================

;mai ne dekhaa ki seetaa sabjee khareed rahee hai.
[S]
agt(see(icl>do):07.@entry.@past.@pred, I(icl>person):00)

obj:01(purchase(icl>event):0R.@entry.@present.@progress.@pred, vegetable(icl>food):0K)
agt:01(purchase(icl>event):0R.@entry.@present.@progress.@pred, Sita(icl>person):0F)
obj(see(icl>do):07.@entry.@past.@pred, :01)
[/S]
;==========

Adverb clauses, adjective clauses and Complex sentences with implicit relative pronouns are also dealt with in the system. This is done with variations of the above strategy.

5.4 Compound Sentences:

The difference between the structure of complex and compound sentences is that there are conjunctions connecting two or more clauses instead of relative pronouns. The strategies of complex sentences are applicable. Some examples of compound sentences are:

1. salta ilaK rhI hO AaOr rama Ka rha hO.

   seetaa likh rahee hai aur raam khaa rahaa hai.

   Sita writing-is and Ram eating-is

   Sita is writing and Ram is eating

2. rama Aaja Kanaa banaaegaa @yaaoMik salta ibamaar hO.

   raam aaj khaanaa banaaegaa kyonki seetaa bimaar hai.

   Ram today food cook-will because Sita sick-is

   Ram will cook food today because Sita is sick.

When the Right Analysis Window comes to the conjunction, the conjunction is deleted adding *CONJADD* and an appropriate marker identifying the *type of the conjunction* to the predicate of the previous clause. The rest of the sentence is analysed in the way a clause is analysed. Finally, the predicates of both clauses are resolved into a relation depending upon the conjunction marker. For example, in 2 above we add the marker *RSN* (reason-denoting-conjunction Resolved) to the predicate *banaaegaa* of the previous clause. After *seetaa bimaar hai* is analysed the node list will be:

```
<<[banaaegaa ][bimaar hai.]>>
LAW UW = [cook(icl>do)](V,RSN,CONJADD)
RAW UW = [sick(icl>state)](ADJ, AOJRES)
```

The rule being applied at this point is:

```
<{V,RSN,CONJADD:-RSN::}{ADJ:&@entry:rsn:}P100;
```

The above rule resolves the two predicates into **rsn** relation.

6 Conclusion

The synchronous morphological, syntactic and semantic analysis helps in highly accurate analysis of sentences. Most of the phenomena of the Hindi Language have been tackled. The Predicate Preserving Parser consists of 5200 rules, which include 1200 modification rules for semantic analysis, 800 combination rules mostly for morphological analysis and the remaining shift rules that control the flow of analysis.

The UNL expressions produced by the system are verified manually, as well as by observing the quality of the Hindi sentences generated using the Hindi Generator developed at IIT

Bombay. Currently the system is capable of producing UNL expressions for sentences as complex as the one given below:

maOM baD,I p/sannatapUva^k [sa sadna kao yah batlaanaa caahata hUÐ ik]nhaoMnao hmaoM yah AaSvast ikyaa qaa ik vao Apnao vairYz sahyaaoigayaaoM sao pramaSa^ kr ko [sa samasyaa kao saulaJaanao kI idSaa maoM ivacaar krogaI.

I have great pleasure in informing the Parliament that she had assured us that after consulting her senior colleagues, she will think in the direction of solving this problem.

;========================= UNL ========================
;mai badee prasannataapoorvak is sadan ko yah batalaanaa caahataa hoon ki unho ne hame yah aashvasta kiyaa thaa ki ve apane varishtha sahayogiyon se paraamarsh kar ke is samasyaa ko sulajhaane kee dishaa mein vicaar karegee.
[S]
mod:03(Parliament(icl>body):0V, this:0R)
aoj:03(pleasure:09, great:04)
obj:03(tell(icl>do):1A.@entry.@emphasis.@pred.@present, Parliament(icl>body):0V)
man:03(tell(icl>do):1A.@entry.@emphasis.@pred.@present, pleasure:09)
mod:01(problem(icl>abstract thing):4I, this:4E)
obj:01(solve:4T.@pred, problem(icl>abstract thing):4I)
mod:01(direction(icl>abstract thing):55, solve:4T.@pred)
aoj:01(senior:39,associate(icl>person):3H.@pl)
mod:01(associate(icl>person):3H.@pl, she(icl>person):33)
obj:01(consult(icl>do):3W.@pred, associate(icl>person):3H.@pl)
agt:01(consult(icl>do):3W.@pred, she:30)
scn:01(think(icl>do):5E.@entry.@pred.@future, direction(icl>abstract thing):55)
seq:01(think(icl>do):5E.@entry.@pred.@future, consult(icl>do):3W.@pred)
obj:02(reassure(icl>do):2H.@entry.@pred.@past.@emphasis, :01)
ben:02(reassure(icl>do):2H.@entry.@pred.@past.@emphasis, we:27)
agt:02(reassure(icl>do):2H.@entry.@pred.@past.@emphasis, she(icl<person):1Y)
obj:03(tell(icl>do):1A.@entry.@emphasis.@pred.@present, :02)
agt(want:1J.@entry.@present.@pred, I(icl>person):00)
obj(want:1J.@entry.@present.@pred, :03)
[/S]
;===

The system needs to be enhanced for word sense disambiguation. UNL needs UWs with correct restrictions (sense) and so we need to add sense disambiguation ability to the system. The UNL expressions are currently produced with only one sense per category of a word.

When fully developed, the system will certainly be a step towards realising *the semantic web* for Hindi by providing a powerful knowledge extraction mechanism for Hindi sentences.

Acknowledgements

The technical and conceptual help provided by Ms. Zhu and Dr. Uchida from the UNL Centre at United Nations University, Tokyo is gratefully acknowledged. Part of the research was carried out under the grant for Technology Development in Indian Languages from the Ministry of IT, Government of India.

References:

1. Uchida H., Zhu M., The Universal Networking Language (UNL) specifications version 3.0, *1998*. Technical Report, United Nations University, Tokyo, 1998.

http://www.unl.unu.edu/unlsys/unl/unls30.doc

2. Hahn, U. and Schnattinger, K., Towards Text Knowledge Engineering, 15th National Conference on Artificial Intelligence, Madison, Wisconsin, 1998.
3. Agichtein, E. and Gravano, L., Snowball: Extracting Relations from Large Plain-Text Collections, 5th ACM International Conference on Digital Libraries (DL'00), 2000.
4. Brin, S., Extracting Patterns and Relations from the world wide web, International Workshop on Web and Databases (WebDB'98), 1998.
5. Bharati, A., Chaitanya, V., Kulkarni, A. P. and Sangal R., Language Access : An Information Based Approach, International Conference on Knowledge Based Computer Systems, Mumbai, 2000.
6. Sinha, R.M.K., Machine Translation: The Indian Context, in International Conference on Applications of Information Technology in South Asian Languages, AKSHARA'94, New Delhi, 1994.
7. Rao, D., Mohanraj, K., Hegde, J., Mehta, V. and Mahadane, P, A Practical Framework for Syntactic Transfer of Compound-Complex Sentences for English-Hindi Machine Translation, International Conference on Knowledge Based Computer Systems, Mumbai, 2000.
8. Clark, P., Thompson, Holmback, H., Duncan, L., Exploiting a Thesaurus-Based Semantic Net for Knowledge-Based Search, 12th Conf on Innovative Applications of AI (AAAI/IAAI'2000), 2000.
9. Senniappan, K. and Bhattacharyya, P., Automatic Generation of Hyperlinks using Semantic Information, International Conference on Information Technology (CIT 2000), Bhubaneswar, 2000.
10. Green, S., Building Hypertext links by computing Semantic Similarity, IEEE Transactions on Knowledge and Data Engineering, Vol. 11-5, 1999.
11. Earley, J., An Efficient Context-free Parsing Algorithm, Communications of the ACM, 13(2), 1970.
12. EnConverter Specification Version 2.1, UNU/IAS/UNL Centre, Tokyo 150-8304, Japan. http://www.unl.ias.unu.edu/unlsys/enco/index.html, 2000.
13. Hutchins, W. J. and Somers, H. L., An Introduction to Machine Translation, Academic Press, 1992.