Anusaaraka:A better approach to Machine Translation
{ A case study for English-Hindi/Telugu}

Akshar Bharati

Amba P Kulkarni

Dipti Misra Sharma

Technologies Research Center,

International Institute of Information Technology,

Hyderabad

(amba@iiit.net,dipti@iiit.net)

1. Introduction:

Conventional Machine Translation systems are necessarily fragile. The main reason behind this is the 'incommensurability' among natural languages. The incommensurability arises because languages encode information partially and follow different coding schemes.

Since languages in general do not code information completely, we can say that the text in a language is like a 'picture' with some explicit 'key strokes'. It is the reader who fills in the gaps by supplying the missing information and interprets it. The missing information may involve various combinations of common sense, the world knowledge, language conventions, cultural background, and domain specific knowledge.

Here is an example illustrating this[1].

Hindi: phala raama khaataa hai

gloss: fruit(m.) Ram(m.) eats(m.)

In the above sentence *kartaa* (or the agent) as well as *karma* (or the patient) is not marked explicitly.

However, a native Hindi speaker, does not have any difficulty in 'understanding' the

[1] Here we give Hindi examples. However, the complete discussion also holds good for other Indian languages, and in particular, Telugu.

sentence. Based on the '*yogyataa*' , he interprets it correctly that a person (Ram, in this sentence) would be the agent of eating, and fruit is the thing which is eaten.

When something against '*yogyataa*' is to be stated, then language marks it through the vibhakti explicitly as in the following example.

Hindi: Lataa sharaaba nahiiM piitii, sharaaba lataako piitii hai

gloss: Lataa alcohol not drink, alcohol Lataa (accus.) drinks.

Thus the information coding is not simple. The process of interpretation, in general, involves a complex interaction between various sources of information. Quite frequently there is also ellipses in sentences. This makes the interpretation more difficult.

1.1 Problems in translation:

The complexity increases when it comes to translating it to another language. Whenever the source language does not code some information explicitly, but the target language requires it to be coded explicitly, then one is forced to 'guess' the missing information.

This 'guessing' makes Machine Translation systems 'fragile'.

2. **Solution**:

There should be a fall-back mechanism to recover from the failures.

It should be noted that the reader has complete access to the world knowledge, common sense, etc. that are necessary to interpret the text. Hence the system should be designed keeping the reader at the central place.

One of the solutions adapted by the MT developers is to provide the 'rough' translation in cases of failures. However, the 'roughness' is not well defined, and hence this solution does not serve the purpose.

The Basic design features for such a system may be stated as:

-- make sure that complete information is available to the user at every stage.

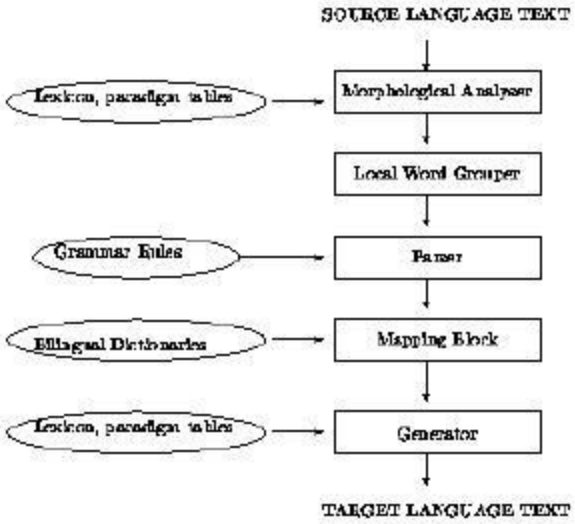
-- present complete information to the reader, without cluttering the scene.

-- separate the resources that can be made, in principle, reliable from those that are, inherently unreliable.

-- enable human being to take charge of the situation, wherever important.

The question that we face now is how to go about doing this?

The conventional MT systems (fig 1) do the dictionary lookup only after complete analysis of the source language text. Different linguistic tools used for the source language analysis, however, are not 'perfect'. For example, the POS taggers, have only 95% to 97% accuracy. In other words, on an average 3 to 5 words out of every 100 are marked with incorrect tag. Once the decision is made at POS level, the other possibilities are filtered out. Therefore, after the dictionary look-up what reader gets is the filtered sense. Since the filtering is done by machine, user does not have any 'control' over the filtering process.



(fig 1)

Secondly, even if the developer would like to present all the possibilities to the reader, it is difficult to present the complete information from analysis in compact form to the user.

Hence, the practical implication of the above guidelines is

-- The order of operations should be reversed, and

-- a Graphical User Interface(GUI) is to be developed to present the 'right' amount of information to the user at 'right' time.

We describe below the architecture of anusaaraka engine (fig 2) which is based on the above guidelines.

3. **Core anusaaraka engine**:

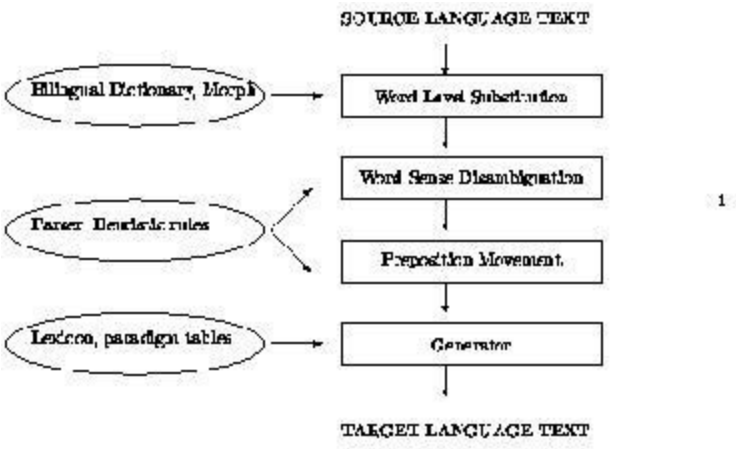
The core anusaaraka engine has four major modules viz.

(i) Word Level Substitution

(ii) Word Sense Disambiguation

(iii) Preposition placement, and

(iv) Target Language preferred Word Order generation



(fig 2)

3.1. **Word Level Substitution**

At this level we provide the gloss for each source language word in the target language. The polysemous words, however, are a major source of problems. When there is no one-one mapping, it is not practical to list all the meanings. Recall that, on the other hand, anusaaraka aims at providing complete information to the reader. The question is now, how can this be guaranteed at word level substitution?

3.1.1 **Concept of *padasutra***

To seek the solution, first let us see why a native speaker does not find it odd to have so many seemingly different meanings of a word. If we look at the various usages of any polysemous word, we observe that these polysemous words often have a core meaning and other meanings are natural extensions of this meaning. In anusaaraka we try to relate all these meanings and show their relationship by means of a formula. We call this formula a *padasutra*[2]. (The concept of padasutra is based on the concept of pravrutti-nimitta from Indian traditional grammars.)

The word padasutra itself has two meanings:

-- a thread connecting different senses , and

-- a formula for pada.

Here is an example of padasutra:

The English word 'leave' as a noun means chutti' and as a verb 'chodanaa' in Hindi and it is obvious that chodanaa and chutti are related.

Hence the Padasutra for leave is

leave: chutti[>chodanaa]

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

3.1.2 Training Component:

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

Thus for the input sentence like

rats killed cats

the output after word level substitution is

cuuhaa{s} maaraa{ed/en} billI{s}

To understand the output thus produced, a human being needs some training. Thus if a user is willing to put in some effort, he 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.

This layer ensures to produce 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.

Thus by division of workload between man and machine and adoption of the concept of padasutra(word formula), we guarantee that the first level output is faithful to the original and also acts as a safety net when later modules fail.

3.2 Word Sense Disambiguation (WSD):

The next module in anusaaraka is the Word Sense Disambiguation.

The WSD task, in general, 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.

e.g. Consider the following two sentences

He chairs the session.

The chairs in this room are comfortable.

In the first sentence, 'chairs' is used as a verb, and in the second sentence 'chairs' is used as a noun. Therefore to decide the meaning of 'chairs' in the respective contexts, one can just rely on POS tags.

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 useful, since they reduce the search space for meanings substantially.

The disambiguation in the case of polysemous words however requires disambiguation rules. It is not an easy task to frame such rules. It is the context that 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 we make such rules efficiently? To frame disambiguation rules manually would require thousands of man-hours. Can we use machines to automate this process? The wasp workbench[3] is a very useful tool for developing such rules semi-automatically.

Anusaaraka uses this tool for developing word sense disambiguation rules semi-automatically.

The output produced at this stage is irreversible, since machine makes choices based on

heuristics.

3.3 Preposition Placement:

In the context of English-Hindi (as an example case, which is applicable to English-Telugu and other Indian languages as well) machine translation, the prepositions of English needs to be replaced by the corresponding postpositions in Hindi. It is also necessary to move the prepositions to proper positions in Hindi before being substituted by their equivalent meanings. To move the prepositions, it is necessary to mark the NP boundary. Since, the parsers often make mistakes, there are chances of failure. Hence, while moving the prepositions from their before-np positions to the after-np positions in target language, in anusaaraka a record of their movements is kept so that in case need arises, user can revert them back to their original position. The transformations performed by this module, therefore, are reversible.

3.4. Target Language preferred Word Order Generation:

Since Indian languages allow certain amount of freedom in the order of words, the anusaaraka output at previous layer makes sense to the Indian language reader. However this output not being natural to the Indian languages, one may not enjoy it as much as one may with natural order. Also it would not be treated as a translation. Therefore in this module our attempt is to generate the correct word order of the target language.

4. Anusaaraka: A better development strategy for building Machine Translation systems

Anusaaraka acts as a microscope for a MT system developer.

Since anusaaraka presents the complete information that is available in the source language text, using target language vocabulary, one may list the following advantages:

-- It points out the ways that different languages encode the information and the amount of information coded.

-- It helps in identifying the problems of incommensurability among the languages.

-- It helps in identifying sources of information and thereby helps to figure out what is in principle possible and what is not, in a scientific way.

-- It brings into focus those phenomena that are of prime importance from translation point of view.

5. Implementing English-Telugu anusaaraka:

The linguistic resources required for developing English-Telugu anusaaraka are:

1. Padasutras for ambiguous words (can be prioritized in terms of highly ambiguous and most frequent ones to begin with),
2. English-Telugu bilingual dictionary for content words as well as function words,
3. Word Sense disambiguation rules,
4. Preposition-vibhakti mapping rules, and
5. Telugu generator

Since Indian languages share certain common things with respect to vocabulary, it may be possible that a number of WSD rules developed for Hindi will also work for Telugu. In that case the English words just need to be substituted by appropriate Telugu words.

6. Enhancing Telugu-Hindi anusaaraka towards Machine Translation:

The Akshar Bharati group has developed anusaarakas among Indian languages. The current Telugu-Hindi anusaaraka has only one layer viz. the word level substitution layer. It needs to be improved further to make it more user friendly. By developing other layers on top of the existing ones, one can enhance the anusaarakas among Indian languages towards Machine Translation.

The following layers on top of the current layer need to be built.

1. Word Sense Disambiguation, and
2. Hindi generator

Since Telugu and Hindi share common vocabulary to a large extent, the volume of work needed for Telugu-Hindi WSD is far less as compared to English-Hindi or English-Telugu WSD. However, there is a necessity to develop a WSD workbench that can help a person to develop the rules semi-automatically.

7. Conclusion:

The new architecture of anusaaraka together with a user-friendly interface is convenient for a user-cum-developer. This new architecture makes a clear-cut distinction between the resources that are in principle reliable and those that are in principle probabilistic. It acts as a microscope for the Machine Translation developers.

8.Acknowledgment:

The English-Hindi anusaaraka is being supported by Satyam Computers Services Limited, and they have kindly permitted to keep anusaaraka available under General Public License. The major contribution in the conceptualisation of the design and architecture of this system taking insights from Paninian Grammar is by Dr. Vineet Chaitanya. Authors have given a concrete shape to these concepts and ideas by actual implementation. Thanks are due to Dr. G. Uma Maheshwar Rao from Center for Applied Linguistics and Translation Studies, at University of Hyderabad, for providing the initial set of English-Telugu data required for starting work on English-Telugu anusaaraka.

References:

[1] 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.

[2] anusaaraka URL: http://nlp.iiit.net/~anusaaraka

[3] Kilgarriff Adam, Tugwell David: WASP-Bench: 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	small	rats	are	killing
^t	చిన్న/తక్కువ	ఎలుక~{s}/స్వామి_ద్రోహము_చెయ్యి~{s}	ఉన్నారు[*]	చంపే[-]/చంపడం/సంహారం[-]/చంపు~{i
^t	చిన్న/తక్కువ	ఎలుక~{s}/స్వామి_ద్రోహము_చెయ్యి~{s}	---	చంపు{tam:wunnA}
	చిన్న	ఎలుకలు	--	చంపుతున్నాయి
	చిన్న	ఎలుకలు	--	చంపుతున్నాయి

big	cats	in	the	jungle.
/పరిపరిగా/విస్తారమైన/పెద్దడైన	పిల్లి~{s}	in{->లో/లోపల}	^t	అడవి{0}.
/పరిపరిగా/విస్తారమైన/పెద్దడైన	పిల్లి~{s}	in{->లో/లోపల}	^t	అడవి{0}.
--	వ్యాఘ్రాలని	->లో		అడవి.
--	వ్యాఘ్రాలని	^#+2		అడవి+లో^#-2

1|2.1	The	small	rats	are	killing	the	big	cats	in	the	jungle
2|2.1	చిన్న	ఎలుకలు	అడవి	లో	వ్యాఘ్రాలని	చంపుతున్నాయి					

Sample English-Telugu anusaaraka output Layer1

Row 1: Original English sentence

Row 2: Word level substitution

Least fragile layer.

Contains Telugu padasutra(word formula) for each English words.

E.g. Small -> cinna^takkuva

rats -> eluku/svaamii~drohamu~ceyyi

Row 3: Word Grouping

A group of words with a new meaning(e.g. Compounding) For example in the above sentence, are + ing = tunnaa

Row 4: Word Sense Disambiguation Attempts to select the appropriate sense

according to the context.

For example, the big cats -> vyaaghramu

Row 5: Preposition Movement

The prepositions are moved to their correct Telugu positions E.g. '->lona-- adivi' is changed to '--- --- adivi+lona'.

Layer2: Telugu anuvaada

Telugu sentence with preferred word order is generated.