ENGLISH TO SANSKRIT MACHINE TRANSLATOR

LEXICAL PARSER

Ms.Vaishali M. Barkade
Information Technology Department
Bharati Vidyapeeth University College of Engineering
Pune- 43, Maharashtra, India

Prof. Prakash R. Devale
Information Technology Department
Bharati Vidyapeeth University College of Engineering
Pune- 43, Maharashtra, India

Abstract— Here we propose to develop a converter which converts English Sentence to Sanskrit sentence. The Proposed modules are as follows:
MODULE 1: LEXICAL PARSER
MODULE 2: SEMANTIC MAPPER
MODULE 3: ITRANSLATOR
MODULE 4: COMPOSER
Here we would concentrate only on the first module that is Lexical parser which parses a English sentence.

Keywords: machine translation, word order, lexical parser, grammar, tree.

I. INTRODUCTION

Machine translation is one of the most important applications of Natural Language Processing. Machine translation helps people from different places to understand an unknown language without the aid of a human translator**.** The language to be translated is the Source Language (SL). The language to which source language translated is Target Language (TL). The major machine translation techniques are Rule Based Machine Translation Technique [1], Statistical Machine Translation Technique (SMT) and Example-based machine translation (EBMT). One of the effective techniques for machine translation is Rule Based Machine Translation. In India, different machine translation systems are implemented. AnglaUrdu (AnglaHindi based) Machine Translation System for English to Urdu **[**2], HindiAngla Machine Translation Systems form Hindi to English, English-Assamese Machine Translation System (Machine Translation System from English to Assamese, MaTra: Human Aided Machine Translation System, AnglaHindi: An English to Hindi Machine-Aided Translation System [3] and AnglaBharti Technology for machine aided translation from English to Indian Languages[4], these are some of the machine translation works implemented in India. Here we are describing about Machine Translation Technique for translating English to Sanskrit. While translating, the syntactic structure and semantics structure of both source language and target language should be considered. Here we are using the Source language as English and Target Language as Sanskrit. We are concentrating on first module that accepts input as English sentence and we parse it to generate a Semantic tree.

II. COMPARISON OF ENGLISH AND SANSKRIT WORD ORDER

English is well known language and Sanskrit is an ancient language. The English sentence always has an order of Subject-Verb-Object, while Sanskrit sentence has a free word order. A free order language is a natural language which does not lead to any absurdity or ambiguity, thereby maintaining a grammatical and semantic meaning for every sentence obtained by the change in the ordering of the words in the original sentence. For example, the order of English sentence (ES) and its equivalent translation in
Sanskrit sentence (SS) is given as below.

```
ES:  Ram        reads    book.
    (Subject)  (Verb)   (Object)

SS: Raamah   pustakam   pathati.
    (Subject)  (Object)    (Verb)
                  or
    Pustakam   raamah   pathati.
    (Object)   (Subject)  (Verb)
                  or
    Pathati      pustakam    raamah
    (Verb)       (Object)    (Subject)
```

Thus Sanskrit sentence can be written using SVO, SOV and VOS order.

III. PARSER

A parser breaks data into smaller elements, according to a set of rules that describe its structure
Parsing is the process of analyzing a text, made of a sequence of tokens (for example, words), to determine its grammatical structure with respect to a given grammar.
Following are the Steps to generate a Parse Tree
Step 1: Input is a English sentence.
Step 2: Lexical Analyzer
Creates Tokens
Step3: Tokens generated acts as an input to Semantic analyzer
Step 4: Semantic analyzer
Creates a parse tree
Step 5: Output is a parse tree

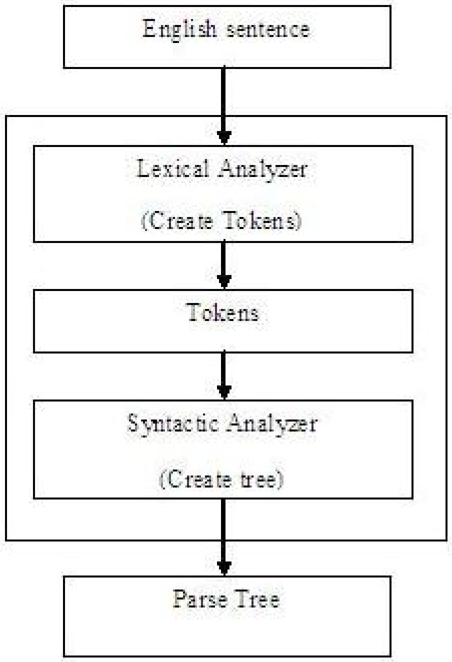

Figure 1 Stages of creating a Parse Tree

IV. LEXICAL PARSER

The semantic standard representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. The sentence relationships are represented uniformly as semantic standard relations between pairs of words. For the sentence:
Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

A. The Semantic representation is:

nsub j (makes-8, Bell-1)

nsubj (distributes-10, Bell-1)

partmod (Bell-1, based-3)

nn (Angeles-6, Los-5)

prep in (based-3, Angeles-6)

conj and (makes-8, distributes-10)

amod (products-16, electronic-11)

conj and (electronic-11, computer-13)

amod (products-16, computer-13)

conj and (electronic-11, building-15)

amod (products-16, building-15)
dobj (makes-8, products-16)

dobj (distributes-10, products-16)

The above relations maps straightforwardly onto a directed graph representation, in which

- NODES - words in the sentence
- EDGES - grammatical relations are edge labels.

Figure 2 gives the graph representation for the example sentence above.

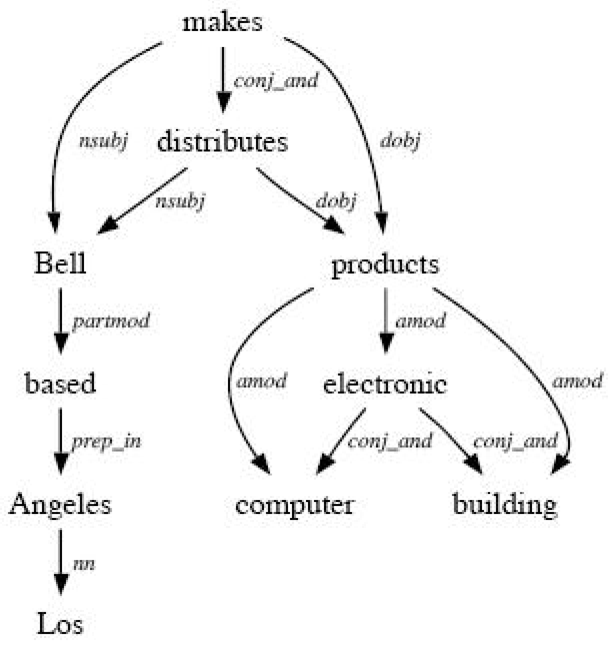

Figure 2
Graphical representation of the Semantic standard for the sentence “Bell, based in Los Angeles, makes and distributes electronic, computer and building products”

B. Definition of Semantic Standard

The dependencies are all binary relations: a grammatical relation holds between a governor and a dependent.
The grammatical relations are as follows

1. abbrev : abbreviation modifier
   An abbreviation modifier of an NP is a parenthesized NP that serves to abbreviate the NP (or to define an abbreviation).
   e.g.
   “The Australian Broadcasting Corporation (ABC)"
   abbrev (Corporation, ABC)

2. acomp : adjectival complement

An adjectival complement of a VP is an adjectival phrase which functions as the complement (like an object of the verb);
an adjectival complement of a clause is the adjectival complement of the VP which is the predicate of that clause.
e.g.
“She looks very beautiful"
acomp (looks, beautiful)

3. advcl : adverbial clause modifier
An adverbial clause modifier of a VP is a clause modifying the verb (temporal clause, consequence, conditional clause, etc.).
e.g.
“The accident happened as the night was falling"
advcl (happened, falling)

4. advmod: adverbial modifier
An adverbial modifier of a word is a (non-clausal) RB or ADVP that serves to modify the meaning of the word.
e.g.
“Genetically modified food"
advmod (modified, genetically)

5. agent: agent
An agent is the complement of a passive verb which is introduced by the preposition “by" and does the action.
e.g.
“The man has been killed by the police"
agent (killed, police)

6. amod : adjectival modifier
An adjectival modifier of an NP is any adjectival phrase that serves to modify the meaning of the NP.
e.g.
“Sam eats red meat" amod(meat, red)
7. appos: appositional modifier
An appositional modifier of an NP is an NP immediately to the right of the first NP that serves to define or modify that NP.
e.g.
“Sam, my brother"
appos (Sam, brother)

8. attr: attributive
An attributive is the complement of a copular verb such as “to be", “to seem", “to appear".
e.g.
“What is that?" attr (is, What)

9. aux: auxiliary
An auxiliary of a clause is a non-main verb of the clause, e.g. modal auxiliary, “be" and “have" in a composed tense.
e.g.
“Reagan has died"
aux (died, has)

10. auxpass : passive auxiliary
A passive auxiliary of a clause is a non-main verb of the clause which contains the passive information.
e.g.
“Kennedy has been killed"
auxpass (killed, been)
aux (killed,has)
11. cc: coordination
A coordination is the relation between an element of a conjunct and the coordinating conjunction word of the conjunct.
e.g.
“Bill is big and honest"
cc (big, and)

12. ccomp : clausal complement
A clausal complement of a VP or an ADJP is a clause with internal subject which functions like an object of the verb or of the adjective; a clausal complement of a clause is the clausal complement of the VP or of the ADJP which is the predicate of that clause. Such clausal complements are usually finite (though there are occasional remnant English subjunctives).
e.g.
“He says that you like to swim"
ccomp (says, like)

13. complm : complementizer
A complementizer of a clausal complement (ccomp) is the word introducing it. It will be the subordinating conjunction “that" or “whether".
e.g.
“He says that you like to swim"
complm (like, that)

14. conj : conjunct
A conjunct is the relation between two elements connected by a coordinating conjunction, such as “and", “or", etc. We treat conjunctions asymmetrically: The head of the relation is the first conjunct and other conjunctions depend on it via the conj relation.
e.g.
“Bill is big and honest"
conj (big, honest)

15. cop: copula
A copula is the relation between the complement of a copular verb and the copular verb.
e.g.
“Bill is big"
cop (big, is)
16. csubj : clausal subject

A clausal subject is a clausal syntactic subject of a clause, i.e. the subject is itself a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb.
e.g. “what she said" is the subject.
“What she said makes sense"
csubj (makes, said)

17. csubjpass: clausal passive subject
A clausal passive subject is a clausal syntactic subject of a passive clause.
e.g. “that she lied" is the subject.
“That she lied was suspected by everyone"
csubjpass (suspected, lied)

18. det: determiner
A determiner is the relation between the head of an NP and its determiner.
e.g.
“The man is here"
det (man, the)

19. dobj : direct object
The direct object of a VP is the noun phrase which is the (accusative) object of the verb; the direct object of a clause is the direct object of the VP which is the predicate of that clause.
e.g.
“She gave me a raise"
dobj (gave, raise)

20. expl: expletive
This relation captures an existential “there". The main verb of the clause is the governor.
e.g.
“There is a ghost in the room"
expl (is, There)

21. infmod: infinitival modifier
An infinitival modifier of an NP is an infinitive that serves to modify the meaning of the NP.
e.g.
“I don't have anything to say"
infmod (anything, say)

22. iobj : indirect object
The indirect object of a VP is the noun phrase which is the (dative) object of the verb; the indirect object of a clause is the indirect object of the VP which is the predicate of that clause.
e.g.
“She gave me a raise"
iobj (gave, me)

23. mark: marker
A marker of an adverbial clausal complement (advcl) is the word introducing it. It will be a subordinating conjunction difierent from “that" or “whether": e.g. “because", “when", “although",etc.
e.g.
“Forces engaged in fighting after insurgents attacked"
mark (attacked, after)

24. measure: measure-phrase modifier
The measure-phrase modifier is the relation between the head of an ADJP/ADVP and the head of a measure-phrase modifying the ADJP/ADVP.
e.g.
“The director is 65 years old"
measure (old, years)

25. neg : negation modifier
The negation modifier is the relation between a negation word and the word it modifies.
e.g.
“Bill is not a scientist"
neg (scientist, not)

26. nn : noun compound modifier
A noun compound modifier of an NP is any noun that serves to modify the head noun.
e.g.
“Oil price futures"
nn (futures, oil)
nn (futures, price)

27. nsubj : nominal subject
A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb.
e.g.
“Clinton defeated Dole"
nsubj (defeated, Clinton)

28. nsubjpass: passive nominal subject
A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.
e.g.
“Dole was defeated by Clinton"
nsubjpass (defeated, Dole)

29. num: numeric modifier
A numeric modifier of an NP is any number phrase that serves to modify the meaning of the NP.
e.g.
“Sam eats 3 sheep"
num (sheep, 3)

30. number: element of compound number
An element of compound number is a part of a number phrase or currency amount.

e.g.
“I lost $ 3.2 billion"
number($, billion)

31. parataxis: parataxis
The parataxis relation (from Greek for “place side by side") is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, a clause after a “:" or a “;".

e.g.
“The guy, John said, left early in the morning"
parataxis (left, said)

32. partmod : participial modifier
A participial modifier of an NP or VP is a participial verb form that serves to modify the meaning of the NP or VP.
e.g.
“Trufies picked during the spring are tasty"
partmod (trufies, picked)

33. mp: prepositional complement
The prepositional complement of a preposition is the head of a clause following the preposition.
e.g.
“They heard about you missing classes"
pcomp (about, missing)

34. pobj : object of a preposition
The object of a preposition is the head of a noun phrase following the preposition. (The preposition in turn may be modifying a noun, verb, etc.)
e.g.
“I sat on the chair"
pobj (on, chair)

35. poss : possession modifier
The possession modifier relation holds between the head of an NP and its possessive determiner,
or a genitive 's complement.
e.g.
“their offices"
poss (offices, their)

36. possessive: possessive modifier
The possessive modifier relation appears between the head of an NP and the genitive 's.
e.g.
“Bill's clothes"
possessive (John, 's)

37. preconj : preconjunct
A preconjunct is the relation between t he head of an NP and a word that is part of a conjunction, an puts emphasis on it (e.g., “either", “both", “neither").
e.g.
“Both the boys and the girls are here"
preconj (boys, both)

38. predet: predeterminer
A predeterminer is the relation between the head of an NP and a word that precedes and clarifies the use of the NP determiner.
e.g.
“All the boys are here"
predet(boys, all)

39. prep/prepc: prepositional modifier
A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, or noun. If the prepositional phrase is a clause, the relation is called prepc when collapsing takes place .
e.g.
“I saw a cat in a hat"
prep (cat, in)

40. prt: phrasal verb particle
The phrasal verb particle relation identifies a phrasal verb, and holds between the verb and its particle.
e.g.
“They shut down the station"
prt (shut, down)

41. punct: punctuation
This is used for any piece of punctuation in a clause, if punctuation is being retained in the typed dependencies.
e.g.
“Go home!"
punct (Go, !)

42. purpcl : purpose clause modifier
A purpose clause modifier of a VP is a clause headed by “(in order) to" specifying a purpose.
e.g.
“He talked to him in order to secure the account"
purpcl (talked, secure)

43. quantmod: quantifier phrase modifier
A quantifier modifier is an element modifying the head of a QP constituent.
e.g.
“About 200 people came to the party"
quantmod (200, About)

44. rcmod: relative clause modifier
A relative clause modifier of an NP is a relative clause modifying the NP. The relation points from the head noun of the NP to the head of the relative clause, normally a verb.
e.g.
“I saw the man you love"
rcmod (man, love)

45. ref : referent
A referent of the head of an NP is the relative word introducing the relative clause modifying the NP.
e.g.
“I saw the book which you bought"
ref (book, which)

46. rel : relative
A relative of a relative clause is the head word of the WH-phrase introducing it.
e.g.
“I saw the man who you love"
rel (love, who)

47. tmod : temporal modifier
A temporal modifier of a VP or an ADJP is any constituent that serves to modify the meaning of the VP or the ADJP by specifying a time; a temporal modifier of a clause is a temporal modifier of the VP which is the predicate of that clause.
e.g.
“Last night, I swam in the pool"
tmod (swam, night)

48. xcomp : open clausal complement
An open clausal complement (xcomp) of a VP or an ADJP is a clausal complement without its own subject, whose reference is determined by an external subject.
e.g.
“He says that you like to swim"
xcomp (like, swim)

49. xsubj : controlling subject
A controlling subject is the relation between the head of a open clausal complement (xcomp) and the external subject of that clause.
e.g.
“Tom likes to eat fish"
xsubj (eat, Tom)

C. Different styles of dependency representation

Four variants of the typed dependency representation are available in the dependency extraction system. The representations follow the same format: a dependency is written as abbreviated relation name(governor, dependent) where the governor and the dependent are words in the sentence to which the word number in the sentence is append. The differences are that they range from a more surface-oriented representation, where each token appears as a dependent in a tree, to a more semantically interpreted representation where certain word relationships, such as prepositions, are represented as dependencies and the set of dependencies becomes a possibly cyclic graph.

1. Basic Typed Dependencies

The basic typed dependencies use the dependencies and form a tree structure. Each word in the sentence (except the head of the sentence) is the dependent of one other word. For the sentence, “Bell, a company which is based in LA, makes and distributes computer products", the basic typed dependencies will be:
nsubj (makes-11, Bell-1)

det (company-4, a-3)

appos (Bell-1, company-4)

rel (based-7, which-5)

auxpass (based-7, is-6)

rcmod (company-4, based-7)

prep (based-7, in-8)

pobj (in-8, LA-9)

cc (makes-11, and-12)

conj (makes-11, distributes-13)

nn (products-15, computer-14)

dobj (makes-11, products-15)

2. Collapsed dependencies

In the collapsed representation, additional dependencies are considered, even ones that break the tree structure turning the dependency structure into a directed graph. So in the above example, the following relations will be added:
ref(company-4, which-5)
nsubjpass(based-7, which-5)

These relations do not appear in the basic representation since they create a cycle with the rcmod and rel relations. Relations that break the tree structure are the ones taking into account elements from relative clauses and their antecedents (as shown in this example), as well as the controlling (xsubj) relations. Moreover dependencies involving prepositions, conjuncts as well as information about the referent of relative clauses are collapsed to get direct dependencies between content words. This “collapsing" is often useful in simplifying patterns in relation extraction applications. For instance, the dependencies involving the preposition “in" in the above example will be collapsed into one single relation:

prep(based-7, in-8)

pobj(in-8, LA-9)

will become

prep in(based-7, LA-9)

The same happens for dependencies involving conjunction:

cc(makes-11, and-12)

conj(makes-11, distributes-13)

will become

conj and(makes-11, distributes-13)

The information about the antecedent of the relative clause

(ref(company-4, which-5))

will serve to expand the following dependency:

nsubjpass(based-7, which-5)

becomes

nsubjpass(based-7, company-4)

In the end the collapsed dependencies that the system gives you for the sentence are:

nsubj (makes-11, Bell-1)

det (company-4, a-3)

appos (Bell-1, company-4)

nsubjpass (based-7, company-4)

rel (based-7, which-5)

auxpass (based-7, is-6)

rcmod (company-4, based-7)

prepfiin (based-7, LA-9)

conjfiand (makes-11, distributes-13)

nn (products-15, computer-14)

dobj (makes-11, products-15)

3. Collapsed dependencies with propagation of conjunct dependencies.

When there is a conjunction, you can also get propagation of the dependencies involving the conjuncts. In the sentence here, this propagation will add two dependencies to the collapsed representation; due to the conjunction between the verbs “makes" and “distributes", the subject and object relations that exist on the first conjunct (“makes") will be propagated to the second

conjunct (“distributes"):

nsubj(distributes-13, Bell-1)

dobj(distributes-13, products-15)

Since this representation is an extension of the collapsed dependencies, it does not guarantee a tree structure.

4. Collapsed dependencies preserving a tree structure

In this representation, dependencies which do not preserve the tree structure are omitted. As explained above, this concerns relations between elements of a relative clause and its antecedent, as well as the controlling subject relation (xsubj). This also does not allow propagation of conjunct dependencies. In our example, the dependencies in this representation will be:
nsubj (makes-11, Bell-1)

det (company-4, a-3)

appos (Bell-1, company-4)

rel (based-7, which-5)

auxpass (based-7, is-6)

rcmod (company-4, based-7)

prepfiin (based-7, LA-9)

conjfiand (makes-11, distributes-13)

nn (products-15, computer-14)

dobj (makes-11, products-15)

V. WORK DONE BASED ON THIS

In this work we have design our own lexical parser for getting the typed dependency information POS tag information and context-free phrase structure grammar representation of source structure. Translation based on above method is implemented with the help of Java codes. The nouns, verb, subject etc are stored in a Data structure i.e. Collection (Array List). We have a lexical parser on Natural Language Parser we have customized and abstracted parser algorithm based on our requirement. This parser is used at lower level of our application.
Step 1: Tokenize the sentence into various tokens i.e. token list

Step 2: To find the relationship between tokens we are using dependency grammar and binary relation for our English language Token list acts as an input to semantic class to represent the semantic standard.

Step 3: Semantic class generates a tree we have a class Tree Transform which will create a tree.

Step 4: Sentence is splitted into words that are nouns, verbs etc.

The sentence :

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

is broken into tokens.

Noun (nsubj)	Token1	Bell
Participial modifier (partmod)	Token 2	based
Preposition (prep)	Token 3	in
Noun (nn)	Token 4	Los
Noun (nn)	Token 5	Angeles
Verb	Token 6	makes
Conjunction (conj)	Token 7	and
Verb	Token 8	distributes
Adjective (amod)	Token 9	electronic
Adjective(amod)	Token 10	computer
Conjunction (conj)	Token 11	and
Adjective (amod)	Token 12	building
Directobject (dobj)	Token 13	products

Table 1 Shows how a sentence is broken into tokens

VI. CONCLUSION

Here more emphasis is on Module 1 LEXICAL PARSER. It shows how an English statement is parsed into tokens and then finds the relationship between tokens using dependency grammar and by using the semantic representation we generate a tree.

VII. REFERENCES

[1] http://en.wikipedia.org/wiki/machinefitranslation
[2] http://tdil.mit.gov.in/tdil-oct-2003/machine%20translation%20system%20.pdf
[3] R.M.K. Sinha, A. Jain ‘AnglaHindi:An English to Hindi Machine-Aided Translation System.’ Indian Institute of Technology, Kanpur, India, 2003
[4] Sinha, R.M.K.; Sivaraman, K.; Agrawal, A.; Jain, R.; Srivastava, R.; Jain ‘ANGLABHARTI: a multilingual machine aided translation project on translation from English to Indian languages’. A.Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century.,IEEE International Conference on Volume 2, Issue, 22-25 Oct 1995 Page(s):1609 - 1614 vol.2
[5] shu-jie liu1, mu-yun yang1,2,tie-jun zhao1 a cascaded approach to the optimization of translation Rules 1- 4244-0060-0/06/$20.00 ©2006 IEEE
[6] Rule Based Machine Translation from English to Malayam Rajan, R.; Sivan, R.; Ravindran, R.; Soman, K.P.; Advances in Computing, Control & Telecommunication Technologies, 2009. ACT’09, International Conference on Digital Object Identifier:10.1109/ACT.2009.113 Publication Year:2009, Page(s):439-441