Design and Architecture of 'Anusaaraka'- An Approach to Machine Translation

By Amba P. Kulkarni

Abstract

Most research in Machine translation is about having the computers completely bear the load of translating one human language into another. This paper looks at the machine translation problem afresh and observes that there is a need to share the load between man and machine, distinguish 'reliable' knowledge from the 'heuristics', provide a spectrum of outputs to serve different strata of people, and finally make use of existing resources instead of reinventing the wheel.

This paper describes the architecture and design of 'Anusaaraka' based on the fundamental premise of sharing the load, resulting in "good enough" results according to the needs of the reader. The architecture differs from the conventional in three major ways:

- Reversal in the order of operations as compared to conventional machine translation systems
- Introduction of interfaces that act like glue and improve the modularity of the system
- Development of a GUI to provide the 'right ' amount of information at the right time

The paper attempts to prove that this new architecture is a better approach to Machine Translation, since a) it makes the machine translation process transparent to the user-cum-developer; and b) it leads to machine translation in stages, thus ensuring robustness.

Keywords: Machine Translation, Word Sense Disambiguation, Anusaaraka, Padasutra, Polysemous

Introduction

Much of the information that is available widely on the World Wide Web and through other media is predominantly in the English language. As a result, regional language users across the world who need this information are unable to use it because of a language constraint. However, the advent of machine translation has proved beneficial, in doing away with these language barriers and helping regional language users to access required data in the language of their choice.

The question still asked is "to what extent can machine translation help in overcoming the language barrier?" The history of machine translation is as old as the history of computers. However, even after 50 years, except a few, there is no evidence of systems that can provide good translation.

The Anusaaraka architecture has been designed and developed based on issues revealed during an evaluation of conventional machine translation and the apparent shortfall arising out of a lack of fallback mechanism. The issues revealed:

- A need to share the load between man and machine
- Distinguish 'reliable' knowledge from the 'heuristics'

- Provide a spectrum of outputs to serve different strata of people
- Make use of existing resources instead of reinventing the wheel

The evaluation study further revealed an imperative need for the total restructuring of the conventional machine translation systems. This would in case of failures, provide a safety net to the users and let the the machine translation system degrade.

The sections ahead highlight the major difficulties faced in machine translation, observations that lead to the new architecture for a machine translation, the detailed architecture of core anusaaraka, the need for interfaces to facilitate plugging in of different linguistic tools (in particular-parsers) the user interface and finally converging to conclude by supporting the research claim.

Problems in Machine Translation

The following points highlight the difficulties in machine translation.

Languages code information partially

Languages do not completely code information. The text is open to different interpretations, based on who is reading it. In that manner, text in any language is like a painting. The viewers interpret the painting in their own way. In the process of interpretation, they bring in their unique combination of common sense, world knowledge, language conventions, cultural background, domain specific knowledge, and current state of mind.

There is a trade-off between precision and brevity. Brevity introduces ambiguity, whereas precision needs to loss of focus. Natural languages have a natural tendency to lean towards brevity. Hence, they have inherent ambiguity.

Information is coded at arbitrarily long place

To resolve text ambiguity, it is not clear how much of text is to be processed. Sometimes text as much as a complete novel may have to be processed.

Differences across languages

- **Differences in coding strategies:** Two languages may code the same information in different ways.

  E.g. Consider the English sentence, 'Rats kill cats.'

  (1) Its Hindi translation is 'cUhe billiyoM ko mArate hEM'

  (2) In English 'vibhakti' information is coded in position. Hindi on the other hand requires an explicit 'vibhakti' marker (ko).

- **Differences in Concept-Word mappings:** Concepts are infinite, whereas words in any language are innumerable. The way conceptual space is divided into small regions differs from language to language, i.e. each region corresponding to a word in one language may differ in another language. This implies there need not be one-to-one correspondence between the meanings of words of one language to those of the other.

All these factors lead to 'incommensurability' between two languages.

Subjectivity in translation

Translation involves understanding the original text and presenting it in other language. The presentation part involves creativity. More than one translations of the text may be undertaken to cater to the needs and likes of different people. Thus, for a given text there is no 'one specific way' of translation. What exists is a spectrum of translations and it is the translator who translates' what they feel is most appropriate.

All these issues make the task of machine translation difficult.

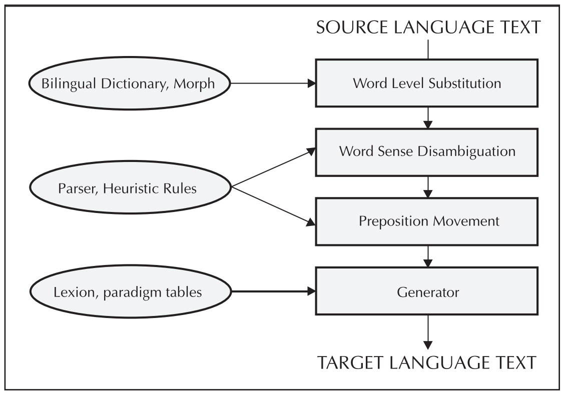


Figure 1 –The architecture of "core" anusaaraka

A fresh look at the problem

Machine Translation is difficult. Ideal machine translation is still a dream, requiring many stages, huge computing power and innovations.

Ultimately, the machine translation output is to be read by humans, and interpreted in their way. So why not involve the human earlier into the machine translation process? Machines are equipped with large memory storage, they can "remember" large quantities of information. Humans are good at interpretation. So the natural suggestion is to share the load between man and machine.

The question is how to share the load between man and machine?

Some of the components like morphological analyzers, dictionaries in principle can produce "accurate" information, whereas some other components like POS taggers, parsers in principle are prone to errors, because they do not have access to the world knowledge.

The question is how to split the system into different modules so that user knows which part of the output is reliable and which is not?

During the course of translation there exists a tension between faithfulness to the original text and naturalness in the target language making accurate translation a difficult task. Machine translation systems favor naturalness over the faithfulness. These systems, therefore, serve only a certain strata of people. But, there are others who would like to have an access to the "original" text (quality of faithfulness) without any "distortions" as introduced by the translation process.

The question is, can the diverse needs of these different strata of people be addressed?

Finally, a machine translation system requires many linguistic resources. Most of them are available for English for free download on the Internet under General Public License (GPL)[7] for. It is natural to make use of these resources rather than reinventing the wheel. In fact there are more than one parsers, POS taggers and morphological analyzers for English available under GPL.

How should the system be designed so that one can plug-in different language tools such as POS taggers, parsers, morphological analyzers, etc.?

The Anusaaraka system attempts to answer these questions.

Architecture of the Anusaaraka system

The Anusaaraka system has two major components.

- Core engine
- User–cum-developer interface

'Core' engine is the main engine of anusaaraka. This engine produces the output in different layers making the process of Machine Translation transparent to the user.

The architecture of "core" anusaaraka is shown in Figure 1.

This architecture differs from the conventional architecture in three major ways

1. The order of operations is reversed. In the new architecture there is initial word level substitution followed by use of other language resources that are less reliable, like POS taggers, parsers, etc.

2. A graphical user interface has been developed to display the spectrum of outputs. The user has flexibility to adjust the output as per his/her needs. There will be users of different kinds based on the level of sophistication required and skill in handling the tool.
3. Special "interfaces", which act as 'glue' have been developed for different parsers, which allow plugging in of different parsers thereby providing modularity.

Core Anusaaraka engine

The core anusaaraka engine has four major modules viz.

I. Word Level Substitution

II. Word Sense Disambiguation

III. Preposition placement

IV. Hindi Word Order generation

Each of the above four modules is described in detail. A justification of how these changes answer the questions raised in section 3 is presented.

Word Level Substitution

At this level the 'gloss' of each source language word into the target language is provided. However, the Polysemous words (words having more than one related meaning) create problems. When there is no one-one mapping, it is not practical to list all the meanings. On the other hand, anusaaraka claims 'faithfulness' to the original text. Then how is the faithfulness guaranteed at word level substitution?

Concept of Padasutra

To arrive at the solution, the user must understand why a native speaker does not find it odd to have so many 'seemingly' different meanings of a word. By looking at the various usages of any Polysemous word, users may observe that these Polysemous words have a "core meaning" and other meanings are natural extensions of this meaning. In anusaaraka an attempt is made to relate all these meanings and show their relationship by means of a formula. This formula is termed *Padasutra*[2]. (The concept of Padasutra is based on the concept of 'pravrutti-nimitta' from traditional grammar) The word padasutra itself has two meanings:

- a thread connecting different senses
- a formula for pada

An example of Padasutra:

The English word 'leave' as a noun means 'Cutti' in Hindi, and as a verb its Hindi meaning is 'CodanA' and it is obvious that 'CodanA' is derived from 'Cutti'. Hence, the Padasutra for 'leave' is

leave: Cutti[>CodanA]

Here 'a>b' stands for 'b gets derived from a' and 'a[b]' roughly stands for 'a or b'.

Thus, by division of workload and adoption of the concept of 'Padasutra—word formula', the research guarantees that the first level output is 'faithful' to the original and also acts as a 'safety net' where other modules fail.

At this level some of the English words like functional words, articles, etc. are not substituted. The reason being they are either highly ambiguous, or there is a lexical/conceptual gap in Hindi corresponding to the English words (e.g. articles), or substituting them may lead to catastrophe.

Thus, for the input sentence 'rats killed cats' the output after word level substitution is

cUhA{s} mArA{ed/en} billI{s}

Training Component

To understand the output produced in this manner, a human being needs some training. The training presents English grammar through the Paaninian view[3].

Thus, if a user is willing to put in some effort, he/ she has complete access to the original text.

The effort required here is that of making correct choices based on the common sense, world knowledge, etc.

The training component layer ensures that the layer produces an output, which is a "rough" translation that systematically differs from Hindi. Since the output is generated following certain principles, the chances of getting mislead are less. Theoretically, the output at this layer is reversible.

Word Sense Disambiguation (WSD)

English has a very rich source of systematic ambiguity. Majority of nouns in English can potentially be used as verbs. Therefore, the WSD task in case of English can be split into two classes:

(i) WSD across POS

(ii) WSD within POS

The POS taggers can help in WSD when the ambiguity is across POSs.

For example: Consider the two sentences

'He chairs the session'.

'The chairs in this room are comfortable'.

The POS taggers mark the words with appropriate POS tags. These taggers use certain heuristic rules, and hence may sometimes go wrong. The reported performances of these POS taggers vary between 95% to 97%. However, they are still useful, since they reduce the search space for meanings substantially.

However, disambiguation in the case of Polysemous words requires disambiguation rules. It is not an easy task to frame such rules.

It is the context, which plays a crucial role in disambiguation. The context may be

- the words in proximity, or
- other words in the sentence that are related to the word to be disambiguated.

The question is how can such rules be made efficiently? To frame disambiguation rules manually would require thousands of man-years. Is it possible to use machines to automate this process?

The wasp workbench [8] is the best example of how, with the help of a small seed data, machines can learn from the corpus and produce disambiguation rules.

Anusaaraka uses the wasp workbench to semi-automatically generate these disambiguation rules. The output produced at this stage is irreversible, since machine makes choices based on heuristics.

Preposition Placement

English has prepositions whereas Hindi has postpositions.

- Hence, it is necessary to move the prepositions to proper positions in Hindi before substituting their meanings. While moving the prepositions from their English positions to the proper Hindi positions, \record of their movements must be stored, so that in case a need arises, they can be reverted back to their original position. Therefore, the transformations performed by this module, are also reversible.

Hindi Word Order Generation

Hindi is a free word order language. Therefore, even the anusaaraka output in the previous layer makes sense to the Hindi reader. However, this output not being natural in Hindi, may not be enjoyed as much as the output with natural Hindi order. Additionally, it would not be treated as a translation. Therefore, in this module the attempt is to generate the correct Hindi word order.

Interface for different linguistic tools

The second major contribution of this architecture is the concept of 'interfaces'. Machine translation requires language resources such as POS taggers, morphological analyzers, and parsers. More than

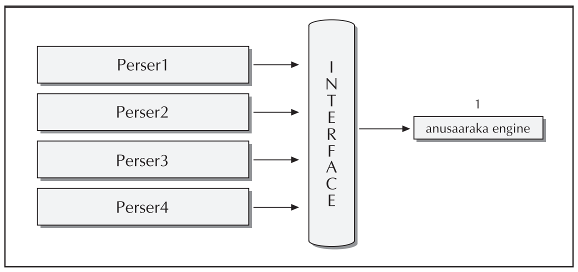


Figure 2 –Interfaces that map output of parsers to an intermediate form

one kinds of each of these tools exist. Hence, it is wise to use these tools. However, there are problems.

For examples – Parsers

I. These parsers do not have satisfactory performance. 40% of the time, the first parse is the correct parse. Parse of a sentence tells how the words are related to each other. 90% of such relations in any parse are typically correct.

II. Each of these parsers is based on different grammatical formalism. Hence, the output they produce is also influenced by the theoretical considerations of this grammar formalism.

III. Since the output format for different parsers is different, it is not possible to remove one parser and plug in the other one.

IV. One needs trained manpower to interpret the output produced by these parsers and to improve the performance of these parsers.

As a machine translation system developer who is interested in the "usable" product one would like to plug-in different parsers and watch the performance. May be one would like to use combinations of them, or may like to vote among different parsers and choose the best parse out of them.

The question then is how to achieve it?

It is not enough to have the modular programs. The parser itself is an independent module. What is required is plug-in facility for different parsers. This is possible, provided all the parsers produce an output in some common format. Hence, interfaces are necessary to map output of parsers to an intermediate form as illustrated in figure 2.

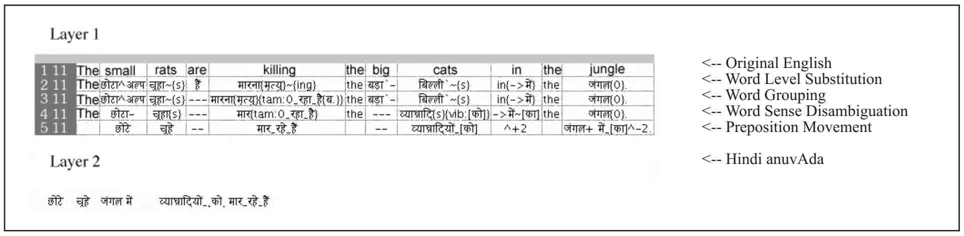
Anusaaraka output and the user interface

The Java based user interface has been developed to display the outputs produced by different layers of anusaaraka engine. The user interface provides a flexibility to control the display.

A snapshot of a sample English-Hindi anusaaraka output with brief explanation of each of the layers is provided :

Layer1

- Row 1: Original English sentence
- Row 2: Word level substitution
  - ❑ Least fragile layer.
  - ❑ Contains Hindi Padasutra (word formula) for each English word.
    For example small -> *CotA^alpa*
    rats -> *cUhA{s}*
- Row 3: Word Grouping
  - ❑ The group of words which as a group give some new meaning (e.g. compounding) are grouped together.
    In the above sentence,
    are + ing = *0_rahA_hE*



Layer 1

1 11	The	small	rats	are	killing	the	big	cats	in	the	jungle	<-- Original English
2 11	The	छोटा^अल्प	चूहा~{s}	हैं	मारना{मृत्यु}~{ing}	the	बड़ा`-	बिल्ली`~{s}	in{->में}	the	जंगल{0}.	<-- Word Level Substitution
3 11	The	छोटा^अल्प	चूहा~{s}	---	मारना{मृत्यु}{tam:0_रहा_है{ब.}}	the	बड़ा`-	बिल्ली`~{s}	in{->में}	the	जंगल{0}.	<-- Word Grouping
4 11	The	छोटा-	चूहा{s}	---	मार{tam:0_रहा_है}	the	---	व्याघ्रादि{s}{vib:[को]}	->में~[का]	the	जंगल{0}.	<-- Word Sense Disambiguation
5 11		छोटे	चूहे	--	मार_रहे_हैं		--	व्याघ्रादियों_[को]	^+2		जंगल+ में_[का]^-2.	<-- Preposition Movement

Layer 2   <-- Hindi anuvAda

छोटे चूहे जंगल में व्याघ्रादियों_.को. मार_रहे_हैं

Figure 3 – Snapshot of a sample Engilsh-Hindi Anusaaraka output

- Row 4: Word Sense Disambiguation
  - ❑ Attempts to select the appropriate sense according to the context.
    For example, the big cats -> *vyaaghraadii*
- Row 5: Preposition Movement
  - ❑ The prepositions are moved to their correct Hindi positions.
    E.g. '*->meM — jangala*' is changed to '*— — jangala+meM*'.

Layer 2: Hindi anuvAda

Proper Hindi sentence is generated

Anusaaraka: A better approach for Machine Translation

The research conducted thus far claims that anusaaraka is a better approach for Machine Translation[3] because it is

Robust

It always produces the output. If the machine fails at higher levels, which are in principle fragile, lower level outputs are still available to the user. However to understand the lower level outputs, some training is required.

Transparent

The output at different levels makes the whole process of machine Translation transparent to the user. This opens up a new opportunity for the persons having an aptitude for language analysis to contribute the Machine Translation efforts even without any formal training in computational linguistics, or NLP.

Future Work

The framework for anusaaraka with initial data set (appendix-1) for English-Hindi pair is almost ready for *user-cum-developers*.

The initial data, is deemed sufficient to start a "boot-strapping" process. However, the data needs to be enhanced qualitatively and quantitatively several times for general use.

What is required is

- Continuous feedback from the user-cum-developers, and
- Enrichment and enlargement of the data

This opens up an opportunity for the language lovers to participate in the development of machine translation systems, thereby also making them participants in the IT revolution rather than mere IT consumers.

Acknowledgment

The major contribution in the conceptualization of the design and architecture of this system taking insights from Paninian Grammar is by Dr. Vineet Chaitanya. The author has given a concrete shape to these concepts and ideas by actual implementation. Dr. Dipti Misra has given the linguistic inputs and the user interface has been developed by Ms. Surekha and Mr. Srinivas Raghavan. Apart from these, there are many other volunteers who have contributed to this project.

References

[1] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, "Natural Language Processing", Prentice Hall of India, 1995.

[2] Akshar Bharati, Vineet Chaitanya, Dipti Misra, Amba Kulkarni. Modern Technologies for language Access: An aid to read English in the Indian Context: Osmania Papers in Linguistics, Ed. V. Swarajya Lakshmi, pp.111-126.

[3] anusaaraka
URL = <http://nlp.iiit.net/ ~anusaaraka>

[4] Bharati, Akshar, Rajeev Sangal, Dipti M Sharma, Amba P Kulkarni: Machine translation activities in India: A survey, In proceedings of workshop on survey on Research and Development of Machine translation in Asian Countries, Thailand, May 13-14, 2002.

[5] GNU General Public Licence:
URL=<http://www.gnu.org/copyleft/gpl.txt>

[6] Kilgarriff Adam, Tugwell David: WASP-Bench an machine translation Lexicographer's Workstation Supporting State-of-the-art Lexical Disambiguation, To be presented at machine translation Summit VII, Santiagode Compostela, September 2001.

Appendix-I

The current English-Hindi anusaaraka has

- An English-Hindi bilingual dictionary with approximately 28,000 headwords, compounds, phrases and idioms.
- A dictionary of Padasutras for around 2000 English words, with an explanation on the connections between different senses for around 400 of them.

Word sense disambiguation rules for around 1000 words. Majority of these rules are generated 'semi-automatically' using the WASP[6] workbench.