Abstract

Machine Translation has evolved tremendously in the recent time and stood as center of research interest for many computer scientists. Developing a Machine Translation system for ancient languages is much more fascinating and challenging task. A detailed study of Sanskrit language reveals that its well-structured and finely organized grammar has affinity for automated translation systems. This paper provides necessary analysis of Sanskrit Grammar in the perspective of Machine Translation and also provides one of the possible solution for Samaas Vigraha(Compound Dissolution).

Keywords: Machine Translation, Sanskrit, Natural Language Parser, Samaas Vigraha, Tokenization

1 Introduction

Sanskrit language and its grammar had exterted an emphatic impact on Computer Science and related research areas. It has resulted to put in extensive efforts in the field of Machine Translation(hereafter referred as MT). MT of Sanskrit is never an easy task, because of structural vastness of its Grammar. Besides, its strutural vastness Sanskrit Grammar is well organized and least ambigious compared to other natural languages, illustrated by the fact of increasing fascination for this ancient Aryan language. Its grammar possesses well organized rules and meta rules to infer those rules, thus proving to be a powerful analogy to context free grammar of a computer language.

Subsequently, it supports the idea of developing a parser for Sanskrit language, that would be helpful in developing a full-fledged MT system. As a part of development of parser, there are other important aspects to be taken care off. A *morphological analyser* and a *tokenizer* are two of the important components that play a vital role in the parser. A morphological analyser is used for identification of the base words from their morphonemes, further to understand the semantics of the original text. A tokenizer also plays its significant part in a parser, by identifying the group or collection of words, existing as a single and complex word in a sentence. Later on, it breaks up the complex word into its constituents in their appropriate forms. In Sanskrit, mainly we have two categories of complex words. They are

- Sandhi
- Samaas

1.1 Sandhi and Samaas

Sandhi: When two words combine to produce a new word whose point of combination is result of annihilation of case-end of former word and case-begin of latter. In short, the resulted new character that has been created at the point of combination is exactly equivalent to the sound produced when those two words are uttered without a pause. The inverse procedure to Sandhi-formation is known as *Sandhi Wicched*.

On the other hand, when two or more words are combined, based on their semantics then the resulting word is known as *Samaas* or Compound. Unlike

Sandhi, the point of combination in Samaas may or may not be a deformed in the resulting word. The inverse procedure of break-up of a Samaas is known as *Samaas Vigraha*. Considering the complexity of this problem, we restricted our focus to Samaas Vigraha or *Compound Dissolution*(hereafter Compound Dissolution is referred as CD for convenience).

1.2 Organization of the Paper

Initially, we would discuss about the problem of focus and the main objective of this paper in detail. Further, a little overview about the Sanskrit grammar and Knowledge Representation, that are required to understand the underlying concepts of the system. Then, we would brief about the existing systems in this areas and the related areas of interest. Later on, we would give a detailed description of the architecture of Vaakkriti. We would give a detailed analysis of the results of our system and finally, throw some light over our contribution to this research area.We shall conclude with some of drawbacks of our system and the challenges we have faced.

2 The Problem

Semantics being the prime focus, we need to learn the factors that effect the formation of a compound from the set of atomic words. The basic problem is identification of factors, by thorough analysis of language structure or with the help of a linguist. Especially various examples of Samaas must be extensively observed. After identification of factors, we need to find out the appropriate form of Knowledge Representation for the rule-base. Here, knowledge being the rules, based on which a particular compound is formed. The importance of CD can be clearly understood, during the process of tokenization. A well-defined set of rules in Sanskrit can be found in "Ashtadyayi", authored by 3^{rd} century grammarian and linguist Panini. Ashtadyayi contains rules of Grammar in a concise form, distributed over eight chapters. Our rule-base system would be based on the work of Kale et. al, that has detailed description of Paninian Grammar.

3 Sanskrit Grammar

As we have already mentioned that, it is necessary to know some of the basic concepts of the Sanskrit grammar. First, we would give some important definitions of terms that are frequently used in this paper.

3.1 Important Definitions

3.1.1 Vibhakti(Declension)

Sanskrit is a highly inflected language with three grammatical genders (masculine, feminine, neuter) and three numbers (singular, plural, dual). It has eight cases: nominative, vocative, accusative, instrumental, dative, ablative, genitive, and locative.

3.1.2 Dhatupata(Verbal Conjugation)

The verbs tenses (a very inexact application of the word, since more distinctions than simply tense are expressed) are organized into four 'systems' (as well as gerunds and infinitives, and such creatures as intensives or frequentatives, desideratives, causatives, and benedictives derived from more basic forms) based on the different stem forms (derived from verbal roots) used in conjugation. There are four tense systems:

- Present (Present, Imperfect, Imperative, Optative)
- Perfect
- Aorist
- Future (Future, Conditional)

3.2 Factors that effect

The list of factors that are involved in a rule are

- Part of Speech(hereafter referred as POS)
- List of Words(a token must be among a set of words to satisfy a rule)
- Case-End
- Case-Begin
- Declension
- Sense(a token with a particular sense is only qualified)
- Meaning
- Affix

- Affix Type(*Taddita* and *Kriti*)
- Number(sng, two, mny)(hereafter we refer number as num)
- Gender(mas, fem, neu)

The list of actions that act as functions in the consequent of a rule are:-

- setDecl(set the declension case for a specified token)
- addBefore(add a string before a specified token)
- addAfter(add a string after a specified token)
- setNumber(set the number of a token(sng,two,mny))
- replace(replace a token with a string related to it)

3.3 Compounds

Nominal compounds occur with various structures, however morphologically speaking they are essentially the same. Each noun (or adjective) is in its (weak) stem form, with only the final element receiving case inflection. Some examples of nominal compounds include:

Itaretara

Example: रामलक्ष्मणभरतशत्रुघ्नाः(RAmaLakshmaNaBaratAH) to रामः च लक्ष्मणः च भरतः च शत्रुघ्नः(RAma ca, LakshmaNa ca, Barata ca)

Rule: $\forall token$ POS($token$,noun) $\Rightarrow$ setDecl($token$,nom)$\wedge$ addAfter($token$,च)

Samahaara

Example: पाणीपादौ(pANIpAdau)to पाणी च पाददम् च(pANI ca pADam)

Rule: $\forall token, \exists sense$ POS($token$,noun) $\wedge$ SenseOf($token, sense$) $\Rightarrow$ setDecl($token$,nom)$\wedge$ addAfter($token$,च)

Dvitiya(Accusative) Tatpurusha

Example: दुःखातीतः(dukhatItaH)to दुःखम् अतीतः(dukham atItaH)

Rule: POS($token1$,noun) $\wedge$ WordList($token2$,श्रित, अतीत, पतित, गत, अत्यस्त, प्राप्त, आपन्न, गमी, बुभुक्षु) $\Rightarrow$setDecl($token1$,acc)

Trutiya(Instrumental) Tatpurusha

Example: दुःखातीतःto दुःखम् अतीतः

Rule: POS($token1$,noun) $\wedge$ (POS($token2$,verb) $\vee$ WordList($token2$,पूर्व, सदृश, ऊन))$\Rightarrow$ setDecl($token1$,ins)

Chaturthi(Dative) Tatpurusha

Example: यूपदारु(yupadaru)to यूपय दारु(yupaya daru)

Rule: POS($token1$,noun) $\wedge$ (Sense($token2$,"material") $\vee$ WordList($token2$,अर्थ, बलि, हित, सुख, रक्षित))$\Rightarrow$ setDecl($token1$,dat)

Panchami(Ablative) Tatpurusha

Example: चौरभयम्(cOrabayam)to चौराद् भयम्(cOraad bayam)

Rule: POS($token1$,noun) $\wedge$ (WordList($token2$, भय, भीत, भीति, भीः, अपेत, अपोढ, मुक्त, पतित, अपवस्त))$\Rightarrow$ setDecl($token1$,abl)

Shashti(Genitive) Tatpurusha

Example: राजपुरुषः(rAjapurushaH)to राजञ पुरुषः(rAjangya PurushaH)

Rule: POS($token1$,noun) $\wedge$ (POS($token2$,noun)$\wedge\neg$ POS($token2$,verb)$\wedge\neg$ NumeralType($token2$,ordinal)$\wedge\neg$ SenseOf($token2$,"quality"))$\Rightarrow$ setDecl($token1$,gen)

Saptami(Locative) Tatpurusha

Example: नगरकाकः(nagarAkAkaH)to नगरे काकः इव(nagare kAkaH iva)

Rule: POS($token1$,noun) $\wedge$ (MeaningOf($token2$,"crow")$\wedge$ SenseOf($token2$,"contempt"))$\Rightarrow$ setDecl($token1$,loc)$\wedge$ addAfter($token2$, इव)

4 Knowledge Representation

We have already learnt that the process of CD is supported by a rule-base system. A production system is a good illustration to understand a rule-base system. To represent a complex rule, it would be better to use First Order Predicate Logic(FOPL). Under FOPL a rule can be written as of the form $P(a) \wedge Q(a) \wedge Q(b) \wedge R(c) \Rightarrow Action_1(a) \wedge Action_2(b) \wedge Action_1(c)$

where P, Q and R are predicates

a, b and c are constant symbols

$Action$ is a function symbol

The rule-base system of Vaakkriti is developed considering the factors as predicates and the tokens as constant symbols. A sample rule would look like this

$POS(tok1, noun) \wedge (POS(tok2, verb)\ decl(tok2, acc)) \Rightarrow setDecl(token1, acc)$.

5 Related Work

In the recent times many efforts have been made to develop various utilities for Sanskrit. The tools developed includes Sanskrit to other Indian Language transliteration tools, simple and primitive translation tools, many grammar analysing tools and many more learning tools. Some of the important works includes Anusaraka, a primitive machine translation tool developed by Akshar et. al. Anusaraka tries to take advantage of the relative strengths of the computer and the human reader, where the computer takes the language load and leaves the world knowledge load on the reader. Besides, these tools, there are some beautiful theory-based research work was also done. The concept of Indian Network Language(INL) is one of such concepts that was proposed by Anupam et. al. It gives a hypothesis to consider Sanskrit as INL because of its important properties like free word order and inherent semantic net structure. There are few other interesting research concepts that have been analysed in the context of Sanskrit language. Rick Braggs et. al have shown in his article how Knowledge Representation in the language of Sanskrit is one of those wonderful concept to show that Semantic Nets. Semantic Nets are concept respresenting structures, that show how a concept is related to other concepts semantically, a semantic net would like in the figure below. Another beautiful research work was comparison of Paninian Grammar and Computer language Grammar. Bhate et al. has analysed to show that how well organized and structured is Sanskrit Grammar and its forgotten valuable contributions to the field of Computer Science.

6 Architecture

An Itrans standard formatted devanagiri text is given as input to the system and the output of the system is the set of tokens produced after CD. The list of components in the system are listed below:

- Input Processor
- Symbol Table
- Knowledge Base
- Inference Engine
- Database
- Rule-Base Editor

The architecture of Vaakkriti can be seen in the figure

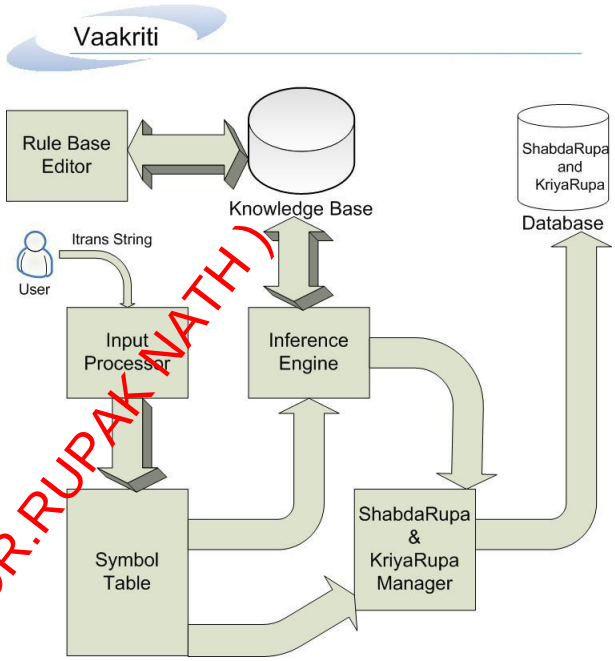

Figure 1: Architecture of Vaakriti

The algorithm of Vakkriti is given below:- A detailed description of each component is as follows.

Algorithm 1 Algorithm of Vaakkriti

```
input ← Itrans-Devanagiri Text
input′ ← breakUp(input)
tokenList ← tentativeTokenize(input′)
tokenInfoList ← tokenList
for token_i in tokenInfoList do
    token_(i) ← extractInfo(token_i
    update token_(i) in tokenInfoList
end for
for each rule(r) in Knowledge-Base(KB) do
    result ← infer(r,tokenInfoList)
    if result is true then
        return r
    end if
end for
```

6.1 Input Processor

The unstemmed compound taken as input to the system is a string in itrans format. First, Input Processor breaks the itrans string into chunks of characters on the basis of Devanagiri Character set. The heuristic for break up procedure is given below:-

The reason behind the breakup procedure is to ease the process of breaking the string into words in their tentative forms. If a string is considered as it is without breakup into devanagiri characters, then there is a high chance of ambiguity while lookup in the dictionary. For example:-

Without breakup of input string

```
aja
ajagaraH-- Found this word
```

With breakup of string into character sequences

```
a,ja
a,ja,ga,raH
```

Later on the chunks of characters are processed as in the procedure below:-

The words lying in input string are tentatively guessed by maintaining a stack of character sequences, thus checking with the dictionary for the right word. But, in most of the cases, the word in the input string do not have an exact match in the dictionary. This is because of the *matra* appended to *Case-End* of a word. Therefore, we have generated tokens for each matra and tried to find it in the dictionary. If the word is found, then the word along with its meaning is stored in the Symbol Table.

6.2 Symbol Table

Now, we shall discuss more about how a Symbol Table fetches those subtle information of a token. Symbol table extracts token information in the following manner:-

6.2.1 Part of Speech

Part of Speech is identified with the help of standard Monier Williams Dictionary, List of Adverbs, List of Prepositions, List of Numerals.

6.2.2 Sense and Meaning

First, meaning of the token is known from the dictionary and the sense of the token is fetched through a special kind of procedure. The technique has following steps:-

1. Identify the nouns in the meaning phrase.
2. Find sense for each noun with the help of English Wordnet.
3. Find a list of "common" senses for all the nouns.
4. That list of senses is assumed to the sense of a token.

6.2.3 Gender and Number

These are fetched from the XML database.

6.3 Knowledge Base

The Knowledge Base(KB) contains facts and rules that supports the system, for identifying a given input. The KB has been classified well, according to the Rule Sets. A Rule Set is a set of rules that are meant for a particular type of compound. Infact, a new rule set can be created whenever there is a new part of speech to be dealt with. It has been assumed that, a rule has clauses(both unit and definite) on antecedent side, whose number is equal to tentative number of tokens in the input parsed string. On the other hand, the consequent or conclusion contains the list of actions that has to be operated over the tokens(in the input string) by the system. More about the rule structure in the next section.

The KB is well integrated with the Rule Base Editor(RBE) and the Inference Engine. Currently, it contains limited number of rules this makes the KB non-monotonic, yet it can be made monotonic, by addition of new rules.

6.4 Database

There is a large database that supports the whole system of Vaakriti. The database is contained in the form of XML files. There are following tables in the database:-

- Nouns, Adjectives, Numerals Declensions.
- Adverbs, Conjunctions and Prepositions.
- Dictionary Database.
- Preverbs database.
- Other Morphonemes.

6.5 Inference Engine

Whenever premises of a particular are satisified by the input parse string, then it is said that a rule is fired. A fired rule applies its consequent part over the parsed string to result in actual goal. This procedure is known as Rule Inference.

6.6 Rule Base Editor

The sole motive of Rule-Base Editor is to free the Knowledge Engineer free from rule entry. A Linguist with little training to operate the GUI can be provided, would suffice this task.

7 Results

The system has been tested with many examples that have been taken from the book written by Kale et al. The set of examples have been chosen from different set of Compounds. In most of the cases system has given correct results with a precision of 90%, but in some of the cases that involve sense, it became quite difficult to produce the result. Lack of linguistic tools like Wordnet for Sanskrit language imposes limitations on word sense disambiguation. We have developed a sense list for a limited set of words by observing some of the important sanskrit texts, based on the knowledge we have acquired.

8 Our Contribution

We have proposed a utility called Rule-Base Editor, besides our actual work on CD. The motive behind Rule-Base Editor is to induce the property of flexibility into the system. It always avails a linguist to enter new rules with the help of Rule-Base Editor without any support from knowledge engineer.

We have already learnt that Samaas Vigraha(CD) is the most important aspect of the tokenization phase in a parser. Implicitly, the acquisition of factors and rules also gather equal importance. Signifying this fact, we have done rigorous survey over the grammar to identify these factors. Hence, we assert that our system will be a significant contribution in this area of research.

9 Future Scope and Conclusion

We assert that Vaakkriti would be a preliminary contribution to the realm of NLP. Adding to the major works that have been done already, Vaakkriti is an attempt to enhance the existing works. We would extend the current system and develop a full-fledged parser that will suffice most of the requirements of MTsystem.

Although, it looks the way that the problem has been solved, but the actual problems arouses when a Sanskrit poem is given as input to a MT system. Usually, a sanskrit poem conveys more than one meaning and sometimes figure of speech is used, that adds fuel to the fire. This becomes a herculean task for a MT system and it will remain as a myth forever.

Acknowledgements

The authors would like to specially thank Gerard Huet for providing linguistic database of declensions and verbal roots, that was quite helpful in making our system fine and complete. The authors gratefully acknowledge financial support from the Universal Digital Library project, funded by the Ministry of Communication and Information Technology (MCIT) India and also Indian Institute of Information Technology, Allahabad.

References

Higher Sanskrit Grammar, M. R. Kale, Motilal Banarasi-Dass Publishers.

"Paninis Grammar and Computer Science", Saroja Bhate and Subhash Kak, Annals of the Bhandarkar Oriental Research Institute, vol. 72, 1993, pp. 79-94.

"Knowledge Representation in Sanskrit and Artificial Intelligence", Rick Briggs

"Artificial Intelligence", Elain Rich and Kevin Knight, 2nd Edition, Tata McGrawHill, 1991.

"Artificial Intelligence, A Modern Approach" Stuart Russell and Peter Norvig, 2nd Edition, Pearson Education, 2003.

"Sanskrit as Indian Networking Language: A Sanskrit Parser", Anupam, 2004.

"Natural Language Processing, A Paninian Perspective", Akshar Bharti, Vineet Chaitanya and Rajeev Sangal, Pearson Education.

"Natural Language Processing using PROLOG" by M. Gerald, M Chris, Addison and Wisley, 1989.

"Cognitive Science Learning Resource", http://www.comp.leeds.ac.uk/ugadmit/cogsci/knowled