5.10 CoIL-Net Centre : Machine Aided Translation from Hindi to English
Indian Institute of Technology, Kanpur

The Indian Institute of Technology has been set up as the Technology support centre under CoIL-Net and with reference to this IIT Kanpur is working on development of Hindi to English Machine Aided Translation System based on Anubharti approach.

A description about the Anubharti Machine Aided Translation System is presented here.

A Hindi to English Machine Aided Translation System Based on Anubharti Approach

1. Approaches to Example-based MT

The new generation of MT systems is characterized by making use of language corpus either explicitly or in an implicit way. A number of names such as 'statistical MT', 'translation memory', 'analogy based MT', 'memory based MT', 'example based MT' etc. have been used representing the manner in which language corpora have been utilized in the process. When the language statistical information collected a-priori (off-line) is used in the analysis of the source language and in synthesis of the target language, the methodology is broadly characterized as statistical MT [40] [2]. When a frequently occurring fragment of source language and corresponding pre-stored translation into target language is used in synthesizing the target language text, possibly with some help of human translator, it is referred to as using translation memory. Translation memory can be used in a variety of ways in a machine translation system [3] [4]. The terms analogy-based, memory-based and example-based are frequently interchangeably used to refer to the same basic methodology [5]. An excellent review of this methodology can be found in the review article by Somers [6]. The basic idea of machine translation by analogy with known translated fragments was given by Nagao [7]. There are three major stages of an EBMT system: _finding nearest match of the source language text with pre-stored examples; _finding the corresponding fragment in the translated example; and finally recombining these translated fragments to yield the target language text. Some of the major issues that need to be addressed in an EBMT system design are: creation of example data-base, usually derived from an authenticated parallel corpora; the size and granularity of example-base; the storage structure of the examples; matching, alignment and recombination strategies; evaluation of obtained translation; and the general issues concerning computational efficiency and speed. For some of the language pairs such as those of developing countries, a parallel corpora is simply not available or whatever is available is inadequate to construct a reliable example-base. In order to circumvent this problem, in our formalism, we have devised an interactive methodology for creation and augmentation of example-base.

As far as granularity of example-base is concerned, it is observed from [8] that the probability of match reduces as the length of the example increases, whereas a shorter example fragment may lead to greater ambiguity due to multiple matches. In addition, the shorter fragment may also result in poor accuracy due to inaccurate chunking and their context dependent meaning usually referred to as the boundary friction problem.

The size of the example-base may be reduced by combining similar examples primarily by tokenizing the constituents of the examples and deleting the duplicates. This process has been referred to as generalization of examples. Brown [9] has used this approach. Even prior to Brown [9], this approach has been used by the authors [28]1 where this has been referred to as the process of abstraction, i.e., the concepts behind the constituent constructs are being substituted with their abstract names. In our formalism, we have designed a _nite-state machine for identification of chunks. The examples are stored both at the chunk and sentence levels. Example abstraction (generalization) is performed both at the chunk and the sentence levels in an automated fashion. The storage of the examples is in linear form. The lexical knowledge associated with the constituents is used both to compute similarity as well as avoid boundary friction problem at the sentence level. The problem of alignment and most of recombination is taken care of by computing distances from sentence level examples. The price paid is in terms of a larger size of the example-base.

The speed and efficiency in our formalism is achieved by partitioning the sentence level example-base on the basis of main verb and by making use of linguistic knowledge in recombination for

constructs directly derivable from simple sentence constructs. Thus, our formalism is a hybrid approach as compared to the pure EBMT [10]. Some systems use annotated tree structure [11] [12] for representation and matching of examples. Kitano [13] uses segment map to provide explicit lexical links in examples. Brown [14] discusses inducing transfer rules in EBMT. The problem of alignment for automatic extraction of transfer mappings from parallel corpora is presented in McTait [15]. Complexity of introducing lexical knowledge in EBMT can be found in [16]. EBMT in a multi translation engine paradigm has been used in Pangloss system [9]. Yamabana et al. [17] use an interactive approach for hybridizing EBMT with RBMT. In contrast, hybridization in our formalism is in tandem: uses an interactive method for example growth and evaluation; a rule-based method for chunk identification; EBMT for fragment translation; and recombination by a mixture of rule and example-based approaches.

Our HEBMT(Hybrid Example-based MT) technique [1] used in Anubharti hybridizes the usual EBMT [5] and some aspects of KBMT [23] approaches to get the better results. While the basic translation engine is that of EBMT but the contents of the example-base are obtained through a shallow level parsing of input source sentence. The hybrid approach uses at least two levels of example-base matching instead of a single level of matching as in usual EBMT. At the first level, chunks are identified and translated using a chunk example-base and at the second level, translation of original sentence is generated by using the main example-base. Here the example-base consists of logical chunks obtained through different levels of chunk parsing [18] using a shallow level of the source language grammar. It is evident from above example that the sequence of six chunks does not represent the deep parse of the sentence.

In general, EBMT example-base consists of source and target language sentences in raw form, i.e., source and target language sentences are kept as it is. One way of reducing the size of example-base may be where instead of keeping the actual words in the example-base, the syntactic and ontological information of the each word is stored. This helps in mapping many words of same category to one example only. But, it also requires morphological analysis and extra processing to correctly identify the role of each word in the sentence. The use of chunks in the example-base further drastically reduces its size. Such an example-base formalism has been referred to as abstracted example-base. The level of abstraction may differ at the chunk and sentence levels. The associated syntactic, semantic and ontological information are used for abstraction. The abstracted example-base is actually a sequence of different types of chunks. A raw example, i.e., a source language sentence is transformed in the form of chunks by finite state machine and corresponding target language example is also stored in the form of chunks with special features of target language. Syntactic and semantic information of each chunk will depend upon the head word of the chunk and the head word of the chunk depends on the chunk type. It may be noted that it keeps all those examples for which source language example pattern may be same but due to difference in feature values of each syntactic unit, the target language example is different. Thus, the sentences having same structure and similar semantics may be translated with the help of only one example. Distinct examples are stored for the sentence structures which have multiple interpretations. This leads to a heavy reduction in the size of the example-base making it more efficient.

In order to reduce the matching time and to avoid incorrect matching, example-base has been partitioned on the basis of the type of main verb and the number of chunks in the input sentence. The decision to partition the example-base based on the type of main verb, is a conscious decision as Hindi and other Indian languages are highly verb-centric. This way the search for the best match is directed to a segment that is most appropriate making it more efficient and less error prone. An input sentence first goes to the correct partition on the basis of verb-type and the number of fragments in the input sentence. Examples from the partition are matched for same type and sequence of chunks as identified in the input sentence. Distances from the input sentence is calculated using a distance function, only for the similar examples using the syntactic and semantic knowledge of fragments. A

simple computational measure of semantic closeness between two words is their distance from each other in the type hierarchy. While the syntactic features are well understood and clearly defined, it is not so for the semantic features. Semantic knowledge of a constituent can differ depending upon the context. Also, it is difficult to judge the need of depth of knowledge. However, the success of finding the best match greatly depends upon the way these features are weighted for capturing the sentence context.

Most of the time, we can disambiguate words [19] that are ambiguous among syntactic classes by syntactically analyzing the sentence that contains them. For example, given the sentence 'The bottle dropped and broke', The word is ambiguous between its meanings 'fall by accident', 'become less', 'give up', 'lose' etc. This ambiguity can be resolved only by the context present in the sentence. As pointed out earlier, the distance function is a weighted sum of differences in syntactic, semantic and ontological category of constituents mapped on to a numeric value for comparison.

One of the major requirements of an EBMT system is the availability of an authenticated bilingual corpus. In absence of this, we developed an interactive strategy for augmentation of the example-base. Figure 1 illustrates the strategy. To begin with,

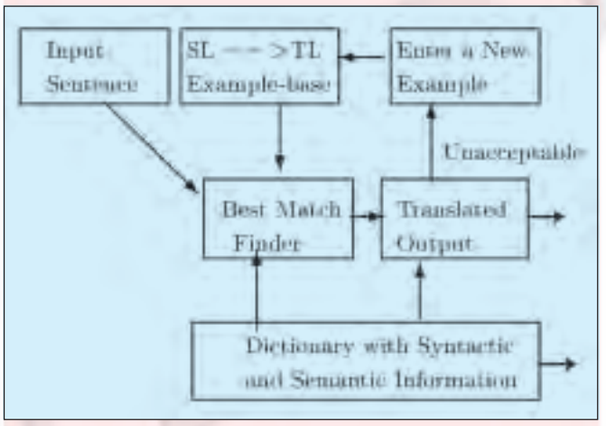


Figure 1: Interactive growth of example-base

the example-base consists of patterns taken from a standard grammar book and storing their manual translations. As the system gets exposed to more and more testing, the example-base grows through translator's intervention, in case the system generates unacceptable translations. Eventually the growth in size of example-base shows a saturating trend as evident from Figure 2. It shows the relationship

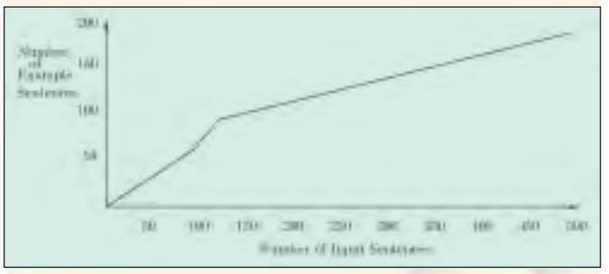


Figure 2: Rate of growth of Example-base

between the total number of sentences and the number of sentences entered as examples for the system developed for Hindi to English translation. Initially examples were taken from Hindi grammar book [20] which covers different kinds of routine patterns of the source language. Initial portion of graph shows such a trend. After that sentences taken from a general text and correspondingly we observe a jump in the example-base growth graph. However, after that ratio of number of new examples and input sentences starts decreasing. We observe that the growth graph exhibits a tendency towards saturation.

2. Anubharti : Hindi to English Translation System: Implementation Overview

This section elaborates on HEBMT approach used in implementing a Hindi to English translation system. Finite state machine is the core part of the HEBMT system. It identifies group of words called syntactic units or chunks of the sentence which behave as a single entity in the sentence. For the development of finite state machine, developer has to analyze many source language sentences and make rules to form the chunks. But, the job of formulating the rules for chunks is much simpler than developing the rules for analysis of entire sentence. HEBMT approach partially uses both the EBMT and HEBMT approaches and it tries to overcome shortcomings of both the approaches.

HEBMT differs from the actual corpus based approach because it does not require large bilingual corpus before building the translation system. System developer can start building the system with no example-base. Example base gets interactively built up as the developer does the testing of the system

and if the testing covers a large variety of sentences, a rich example-base is created. Actually developer can build different example-bases for various domains depending on the need of the user. Furthermore, the size of the example-base is drastically reduced since it does not store the examples in raw form.

Figure 3 represents the detailed architecture of HEBMT system. The design details of some of the major modules shown in the _gure have been elaborated in the following subsections.

Fragmentation of Input Sentence

The HEBMT system stores examples for simple sentences only. A sentence analyzer has been designed which identifies whether a given sentence is a simple, complex or compound sentence. Accordingly sentence is broken into simple sentences and clauses. These sentences are translated individually and put back into the target translation for generating the final translation of the sentence.

For example, the compound Hindi sentence:

jaika eka acchaa dosta hai lekin vaha kala jaa rahaa hai (Jack is a good friend but he is going tomorrow) gets broken into two simple sentences

'jaika eka acchaa dosta hai' (Jack is a good friend)

and

'vaha kala jaa rahaa hai ' (He is going tomorrow)

Similarly the complex sentence

vaha larakaa jo gaa rahaa thaa meraa bahuta acchaa dosta hai

(English sentence: the boy who was singing is a good friend of mine)

is broken into two sentences

'vaha larakaa meraa bahuta acchaa dosta hai.' (that boy is a good friend of mine)

and

'jo gaa rahaa thaa.' (who was singing)

Breaking of complex and compound sentences into simple sentences is done on the basis of keywords used as conjunctions and other semantic information. Heuristics are used to resolve the ambiguous cases. This part of the system is

still under development and testing.

Morphological Analysis

When an input sentence is fed for translation, the morphological analyzer identifies the proper words in the sentence and retrieves the necessary information about those words from the lexical database. Lexical database stores only the root form of the word and its syntactic and semantic information. With the help of paradigm files, root word is extracted from the original word and all the information about that word is retried. For example, only the word 'fly' is stored in lexical database and if the word ' flies' (or flying or flew) is found in the sentence, it is transformed to ' fly' and then matched in the dictionary. However, all derivatives for all words are not analyzed since it may not be possible to derive the correct meaning from the root word. For example, for the word 'nation', we do not keep 'nations' in the database but the words 'national' and 'nationality' are separately stored in the lexical database.

Chunk identification and FSM design

This is the core module of the system which makes use of knowledge based approach. The developer has to understand all types of the bindings possible among the words of a sentence. (S)he has to carry out the exercise of identifying the bindings for different types of sentences of source language. In simple words, it can be said that rules have to be formulated or the grammar needs to be defined for bindings for the source language. However, this grammar will be a very small as compared to the grammar needed to be designed for complete sentences.

Chunk parsing for Hindi Sentences

Chunk parser [21] has been implemented as a finite state machine to identify the chunks for Hindi sentences (HFSM). It uses the basic knowledge of words provided by the lexical database to find out the different types of bindings among the words of the sentence and then replacing them by groups. To start with, the finite state machine reads the category

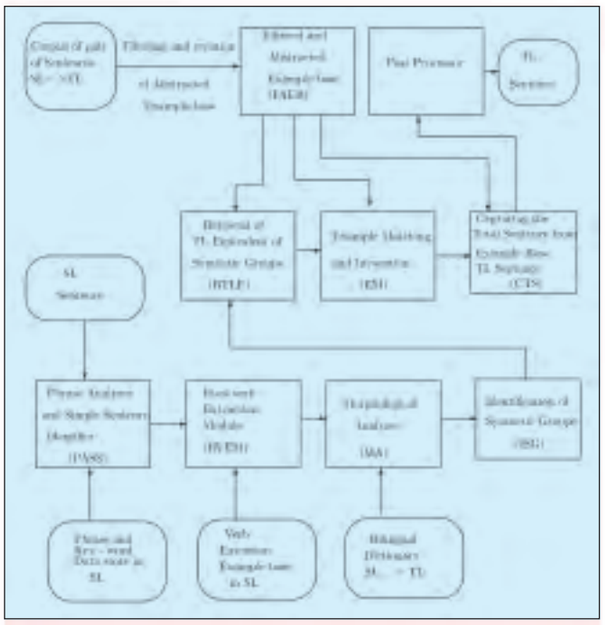


Figure 3: Architecture of HEBMT System

of the first word from morphological output, and generates certain expectations. If the expectation is fulfilled, expectations for the next word are generated, and this process continues till the proper syntactic unit or fragment or chunk is identified. After the identification of the syntactic unit, the finite state machine also stores all syntactic and semantic features about that syntactic unit and then the same procedure of identifying the syntactic units is repeated till the end of the sentence. Backtracking takes place and alternative categories are used to find the syntactic units whenever proper expectations are not met. But, if any category of the next word is not one of the generated expectations, the finite state machine declares the sentence to be illegal. HFSM here represents a deterministic parser where lexical ambiguities and parse ambiguities are handled respectively by postponing the decisions and by using heuristics. In case of more than one alternatives, the most frequent type of fragment is chosen. For example, if the _rst symbol is a 'possessive-case' (like meraa, mere, usakaa, etc.), HFSM will generate the following six expectations (since words or phrases belonging to any of these six categories can follow a possessive-case):

'common noun',

'post-positions' (lie, badr . aa, etc),

'noun form of a verb' (giranaa, lene, etc),

'adjective',

'adverb',

and

'auxiliary verb',

If the next word is not one of the generated expectations, HFSM tries to find out if the original word also belongs to any other syntactic category. If so, the HFSM tries out expectations according to this new syntactic category. Otherwise, the HFSM halts and concludes that the input sentence is syntactically incorrect. After the identification of syntactic unit, the finite state machine stores its type, its target language meaning, and other syntactic and semantic information, and then continues the processing the rest of the sentence.

Although, our discussion will be mainly focused on the finite state machine for Hindi language only, most of the discussion can be extended to other Indian languages also.

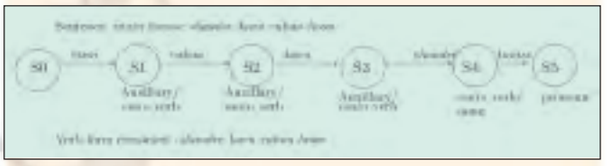


Figure 4: Extracting the verb form

Basically, HFSM can be divided into two finite state machines : one for identifying the verb chunk (verb phrase of the sentence) and another one for identifying other types of chunks (noun phrases, adverbial phrases, adjective phrases) of the input sentence.

Identification of verb chunks in Hindi

Hindi is a verb final language where verb phrases have exocentric close-knit construction. Like other languages, verbs in Hindi sentences also form the central part of the sentence and other constituents of the sentence revolve according to the type of the verb and its syntactic and semantic features. Verb phrases in Hindi are also constituted by two parts - main verb part and the auxiliary verb part. In most of the languages, generally main verb part is a single word which may be either the root verb or some

modified form of root verb. The auxiliary verb part depends upon the mood, aspect, tense and modal forms and may consist of one or more words. For example in English, we have verb chunks of types eat, will eat, have been eating, would have eaten, etc. It can be noticed from the above examples that the main verb is always one word (root verb or modified). But, in Hindi, the main verb part itself may consist of many words and each word, if examined separately, may be a verb, an adjective or a noun. This creates ambiguity in identi_cation of verb phrase in Hindi sentences. Different types of combinations have been discussed [20] for the main verb part. For example, in the following Hindi verbs-shaadee kara loongaa, shaadee kara rahaa hoon, the main verb is shaadee kara but kara, loongaa, rahaa may play the role of main verbs when constituted separately. Main verbs are generally combinations of nouns/adjectives plus some basic verbs. For example, the word 'intajaara' is a noun but 'intajaara kara' is a verb. Similarly, 'bimaara' is an adjective but 'bimaara ho' and 'bimaara kara' are verbs. Furthermore, in a verb phrase, the main verb part is followed by auxiliary verb part and this makes identification of verb phrase in the sentence even more difficult. All the above facts make the verb analysis complicated and at times, even ambiguous because of the various types of verbs formed from nouns and adjectives, and di_erent types of modi_cations that are possible in verb phrases.

After analyzing a variety of Hindi sentences, it was found that many ambiguities can be resolved and verb analysis can be done in a more systematic manner if we _rst identify the verb chunk and then the rest of the chunks are identified.

Therefore, initially verb-part of the input sentence is extracted with the help of verb analyzer (which is also imple-mented as finite state machine) and then the rest of the sentence is analyzed. Extraction of verb helps in identifying the type of the sentence and also in getting the correct form of the root verb. Therefore, a separate finite state machine(Figure 4.) has been designed for the identi_cation of the verb chunk. This machine starts operating from the end of the sentence as Hindi is SOV and tries to extract the auxiliary verb part and main verb part. Since 'O' in SOV can be part of main verb, properties of verb are used to generate appropriate expectations leading to correct identification of verb chunk.

Identification of other types of chunks in Hindi

Although Hindi is said to be a relatively free word order language, it is primarily true only for the syntactic units and not for the individual words. Another part of HFSM is the main chunk parser which joins those words of the sentence which cannot move independently in the sentence, but inter-related groups can move around in the sentence to a large extent without affecting its

Illustration 1:

Let the entry be:

H1. j*auna ne chora ko pakar . a liyaa hai.*

X1. John has caught the thief.

In the sentence H1, the verb-part is 'pakar . a liyaa hai', and the remaining part is 'jauna ne chora ko'. The verb-part is analyzed by verb-part synthesizer . In the remaining part, 'jauna' is recognized as 'noun' and 'ne' as a 'Vibhakti' sign (case marker) of type type-1. Therefore, 'jauna ne' becomes one syntactic unit or chunk, and it is named 'npk1' . Syntactic unit name 'npk1' indicates that it consists of noun phrase followed by a case marker of type-1. After this,

the next symbol is 'chora' which is a 'noun', and it is followed by 'ko', which is a case marker of type-2. Therefore, 'chora ko' is identi_ed as 'npk2'. Thus the _nal pattern of H2 is:

<npk1> <npk2> <mv>

Here 'mv' denotes the main verb.

Illustration 2:

Let us examine the entry:

H2. *jauna ke bar . e bhaaeene mairee ke sabase acche dosta ko maaraa thaa.*

X2. *John's elder brother had hit Mary's best friend.*

In the sentence H2, 'maaraa thaa' is recognized as main verb, 'jauna ke bar . e bhaaee ne' is recognized as npk1", because non-karak 'ke' follows the noun

symbol 'jauna', and this non-karak has the expectation of noun or adjective.

After finding the adjective 'bare', the new expectation is for a noun. This way, the HFSM proceeds and stops at the case marker 'ne'. Similarly, 'mairee ke sabase acche dosta ko' is identified as npk2".

Note that, after the identification of the syntactic groups, sentences H1 and H2 look exactly the same at the surface level. It also shows that 'jauna' can be mapped to 'jauna ke bare bhaaee' and'chora' can be mapped to 'mairee

ke sabase acche dosta'.

Figure 5 and 6 show the state transition graph (STG) of _nite state machine for two di_erent Hindi sentences H3 and H4.

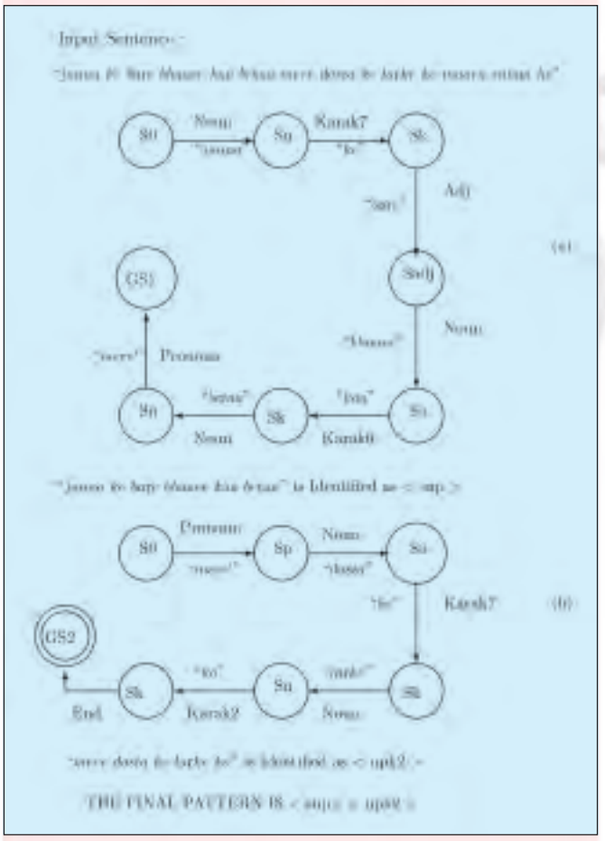


Figure 5: State Transition Graph 1

H3. jauna ke bar . e bhaaee kaa betaa mere dosta ke lar . ake ko maara rahaa hai.

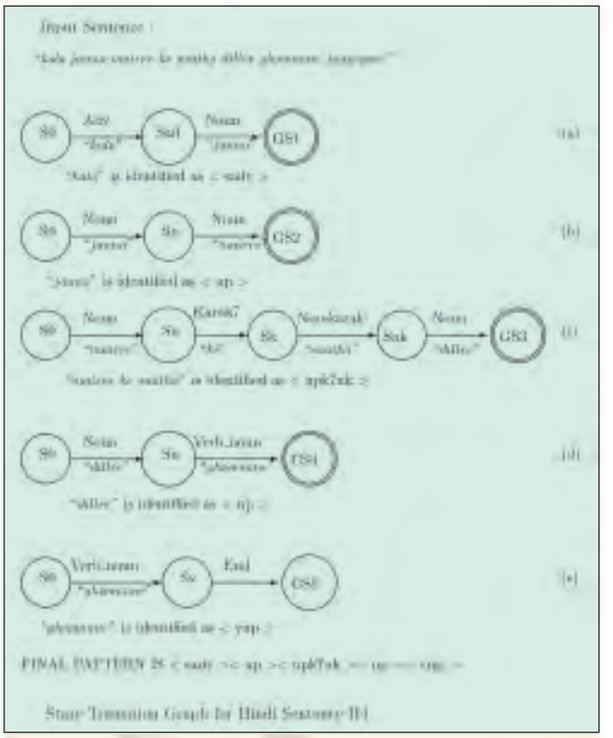


Figure 6: State Transition Graph 2

X3. John's elder brother's son is hitting my friend's son.

H4. kala jauna mairee ke saatha dillee ghoomane jaayegaa.

X4. Tomorrow John will go to see Delhi with Mary.

Translation of chunks in Target Language

In order to obtain the translation of a sentence, the syntactic units or chunks have to be translated in the target language after their identification.

In HEBMT system, we try to resolve the target language meaning of syntactic unit at the time of its identification only. Thus, the need for separate text generator is avoided and it makes the process of translation comparatively faster since HFSM does not have to store detailed information about each syntactic unit for text generator. It may appear from the above discussion that the translation of each syntactic unit is independent of the other. In practice, that is not the case. In most cases, syntactic and semantic knowledge about the successor words and the features of verb-part is used to identify the correct syntactic unit and its correct meaning.

Semantic knowledge of the constituent words of the syntactic category is used to pickup the most suitable meaning of those words. This is explained with very simple examples of Hindi to English translation when pronouns and their possessive forms are used in Hindi sentences. In Hindi, most of the proper pronouns and possessive-pronouns are 'gender' independent, and they also play the role of different syntactic units in different types of sentences. For example, the different roles of the pronoun 'vaha' are evident from the following example sentences.

vaha jaa rahaa hai.	He is going.
vaha pera hai.	That is a tree.
vaha lar . akaa jaa rahaa hai.	That boy is going.
vaha khaanaa khaa rahii hai.	She is eating food.
vaha pera lambaa hai.	That tree is long.
vaha meraa dosta hai	He is my friend.
vaha chora bhaaga gayaa	That thief ran away.
vaha chora kaa bhaaii hai	He is thief's brother.
vaha mere saatha rahatii hai.	She lives with me.

All the above sentences are using one pronoun 'vaha' which has di_erent meanings he/she/that and selection of correct meaning depends upon the type of main verb, semantic knowledge of adjacent words, etc.

Similarly, the various roles of possessive-pronoun (mere, meree) are illustrated in the following examples.

mere paasa chaara kitaaben hain
I have four books.

ye kitaaben merii hain
These books are mine.

pitaajii mere lie kitaaben laaye hain Father has brought books for me.

mere pitaajii adhyaapaka hain
My father is a teacher.

From the above examples, it is evident that knowledge of sentence structure and of the characteristics of verb-part, are necessary to disambiguate the correct role and the correct meaning of most of the pronouns and possessive-cases. The HFSM has been designed to take care of such contextual knowledge. This added feature makes the finite state machine language pair dependent, since many features of source and target language have been used to generate correct syntactic groups and their translations.

It is also observed that modifications in the finite state machine to cater to new sentence forms, not encountered earlier, are easy to carry out. The HFSM offers a mechanism for compressing the Hindi example-base from its raw form to an abstracted form which is used in the translation system. Although a lot of study has been done on Hindi grammar, to the best of our knowledge, this is the first time a system has been implemented for Hindi in a comprehensive form based on heuristics. Example-based method can also be used to translate chunks where either raw example-base or syntactic category based example-base may be used. This example-base will not be very large due to limited number of chunk types. Similarly, a target language text generator can be developed using knowledge based approach. Text generator has to synthesize the chunk and then translate. Further investigations on these lines are in progress.

Distance Evaluation for Hindi to English Translation System

After reaching to the appropriate partition of Hybrid example-base, and after identifying the structurally similar examples from that partition, distance is calculated between input sentence and all similar example sentences.

Example Sentence Group	Input Sentence Group	Distance
jauna (snp,[ani,mas, sing,third],[human])	mairee (snp,[ani,fem, sing,third],[human])	0.03
elisaa ko (npk2,[ani,fem, sing,third],[human])	somavaara ko (npk2,[inani,neu, sing,NA],[day])	0.73
par.haataa hai (mv,[transitive])	jaa rahee hee (mv,[intransitive])	0.5

Table 1: Distance Calculation

Translation of input sentence is generated on the basis of the target language example pattern of closest source language example pattern (with minimum distance value).

The distance function to retrieve the best example for a given input sentence plays an important role in determining the success of HEBMT or any example-based system.

The distance between input sentence and example sentence depends on attribute difference, status difference, gender difference, number difference, person difference, additional semantic difference, and verb category difference between the two sentences. Different weights are assigned to distance parameters according to their effect on translation. Additional semantic difference (ASD) is retrieved from the semantic difference table. Additional semantic difference table is generated based on hierarchy of semantic tags in the semantic network. Figure 5 illustrates a semantic network with ontology used in our system. Values to different weights can be assigned on the basis of experimentation of some corpus and then adjusting them dynamically as the example-base increases. However, in our system, weights have be en assigned based on examination and observation of a variety of source sentences. Further investigations are in progress to design a system which will learn these weights automatically through examples. This is itself a complex task. The values of weights assigned for now are ad-hoc, but provide reasonably good matching, as can be seen from the translated outputs shown in sample output.

An Illustration of Distance Computation

Procedure of distance evaluation has been explained by taking some example sentences and some input sentences. Example sentences have been denoted by 'X' and input sentences have been denoted by 'I'. Consider the following two example sentences which have the same pattern for source language sentence, but different target patterns from our sample example-base:

X1. jauna elisaa ko parhaataa hai.

X2. meraa dosta itavaara ko aayegaa.

Both of these examples have source pattern as '<snp> <npk2> <mv>'.

Target pattern corresponding to example X1 is '<snp> <mv> <npk2>'.

Target pattern corresponding to example X2 is '<snp> <mv> fong <npk2>'.

The following input sentences which also have the same pattern like the example sentences, were entered for translation. Each input sentence was matched with both example sentences, and distances were calculated. Target language sentence was generated using the target pattern of the example sentence which had lesser distance with input sentence.

I1. mairee somavaara ko jaa rahee hai.

I2. sudheera mere bhaaee ko peet . a rahaa hai.

I3. meree kulhaar . ee kisee bhee padaartha ko kaat . a sakatee hai.

I4. meraa chot . aa bhaaee mujhako pyaara karataa hai.

I5. usakaa bhaaee itavaara ko vaapisa aayegaa

Table 1 shows the individual distances between corresponding syntactic units of sentences X1 and I1.

The distances between the pairs of input sentences and example sentences are :

	I1	I2	I3	I4	I5
X1	1.27	0.53	1.47	0.57	1.52

SN	Translated English output	Matched Example No.
I1.	Mary is going on Monday.	X2
I2.	Sudhir was beating my brother.	X1
I3.	My axe can/may cut any material.	X1
I4.	My younger brother loves me.	X1
I5.	His/her brother will/shall come	X2

After matching each input distance with both examples, the target pattern of example with smaller distance from the input sentence was selected, and translation was obtained as shown in the Table 2.

It can be seen from the above examples, that the system is able to pickup the right example, though there may be situations when a wrong example is picked up due to inappropriate distance.

English Sentence Generation

After finding the example with minimum distance, the English sentence is generated using the target pattern of that example. Let us illustrate this by taking the input sentence I1 of section 4.5, which matches with example sentence X2. Steps 1(a) to 1(f) below, show the process of translation.

1(a). mairee somavaara ko jaa rahee hai {Input sentence}

1(b). <snp> <npk2> <mv> {Syntactic grouping}

1(c). (Mary) (Monday) (go) {English translation of syntactic groups.}

1(d). <snp> <mv> fong <npk2> {Target pattern of example X2}

1(e). (Mary) (is going) on (Monday) {Translation after substitution}

1(f). Mary is going on Monday. {Final translated output}

For target language sentence generation, the system substitutes the corresponding translations of chunks identified by the identified state machine in the target pattern. The appropriate form of verb part is generated only at this point.

System uses the 'tense', 'number', 'gender', 'root verb' to generate the correct form of verb.

It is obvious from the above discussion, that the process of example matching and target language generation are source and target language independent.

Post Editing

In some cases, the user may not be entirely satisfied with the translation, or he may get more than one possible translations. In that case, he may carry out the post-editing to get a required translation. The need and types of the post-editing can be illustrated by the following examples. In the following sentences, the system provides options to the user to select the proper word according to the context and need.

1a. This cloth will/shall last long. (yaha kaparaa kaaphee chalegaa)

1b. This cloth will last long. (Correct Translation)

2a. He/She gave meapen.(usane mujhe paina diyaa)

2b. He gave me a pen.(Correct Translation)

OR

2c. She gave me a pen. (Correct Translation)

However, in the following sentences, the user has to add appropriate articles to make the translation more appropriate.

It may be noted that Hindi and all Indian languages do not use articles.

3a. This is pen. (yaha kalama hai.)

3b. This is a pen. (Correct Translation)

4a. Servant has broken glass.(naukara ne gilaasa tora diyaa hai.)

4b. Servant has broken the glass. (Correct Translation)

Although most of the time post-processing is either of the above two types, sometimes, the user may have to make minor changes in the translated sentence to correct it grammatically. In some cases, the system fails to find the correct 'number' of the subject due to the exceptional behaviour of source language, and in other cases, it becomes difficult for the system to find the correct 'number' of a noun because that noun has the same form in the case of plural verb also. This is illustrated in the translated outputs shown below.

5a. My father are going to Kanpur. (mere pitaajee kaanapura jaa rahe hain.)

5b. My father is going to Kanpur. (Correct Translation)

Please note that the system has produced 'are' due to the presence of 'jaa rahe h ~ ain' in the input sentence, which is the correct literal translation. However, in this case, the plural form of the verb is used in the Hindi sentence to show respect for an elderly person. Since no such provision is available in English, the user expects only a singular

form, i.e., 'is'.

6a. Both brother were studying in nearby school.(don~ o bhaaee paasa ke skoola m~ e par . haate the.)

6b. Both brothers were studying in nearby school. (Correct Translation)

In this case, the system produces 'brother' in place of 'brothers', since the Hindi sentence has used the singular form of the noun bhaaee. If the word used in Hindi was 'bhaaeey~ o', we would have got the perfect translation.

It is evident from the above post-editing examples that though sometimes post-processing of the translated output is required, the time and effort required for the post-editing will be quite small. The user needs to know only the target language to perform the post-editing, he doesn't need to have any knowledge of the source language for it.

The system is currently able to identify and translate the following types of sentences besides the simple affirmative sentences, provided that the pattern is found in the examplebase.

1. Sentence with more complex verb forms such as:

- तुम्हें यह काम करना पड़ सकता है ।

  You may have to do this work.

- तुम्हें परीक्षा के लिए अध्ययन कर लेना चाहिए ।

  You should study for examination

- यह पाठ राम के द्वारा याद किया जा रहा है ।

  This lesson is being learnt by Ram.

- यह पाठ कल याद किया जा सकता था ।

  This lesson could be learnt yesterday.

2. Sentences with negation such as:

- राम अपने माता पिता की आज्ञा नहीं मानता है ।

  Ram does not obey his parents.

- मेरी दादी ने कल रात मुझे कोई कहानी नहीं सुनायी ।

  My grandmother did not tell me any story yesterday night.

- वह कल तक इस कार्य को समाप्त नहीं कर पायेगा ।

  He will not be able to finish this work till tomorrow.

- राहुल सचिन और सौरभ के साथ अनिल के घर कभी नहीं जायेगा।

  Rahul will never go to Anil's house with Sachin and Saurabh.

3. Interrogative sentences such as:

- क्या यह प्रणाली प्रश्नवाचक वाक्यों का अनुवाद कर सकती है।

  Can this system translate interrogative sentences?

- हम इस वाक्य को अंग्रेजी में कैसे अनुवाद करेंगे।

  How will we translate this sentence in English?

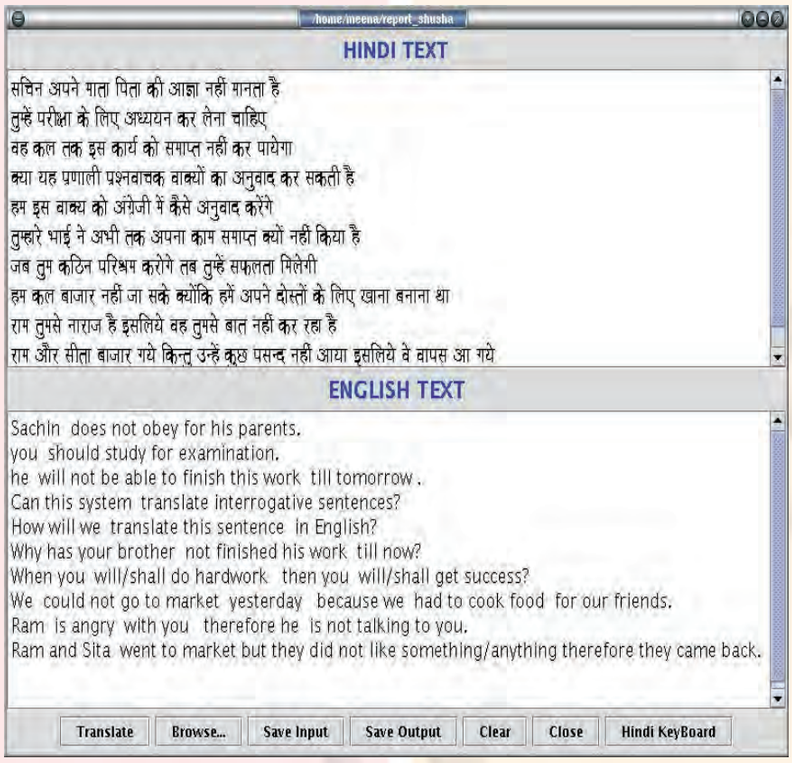
- तुम्हारे भाई ने अभी तक अपना काम समाप्त क्यों नहीं किया है ।

  Why has your brother not finished his work till now?

- तुम कल मेरे साथ बाजार क्यों नहीं जा रहे थे

  Why were you not going to market with me yesterday?

Screen shot for sample output



4. Compound sentences formed by joining complete simple sentences by coordinate conjunctions :

- हम कल बाजार नहीं जा सके क्योंकि हम अपने मेहमानों के लिए खाना बनाना था

  We could not go to market yesterday because we had to cook food for our guests.

- जब तुम कठिन परिश्रम करोगे तब तुम्हें सफलता मिलेगी

  When you will do hardwork then you will get success.

- मोहन तुमसे नाराज है इसलिये वह तुमसे बात नहीं कर रहा है ।

  Ram is angry with you therefore he is not talking to you.

- राम और सीता बाजार गये किन्तु उन्हें कुछ पसन्द नहीं आया इसलिये वे वापस आ गये ।

  Ram and Sita went to the market but they did not like something/anything therefore they came back.

Future Plans

We are working on the translation of complex sentences and translation of Compound Sentences which are not formed by the combination of complete simple sentences such as

राम ने खाना खाया और सो गया ।

हमें बुरे कार्यों से घृणा करनी चहिये और सदैव अच्छे कार्य करने चाहिये ।

Website for HindiAngla Translation System will also be launched soon.

Courtesy : Prof. R.M.K. Sinha
Prof. Ajai Jain
Department of CSE
Indian Institute of Technology,
Kanpur 208 016 (UP)
Tel. : 0512-2597174 (O), 0512-2590260 ,
0512-2590007
E-mail : rmk@iitk.ac.in
ajain@iitk.ac.in