

HEBMT: A HYBRID EXAMPLE-BASED
APPROACH
FOR MACHINE TRANSLATION
(DESIGN AND IMPLEMENTATION
FOR HINDI TO ENGLISH)

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

By
Renu Jain

Under the Supervision
of Prof. R.M.K.Sinha
Department of
Computer Science and Engineering
I.I.T. Kanpur

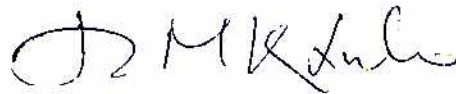
BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA

1995

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI RAJASTHAN

CERTIFICATE

This is to certify that the thesis entitled “HEBMT: A Hybrid Example-Based Approach for Machine Translation (Design and Implementation for Hindi to English)” and submitted by “Renu Jain”, ID. No. 92PHXF014 for award of Ph.D. Degree of the Institute, embodies original work done by her under my supervision.



DR. R.M.K.SINHA
PROFESSOR
Department of Computer Science
& Engineering,
Indian Institute of Technology,
Kanpur.

Date : July 4, 1995.

Synopsis

The recent inventions in the areas of communication and transportation has made the world as a global village, by reducing the time to communicate between two ends of the globe to almost zero. However, the multiplicity of languages is still a big barrier in the interchange of ideas, thoughts and culture among the people belonging to different geographical and cultural regions on the earth. There have been continuous efforts by computer scientists and linguists for overcoming this barrier by automating the process of translation, thus making Natural Language Processing as one of the major thrust areas of research in applied Artificial Intelligence. In the past few years, the work on machine-translation has received great attention of researchers because of its potential applicability and technological advancements which has made the design of such systems feasible.

The main problem faced by researchers and system developers in processing any natural language is due to the inherent ambiguities in the structures, references, and meanings of words. People are able to eliminate these ambiguities on the basis of the knowledge of the context of the sentence and exact perception of objects represented by words. However, the task of incorporating the knowledge of context, and of the exact perception of objects in specified context into the computer is very difficult. Since natural language is a language used by natives for their communications, it may not be feasible to capture all possible constructs and eliminate illegal constructs by using a formal grammar, thus making the job of a Machine Translation System more difficult. Some other features of natural languages which make automated translation difficult are: multiple meanings of a word, multiple syntactic categories of a word, structural differences of a sentence from one language to another, peculiarities of each language, ambiguity in pronoun reference, and identification of proper rules for the use of markers like prepositions, articles, post positions etc. which differ from one language to another.

There are many approaches to machine translation presented in the literature. In this thesis, we have categorized them into two main classes: traditional approaches

and memory-based approaches. The schemes following the traditional approaches use a set of hand crafted rules to reflect the movement of substructures from source language to target language. Translation into the target language is obtained by using a text-generator. On the other hand, in memory-based systems, the emphasis is on exploiting language models based on corpus, statistics, and examples. Memory-based systems obtain the translation from source language to target language by finding the closest example from the example-base, and making word or phrase level transformations.

In this thesis, we analyze both approaches and discuss their relative merits and demerits. The discussion motivates us to propose a new approach to machine translation which tries to capture the best of both worlds. This leads to the design of the Hybrid Example-Based Machine Translation System (HEBMT), which has been presented in detail in this thesis. The design has been implemented and tested for Hindi to English translation.

Some of the main distinguishing features of the HEBMT system are as follows:

1. Utilization of filtered and abstracted example-base from corpus of examples.
2. Efficient searching using partitioning of example-base.
3. Simple matching function to invoke the best example.
4. Use of syntactic groups in the input sentence for matching and transfer to the target language.

The process of system development, implementation details, and the process of translation are briefly discussed in the following paragraphs.

The system development starts with the creation of an example-base. The example-base in any example-based system contains the examples of pairs of sentences from the source and target languages. However, the HEBMT differs from other example-based systems in the manner in which these pairs are stored. Most example-based systems store the examples in raw form, which are actual sentences in their natural form in the source/target language. However, in the HEBMT, the examples are stored in an abstracted form. An abstracted example is a combination of syntactic units, where each syntactic unit is a group of words or a single word, such that there is a tight binding between the words of a syntactic unit, and this remains as it is in the target language also. It has been shown in the thesis that storing the examples in the abstracted form reduces the size of the example-base as well as the time required for finding the best match, thus giving us a both time and space efficient system.

The creation of an abstracted example-base and the process of translation are inter-related. Initially, an input source language sentence is entered for translation,

and finite state machine transforms the sentence into an ordered set of syntactic groups. Then, the pattern of abstracted source language sentence is searched in the example-base. If a match is not found, this abstracted sentence along with the abstracted example in target language is entered in the example-base, leading to the dynamic growth of the example-base. If a match is found, the system translates the source language sentence into target language according to the abstracted target language example. However, if the translation is not upto the satisfaction of the user, the user can enter a new example with a different target abstracted example for future use.

In order to translate an input compound sentence, the sentence is first broken into two or more sentences, which can be simple or complex. The type of the sentence is identified on the basis of key words using heuristics. The compound sentence is translated by combining the translations of the constituent simple or complex sentences.

Since most of the Indian languages are verb final languages, it is simpler as well as more accurate to extract the verb part of the sentence from the end of the sentence. An expectation driven module has been developed to analyze the verb-part of the sentence. Along with the root verb and its other morphological information, this module also returns the 'verb-type', 'tense' of the sentence, the 'number' and the 'gender' of the subject. After the verb-extraction, the input sentence is passed to the morphological analyzer, where proper words of the source language are identified, and the target language meaning, syntactic information and semantic information are retrieved from the lexical database.

Since the examples are stored in their abstracted form, the input sentence also has to be transformed to its abstracted form. We use a finite state machine for performing this transformation by identifying the syntactic units of the sentence. To start with, the finite state machine reads the category of the first word and generates certain expectations. If the expectation of the next word matches with any expectation, the expectations are generated for the next word, and this process continues till the proper syntactic unit is identified. After identification of the syntactic unit, the finite state machine writes all the information about that syntactic unit and halts. The finite state machine is then restarted for identifying the syntactic units in the remaining part of the sentence. If, at any stage, the category of any word doesn't match with any expectation, the sentence is rejected indicating the inadequacy of the input sentence. The translation of each syntactic unit in the target language is obtained by a simple text generator which makes use of a small rule-base combined with the finite state machine to increase the efficiency of the

system.

After the abstraction of the input sentence, the abstracted sentence is matched with the abstracted examples from the appropriate partition. All examples with exact match are retrieved. The distances of the input sentence from all the matched examples are calculated with the help of syntactic and semantic information of the corresponding syntactic units. A heuristic based human engineered distance function has been defined to evaluate the distance. All examples having minimum distance are invoked for translation. Finally, the translation is obtained by substituting the target language translations of the constituent syntactic units in the best matched example (the example with minimum distance from the input sentence). If needed, post editing of the target sentence can be done for further refinement. The user doesn't need to have any knowledge of the source language for performing post-editing.

We have implemented the HBMT system from Hindi to English translation. The system has been tested by creating a sample example-base consisting of about 200 sentences from various domains, and many input sentences have been successfully translated with reasonable accuracy. The results presented in the thesis are very encouraging, and support our hypothesis that a good translation is possible using this hybrid example-based approach. As expected, initially we need to enter many examples before getting the correct translation of any input sentence. However, when example-base starts growing, more and more input sentences are successfully translated. The results also prove the efficiency of the hybrid system in terms of the reduction of the size of the example-base due to the abstraction of examples (in place of storing the raw examples). We are confident that if the system is implemented on a full scale by creating a larger database and expanding the example-base by including examples from other domains, it will provide reasonably accurate translation. Since many of the modules are language independent, the system can be extended for translation between other language pairs.

Acknowledgments

It is a pleasure to thank my thesis supervisor Prof. Sinha for his constant help, encouragement and support throughout the course of this work. Prof. Sinha first introduced me to this problem and motivated me to work on it for which I am grateful to him. I would also like to thank Prof. Raju, Unit Chief, IPC for motivating and encouraging me to work in the wonderful area of Parallel Processing and Artificial Intelligence. I am grateful to Prof. R.C. Jain, for providing me continuous moral support and help. I also thank Prof. L.K. Maheshwari, Dean R & C, and Mrs. Rajni Garg for their constant help. I would also like to thank Prof. B.N.Patnaik, Prof. Suraj Bhan Singh and Prof. Subba Rao for many enlightening discussions in the area of NLP, which helped me a lot in dealing with the complex problem of Machine Translation. I would also like to thank Mrs. Veena Bansal, Mr. Rakesh Kumar Srivastava and Ms. Sarita Agarwal, for many fruitful discussions and their help at the time of writing the thesis. I would also like to thank Dr. Kiran Biswas for proof reading my thesis. I thank Mrs. Veena Sharma, Mrs. Sudha Shukla and Ms. Deepti Kushwaha for data entry, which helped me in testing of the system. Finally, I would like to thank my husband and two wonderful sons for cheerfully bearing with my whims, and my long absences from home.

Renu Jain

Contents

List of Tables	xi
List of Figures	xii
1 Introduction	1
1.1 Organization of the Thesis	6
2 Approaches To Machine Translation	7
2.1 Strategies used in Machine Translation	9
2.1.1 Direct Translation Strategy	9
2.1.2 Transfer Strategy	10
2.1.3 Interlingua Strategy	11
2.2 Current Trends in Machine Translation	13
3 HEBMT : A Hybrid Approach to Machine Translation	18
3.1 Introduction	18
3.2 HEBMT: A Hybrid example-based approach to Machine Translation .	21
3.3 System Architecture	23
3.4 Source and Target Language Dependency of the HEBMT Modules .	31
4 Morphological Analysis	32
4.1 Features of a Natural Language	32
4.2 What is Morphological Analysis?	33
4.3 Difficulties Encountered in Morphological Analysis	34
4.4 Strategies for Implementing a Morphological Analyzer	35
4.5 Morphological Analyzer for Hindi	36
4.5.1 Analysis of the Verb-Part	36
4.5.2 Analysis of the Remaining Part of the Sentence	38
4.5.3 Analysis of a Hindi Sentence	38

4.5.4	Verb-part Synthesizer	41
4.5.5	Noun-part Synthesizer	42
5	Creation and Arrangement of Example-Base	44
5.1	Concept of Dynamic Creation of Example-Base	45
5.2	Creation of Abstracted Example-Base	45
5.3	Partitioning of the Example-Base	50
6	Identification of Syntactic Units in a Sentence	52
6.1	Main Features of Hindi Language relevant to the Identification of Syntactic Units	54
6.2	HFSM : A Finite State Machine for Identifying Syntactic Units in Hindi Sentences	55
6.2.1	Illustrations	57
6.2.2	State Transition Graphs of the HFSM	60
6.3	Translation of Syntactic Units in Target Language	60
7	Example Matching and Target Language Sentence Generation	65
7.1	Semantic Tags	65
7.2	Lexical Database used for HEBMT and Multiple Meaning Disam- biguation	67
7.3	Matching Process and Design of Distance Function	70
7.4	Illustration of Distance Procedure	73
7.5	Target Language Sentence Generation	75
8	Implementation and Experimentation	77
8.1	Implementation of Modules of the HEBMT System	78
8.2	Experimentation	87
9	Conclusions and Discussions	91
9.1	Features of HEBMT Approach	91
9.2	Limitations and Suggestions for Future Work	93
	References	99
	A List of Semantic Tags	102
	B HFSM Symbols	105

C	Sample Entries from Lexical Database	107
D	Examples of Translation for different Verb-forms	111
D.1	Examples of Translation of Input Sentences in the Present Tense . . .	111
D.2	Examples of Translation of Input Sentences in the Past Tense	112
D.3	Examples of Translation of Input Sentences in the Future Tense . . .	112
D.4	Examples of Translation of Input Sentences in Modal forms	113
D.5	Examples of Translation of Input Sentences with Different Forms of Noun Phrases	113
E	Mapping file for Different Tenses	115
F	Sample Entries from the Raw and Abstracted Example-Base	121
G	Sample Output of Translation from Hindi to English	131

List of Tables

2.1	Overview of MT History	12
4.1	Example of derivatives of some words	33
6.1	Karaks and Post-Positions	55
6.2	Post-Positions and Karaks / Non-Karaks	56
6.3	Input Hindi sentences	59
6.4	Identified Syntactic Units of Hindi sentences	59
7.1	Distance Calculation	74
7.2	Example Matching	75

List of Figures

2.1	Basic Architecture of EBM'T [EH91]	16
2.2	Overall Architecture of MBMT System[H.K93a]	17
3.1	Architecture of an example-based approach	19
3.2	System Architecture	24
5.1	Algorithm to create example-base	46
6.1	State Transition Diagram for the sentence H5	61
6.2	State Transition Diagram for the sentence H6	62
7.1	Semantic Tree	68
8.1	Information Flow Diagram of RVEM	79
8.2	Information Flow Diagram of ISG module	81
8.3	Information Flow Diagram of MA module	83
8.4	Information Flow Diagram of EM module	85
8.5	Rate of Growth of Example-base	89

Coding Scheme for Hindi Alphabets Used in the Thesis

a	ā	i	ī	u	ū	ri
अ	आ	इ	ई	उ	ऊ	ऋ
e	ē	o	ō	~	=	
ए	ऐ	ओ	औ	अ	आं	
k	kh	g	gh	ñ		
क	ख	ग	घ	ङ		
c	ch	j	jh	ñ		
च	छ	ज	झ	ञ		
ṭ	ṭh	ḍ	ḍh	ṇ		
ट	ठ	ड	ढ	ण		
t	th	d	dh	n		
त	थ	द	ध	न		
p	ph	b	bh	m		
प	फ	ब	भ	म		
y	r	l	v	sh		
य	र	ल	व	श		
ṣa	s	h	ksh	gya		
ष	स	ह	क्ष	ज्ञ		
ṛa	ṛha					
ठ	ड					

Chapter 1

Introduction

Natural language is the language as used by natives for communication among themselves. As opposed to formal languages used as computer languages, a natural language is full of ambiguities. While humans are capable of interpreting and understanding the language inspite of the ambiguities using a wide variety of world knowledge and contextual information, it poses a challenge for the computers. Natural language processing (NLP) is concerned with the study of various aspects of a language with a view to develop systems with wide range of applications [Jos91]. These applications include NLP integrated speech and document understanding systems, interfaces to databases and knowledge bases with human-like interactions, multilingual interfaces, machine translation etc. Machine translation is one of the oldest applications of NLP. As soon as modern digital computer was invented, the researchers started exploiting its potentiality for automating the translation process. The hostilities during second world war further fuelled research and development efforts in this direction.

Even in the present context, multiplicity of languages is a barrier to interchange culture and thoughts between the people of the earth, and is a serious deterrent

to international understanding. Furthermore, the rapid growth in the international trade among various countries also necessitated the increased research and development efforts in the area of automated translation by exploiting the technological advancements in the design and development of high speed computers with large storage and processing capabilities.

Besides this, in a multi-lingual country [JJR95] like ours, breaking the language barrier or providing better and easy linguistic interfaces assume much greater importance from national integration, development, and prosperity view points. This thesis is motivated keeping such a necessity in view.

All natural languages are inherently ambiguous in their structure, with ambiguities in references and meanings. People are able to eliminate these ambiguities on the basis of the knowledge of the context of the sentence and the exact perception of the objects represented by the words. It is automatic for a person to understand the sentence based on its context. However, it is a challenge to computer scientists to incorporate such knowledge including context and exact perception of objects in a specified context in the computer. Another limitation of any natural language is that it is not possible to represent natural language in the form of a formal grammar. Some of the features of natural languages which make the translation a complex task include: multiple meanings of a word, multiple syntactic categories of a word, ambiguity in pronoun reference, differences in structures of the sentences from one language to another, peculiarities of each language, ambiguity in sentence structure, and difficulties in identification and formation of proper rules for the use of markers like prepositions, articles, post-positions etc. which differ from language to language etc.

A lot of research has been carried out in the field of machine translation for

the design and development of efficient and accurate translation systems. The techniques used have primarily been based on formal grammar and/or heuristic rules. The traditional strategies used for machine translation can be broadly classified into three categories [A.B87]: direct translation strategy, transfer strategy, and the interlingua strategy. Some of these systems also make extensive use of AI (Artificial Intelligence) techniques. However, despite the long history of machine translation and technological breakthroughs, it has not been yet possible to develop a satisfactory translation system mainly due to the ambiguities involved in a natural language.

Generally, the traditional approaches to machine translation use a set of hand crafted transfer rules /formal grammar to reflect the movement of substructures from source language to target language. The final translation into the target language is obtained using a text generator. In general, the following observations have been made about the traditional models:

- . They need all the knowledge to be presented explicitly using rigid and formal representation schemes.
- . They use rule driven inferencing.
- . Rules and knowledge are generally hand-crafted by the experts.

The traditional models of AI for machine translation are not suited for the problem size needed for real world applications due to the following [H.K93b] :

Incompleteness: A good translation system using traditional AI models needs a complete knowledge about the structure and grammatical rules of both source and target languages, which is almost impossible to obtain. Most of the time the experts themselves do not have the complete knowledge. Many times knowledge may be available, but can not be expressed in a formal manner. Thus, a certain

portion of the knowledge is always absent while developing a machine translation system.

Incorrectness and Inconsistency: A translation system can be accurate only if the knowledge used is itself correct as well has been encoded correctly. It is very difficult to guarantee either of these two. Expert knowledge itself may be inconsistent because some of the contextual factors have not been taken into account. Thus a large knowledge base is bound to have some errors.

Scaling from prototype systems: A very common misconception about many traditional AI models that a highly intelligent system developed in a restricted domain can be very easily scaled up with larger funding and increased effort, has been proven wrong in the field of machine translation.

The bottlenecks outlined above make the traditional approaches impractical for large systems. The above observations forced researchers to place memory as a basis for intelligence rather than fragile hand crafted rules. The approach followed by many researchers and developers in the current generation of machine translation, has been to make extensive use of language models based on corpus, statistics, examples, and constraints. The technological developments in the area of computer hardware with the invent of high-speed massively parallel processor, made it feasible to store and analyze the large amount of information needed for the purpose. Out of these models, the example-based techniques received greater attention of MT researchers and developers. This is due to the fact that examples offer a natural and simple way of illustrating the translation process.

Even in a formal teaching environment, many concepts of a language are explained only on the basis of the standard examples of that language since it is not possible to define exact rules for them. Therefore, very often we translate sentences

from one language to another on the basis of our past experiences of translation. This natural process of learning is the basic philosophy behind the example-based approach of machine translation. It is imperative that MT researchers explore this avenue, which is distinct from traditional approaches followed so far, and many researchers and developers of current generation are doing exactly the same.

After studying the history of machine translation and various approaches used for it, we observe that though it is easy to implement rules in computer for translation purposes, formulating perfect and comprehensive rules for any natural language is almost impossible. A natural strategy is to start from the source, i.e., start from the examples from where the grammatical rules themselves evolve. The methodology of example-base translation is in line with this philosophy. However, the main criticism of example-based approaches is due to the large size of the example-base that will be needed for a practical system. This may require enormous storage, and may take very long time for searching the huge example-base to find and invoke the best match for accurate translation.

In this thesis, our attempt has been to explore the use of example-base approach, and design a practical system, which compresses the example-base using a hybrid technique. The hybrid technique referred as the Hybrid Example Based Machine Translation (HEBMT), attempts to make the best use of traditional as well as memory-based techniques. Using this hybridization, we are able to significantly reduce the size of the example-base by a process of abstraction. The technique has been implemented for Hindi to English translation. A sample example-base has been developed using sentences from various domains. The experimental results strengthen our confidence in our approach as it provides a fairly accurate and efficient translation. Though, in this thesis, we mainly discuss the implementation details

for Hindi to English translation, as will be evident during the discussions in the later chapters, most of the modules themselves are either independent of source and target languages, or can be easily modified to suit other language pairs.

1.1 Organization of the Thesis

In Chapter 2, we present history of machine translation and discuss in detail, various strategies and approaches used so far for machine translation, with their advantages and limitations. In Chapter 3, we present the basic architecture of the HEBMT system proposed by us, with the description of the functionalities of its main modules. In Chapter 4, the process of morphological analysis has been discussed with special emphasis on the analysis of a Hindi Sentence. The creation and the management of the example-base used by the HEBMT has been described in Chapter 5. In Chapter 6, we present the mechanism for identifying the syntactic units in a sentence using a finite state machine. The process of matching an input sentence with the best example in the example-base using a distance function, and then generating the translation in the target language has been discussed in Chapter 7. In Chapter 8, the implementation details for the main modules of the HEBMT have been presented. The chapter also includes the details about the testing of our system, and results have been presented showing the compression in the size of example-base by using hybridization technique. In Chapter 9, we present conclusions and suggestions for future work.

Chapter 2

Approaches To Machine Translation

Historically, the work on machine translation (MT) started at the same time when the modern digital computers came into existence. MT was the first nonnumeric application which attracted the attention of many computer scientists. Between years 1940 and 1960, many efforts were made in the area of machine translation. However, the researchers could not succeed in achieving their goal of building good quality fully automated translation system [Nir87]. The main mistake of early MT developers was that they underestimated the need of and complexities involved in understanding a natural language for machine translation. After a deep analysis of translation problem, it was found that the variety and the amount of knowledge used for the solution to this problem is enormous. Researchers also realized that good quality fully automated systems in broad domain are not feasible. However, it is possible to design automated systems which are restricted to a particular domain, or which take help of a human translator at some point. Therefore, most of the systems developed are either Machine-aided Human Translator (MAHT) systems or

Human-assisted Machine Translation (HAMT) systems.

Realization of the complexities involved in MT and the need to have some degree of text comprehension ([Hil60], [Com66], [JRA81]), forced the developers of machine translation systems to drop the idea of developing fully automated translation systems and to focus on the more pragmatic objective of building systems that increase the throughput efficiency of human translators. In this context, some of the directions in which the work progressed are as follows:

Translation aids:

One approach is to improve the efficiency of a valuable, experienced human translator by providing him with high-powered computational tools for time consuming tasks. Such tools range from split screen editing systems to format documents and from graphic layout modules to on-line technical dictionaries and grammar checking programs. Such systems are called Machine-aided Human Translators.

Post-editing approach:

Another approach is to have a human translator working as a post-editor. In this approach, there exists a translation system which translates the source text into the target language, but it produces rough translation and the post-editor (an expert human translator) fixes the errors in the translated output. Since the post-editing requires significantly lesser time compared to that needed for complete translation, human efficiency is increased. Post-editing approach is still being used, but efforts are being made to have a user (who has the knowledge of target language only) as a post-editor instead of a human translator. Such systems are called Human-assisted Machine Translation Systems.

Pre-editing approach:

Many a time during translation, the translation system is unable to interpret the

source text correctly due to the inherent ambiguities. Errors in the interpretation lead to incorrect translations. Therefore, the pre-editing approach tries to eliminate the difficulties such as complex grammar structures, ambiguous words etc., to minimize or to avoid the effort of the post-editor. Such systems are also called Human-assisted Machine Translation Systems.

2.1 Strategies used in Machine Translation

Three major strategies have been round the corner for the design of MT systems over the last three decades [A.B87], namely, direct translation strategy, transfer strategy, and interlingua strategy.

2.1.1 Direct Translation Strategy

The direct translation system is designed for a specific source and the target language pair with no intermediate representations. The source language text is translated into the target text with almost word to word translation. Afterwards, minor adjustments are done according to the target language structure. No general linguistic theory or parsing principles are necessarily present for the direct translation to work. These systems depend on well developed dictionaries, morphological analysis, and the text processing software to get the translation. Many systems were developed using 'direct approach', but only very few were used commercially. A system was developed by the Logos Corporation for the US Air Force to translate American aircraft manuals into Vietnamese. Xonics and PAHO systems were experimental. However, the best known system of direct approach is SYSTRAN [P.T77], designed initially for Russian-English, and later adapted for English-French translation system for the Commission of the European Communities. ANUSARAK [Nar95] being

developed at I.I.T.Kanpur uses a modified form of direct approach to translate from Kannada to Hindi. This approach of ANUSARAK is applicable only to similar language pairs.

2.1.2 Transfer Strategy

In the transfer strategy, a source language sentence is first parsed into the phrases according to the structure of the source language. Thereafter, a 'transfer' is made either at the lexical level, or at the grammatical level, or at the structural level built by the grammar, into corresponding structures in the target language. In the third stage, the translation is generated. Three dictionaries are needed for transfer: a source language dictionary, a bilingual dictionary, and a target language dictionary. The level of transfer differs from system to system. The representation varies from purely syntactic deep structure markers to syntactico-semantic (compositional semantics, case frame information, and others) annotated trees. Note that the transfer stage involves a bilingual component, i.e., a component tailored for a specific SL-TL pair. This adds a relatively significant level of complexity in a multilingual environment since a transfer block will have to be written for every such language pair. This approach has been in use for a long time. However, many attempts have been made to examine the possibilities to work with language independent representations. The experimental examples of this approach are Russian-English project at EURATOM and Moscow Patent Office system for translating American patent abstracts into Russian, POLA system for Chinese to English at Berkley and the MIAN system for French-Russian. This strategy was popularized by the systems like SUSY [Maa84], TAUM-METEO for translating weather reports from English to French in Canada (by Canadian broadcasting), GETA system at Grenoble and

Eurotra project of European Communities.

2.1.3 Interlingua Strategy

An alternative approach to transfer approach was to develop an universal, language-independent representation for text, known as interlingua. The interlingua-based approach depends upon the fact that a suitable intermediate representation can be defined for the source text which can then be mapped into the target text. This intermediate representation is presumed to have resolved all ambiguities, and so it should be possible to generate text in any language at the target from this representation. This approach has two phases: analysis phase and generation phase. The analysis phase is source language dependent but target language independent. The generation phase is only target language dependent. In principle, we can dispense with bilinguality. For a multilingual system with n source languages and m target languages the transfer approach will require $m * n$ transfer blocks (if the sets of source languages and target languages are disjoint), in addition to n analyzers and m generators. In the interlingua approach, only n analyzers and m generators will be needed. However, it is very difficult to define a truly language independent intermediate representation. Current interlingua-based systems have used some combination of transfer-based approach and interlingua approach. None of the interlingua-based systems claimed the universality of the their intermediate representation. For a high quality translation, mostly, it is necessary to have access to some particular aspects of the source and the target languages, and it is not clear that how these aspects can be made available during generation without explicitly encoding into intermediate representation.

CETA (Russian-French project) and LRC (German-English project) were early

Year	Grammar for sentence analysis	Main Systems	Translation Strategy
1960's	CKY Parser, Early Parser, LR Parser	GAT(RE), Yamato(EJ) , SYSTRAN (RE, EF, FE, EI)	Direct Translation
1970's	ATN Parser, Chart parser, DCG Interpreter, Case structure Parser	CETA(RF), METAL(GE, RE), TAUM- METEO(ER) Eurotra, Smart(EF) etc.	AI based approach and Transfer approach
1980's	Unification Parser	DLT(Esp), TITRAN(EJ), ATLAS(JE, JG), Logos(GE, EG), LUTE(JE, EJ), ANGLABHARTI(EH)	AI based transfer and interlingua approach
1990's		Prototype (EJ), TDMT(EJ)	Example-based approach

Table 2.1: Overview of MT History

systems which used the interlingua approach. In the late seventies, the concept of the intermediary representations, formulated in terms of logical formulae was found in the SALAT project at the university of Heidelberg, and in the Philips project at Eindhoven in Netherlands. KBMT at CMU, DLT, Rosetta and project by CICC also used the interlingua approach. ANGLABHARTI [RKA⁺95] system being developed at I.I.T Kanpur, from English to Indian languages is one of the recent systems using a combination of transfer strategy and to some extent, interlingua approach.

Table 2.1 presents a historical perspective of evolution of major MT systems/projects. Table 2.1 represents a historical perspective of evolution of major MT systems/projects.

Abbreviations:

C=Chinese, E=English, Esp=Esperanto, F=French, G=German, H=Hindi I=Italian, J=Japanese, R=Russian, S=Spanish.

2.2 Current Trends in Machine Translation

A new generation of machine translation began, when the emphasis shifted from linguistic consideration to exploiting language models based on corpus, statistics, and examples [J.H93]. In the era of fifth generation computers, MT researchers diverted from forming explicit rules for translation. Attempts are being made to develop statistical techniques and memory-based techniques to perform translation using corpus of pair of languages, which were translated by human translators. An MT system has been developed using statistical techniques with the use of Hansard bilingual corpora (Hansard contains transcripts of the proceedings of the Canadian Parliament both in English and French). It was found that statistical techniques would be more feasible if developed for a specific application. It would be very difficult to develop a general translation system using statistical techniques. Currently the trend is to use memory of the system and the corpus of parallel text and apply analogy principle to get the translation. The problem of translation is examined and analyzed from a human learning point of view. The thrust of example-based translation is to use examples of past translation to produce translation of given input through an analogical process [JRJ95b]. In the following paragraphs we illustrate the role of examples in a human learning process.

Let us consider the scenario, where one is trying to acquire a second language. To begin with, the person acquires few meanings of a few words, which expands with additional examples gained through the experience with passage of time. When

exposed to an example translation from first to second language, it is observed that when a “similar” sentence is presented, (the similarity being judged only among the constituents of the first language), the subject is able to translate it according to known sentences and with the knowledge of the meanings of the words involved. As an illustration, if one knows the target language translation¹ of these two input sentences,

E1. He goes home in the evening. (Example English sentence)

H1. *vaha shama ko ghara jatā hē.* (Target Hindi sentence)

E2. They are waiting in a hotel. (Example English sentence)

H2. *ve hotala mē intajāra kara rahe hē*². (Target Hindi sentence)

In the pair (E1 → H1) mapping is as follows:

He → *vaha*,

goes → *jātā he*,

home → *ghara*, and

‘in the evening’ → *shama ko*.

Then it is possible for the subject to construct translations for the sentences “*We take lunch in the afternoon*”, “*She came back in the night*” according to example sentence E1. The subject will also translate the sentences “*He was sitting in the car*”, “*We will take our lunch in a restaurant*” correctly, by substituting meanings after selecting the closest example E2.

If the second example “*They are waiting in a hotel*” is not known, it is not surprising that one would obtain a translation corresponding to the first example sentence E1. But, when corrected by someone, one more example sentence gets

¹Here, the target language is Hindi. The sentences marked with ‘E’ are English sentences and the sentences marked with ‘H’ are Hindi sentences

²Note that “in” of first example E1 is translated as “*ko*”, and “in” of second example E2 is translated as “*mē*”

added to the subject's database for future use. After knowing sentences of the type, "*He wants milk*" and "*He does not want water*" and "*Do you want milk*" etc., the subject acquires the feeling of the grammaticality of the language in the implicit form. Then it becomes possible to derive translation of similar sentences such as "*Mary likes mangoes*", "*They do not eat apple*" and "*Does Mary enjoy soup*" etc. Note that one does not have to memorize all valid possible combinations of all known words. A very small subset of sentences, named example sentences, will suffice since the other sentences can be generated, when needed, using example sentences. Right choice from example sentences will be done on the basis of the type of sentence and semantics of the sentence constituents. However, it can not be denied that each natural language has its own grammar and its rules, with some exceptional cases. And, in the above illustration also, the person is learning rules, but he is not aware of that.

Same theory is being used for MT, where rules of the language are not fed explicitly, instead, the system acquires the rules from the examples in implicit form. This way, MT system developers do not have to represent the knowledge in the form of explicit rules. Identification of the set of rules which are unambiguous and complete, and coding of those rules such that the system interprets them correctly, are almost impossible for an human being for any natural language.

Example-based approach was first proposed in [M.N85] by analogy. Then, the same idea was presented as memory-based reasoning by [CD86] and case-based reasoning by [RS90]. [EH91] proposed a EBMT (Example-Based Machine Translation) system. Figure 2.1 depicts the basic architecture presented by [EH91].

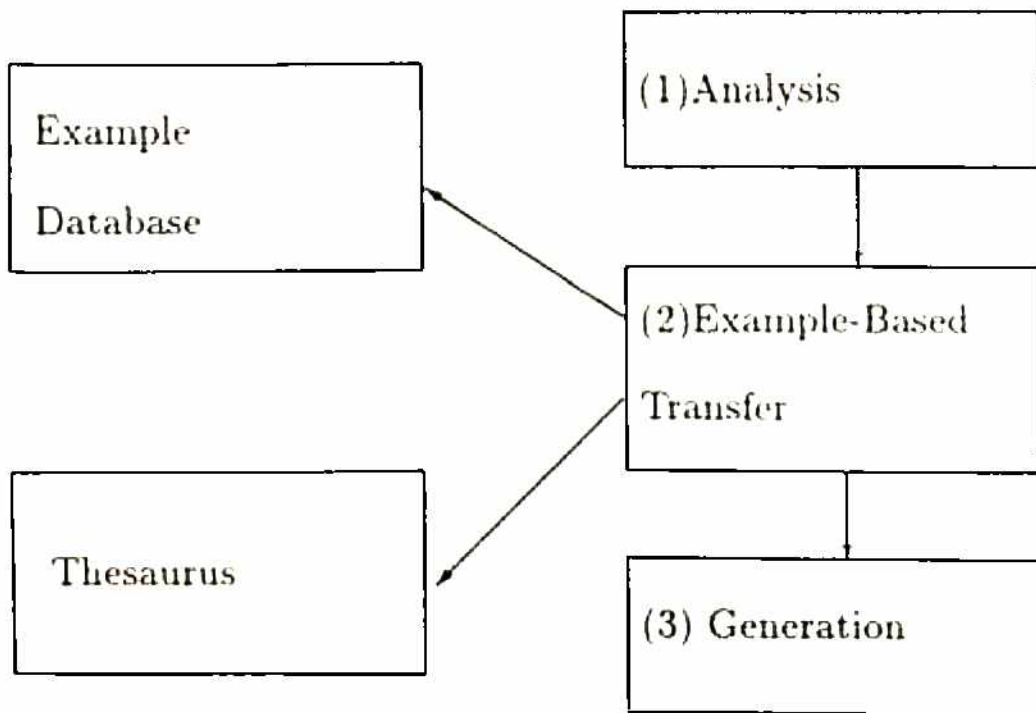


Figure 2.1: Basic Architecture of EBMT [EH91]

[OH92] developed a "Transfer-Driven MT" system for spoken language translation using EBMT approach. [SM90] highlights on the problem of utilizing more than one example for translating one source sentence. [H.K93a] has presented a modified architecture (Figure 2.2) for MBMT (Memory Based Machine Translation) system. Kitano has used rules for handling specific language phenomenon (like zero-pronoun in Japanese-English translation) and to refine the translated output.

[H.M93] argues that pattern matching can be performed as a form of pure context-free parsing and also proposes an algorithm called Context-free Transducer for a very large set of patterns. Example-Based system can work very efficiently by installing the system on massively parallel processors, has been shown by [EKH⁺93]. They have also shown that example-based systems get a good amount of accuracy

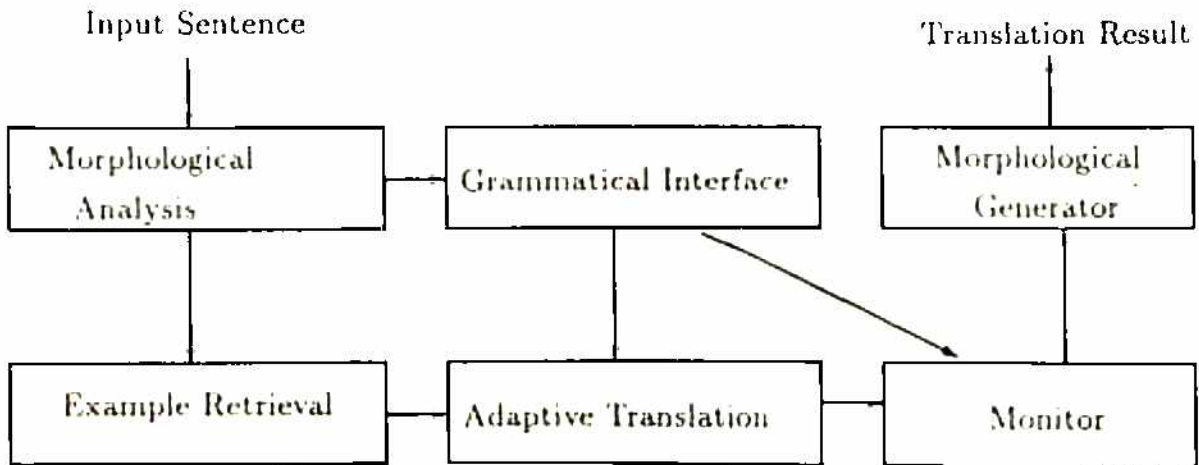


Figure 2.2: Overall Architecture of MBMT System[H.K93a]

in comparison to traditional systems when implemented on massively parallel machines.

In example-based systems, translation is attempted from the raw examples by invoking the best suited example using a distance measure. A raw example is the actual sentence of that language in its natural form. The example-base of EBMT system is the actual corpus of source language to target language translations. To get a practical EBMT system, a very large example-base is needed. As matching time of input sentence to the source language sentence is directly proportional to the size of example-base, system efficiency of a practical EBMT system is reduced considerably. The job of designing a powerful distance function which can pickup the closest example from the example-base is also quite complex. This complexity grows as the size of the example-base grows.

Chapter 3

HEBMT : A Hybrid Approach to Machine Translation

3.1 Introduction

The HEBMT system [JRJ95a], proposed by us and presented in this thesis, uses the basic philosophy of example-based approach of Machine Translation. The basic idea behind any example-based approach is to use similar past examples for translation. It assumes that all different events are stored in memory, and response to a new event is handled by first recalling past events which are similar to the new input. In an example-based approach, rules of the source and target language are acquired from the example sets in implicit form. Source and target language example pairs are stored as they are, in the database, called example-base. For translation, the input sentence is matched with example sentences of the source language. A distance function is defined to calculate the closeness of the example sentence with the input sentence. Example target sentence corresponding to the closest example sentence (sentence with minimum distance) is invoked to translate the input sentence. Figure 3.1 represents the general architecture of an example-based approach.

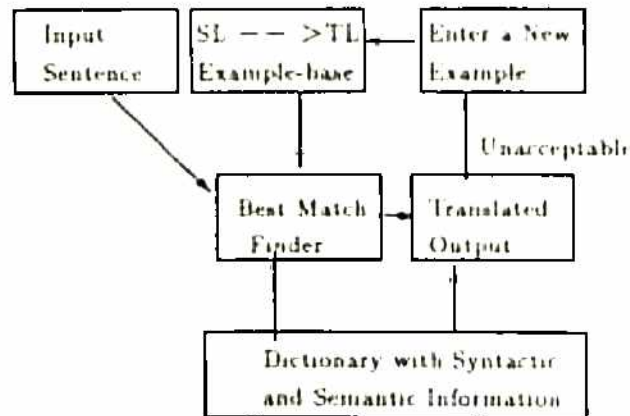


Figure 3.1: Architecture of an example-based approach

The example-based approaches, in their present form, have following shortcomings which we have tried to overcome using the proposed approach presented in the next section.

- Example-base used by existing example-based system is the collection of example pair sentences (source and corresponding target sentences) in their raw form. In general, the EBMT systems require a very large database of examples due to the entry of example sentences in raw form. Since the time needed to match the input sentence with examples from the example-base directly depends on the number of examples, this time increases with the growth of the example-base.

- To evaluate the distance between the input and the example sentence, distances between the corresponding content words are evaluated. Therefore, even if an example of exactly the same type is not present in the example-base, the distance is calculated on the basis of statistical information, and the target sentence is generated. Whereas an intelligent system is expected to point out the inadequacy of the example-base (absence of a suitable example), based on some criteria. It may not

be feasible to develop fool-proof criteria to indicate this inadequacy of the example-base. However, it is possible to provide limited feedback based on syntactic grouping and semantic tagging of the words of the source sentence. This can be explained more clearly by the following example:

Suppose our example-base contains the following four sentences:

1. Ram saw a girl in the market.
2. Ram is going to a very big market.
3. Son of Ram is my best friend.
4. I have borrowed four story books from Shyam.

Even though all four sentences contain 7 words each, they are quite different in nature. If the system matches the input sentence with all examples having number of words equal to the number of words in the input sentence, all four examples will be tried for matching to translate any input sentence with 7 words inspite of their different natures. It is possible that the distances from some of these examples may be within allowable limits set by the system designer, and therefore, the system will give a translation which can be quite absurd due to the inherent differences between the nature of input sentence and examples. An intelligent system should try to match only those examples which are similar to the input sentence in nature, so that the most appropriate match is obtained by calculating the distances between the content words of input sentence and appropriate examples only.

- In many EBMT systems, during the example matching, the first word of the input sentence is matched with the first word of example sentences. Next word of the input sentence is matched with the second word of all those examples which had the distances below the threshold distance. This process of matching continues till all the words of input sentence are exhausted. The maximum word length in a commonly

used sentence is about 25 words [H.K93b]. If we assume the average length of a sentence in a text to be 10 words, and it is matched with 1000 examples, on average 10000 comparisons will be required for translating every sentence. Therefore, the matching time itself can be significantly large for the development of a practical translation system.

- Expert's knowledge, in the form of rules, is needed for the correct choice of word meaning. Though these rules are different from Rule-Based MT system, yet for a practical system, it may not be possible to incorporate all the knowledge in the system. Furthermore, this additional large knowledge base will further slow down the translation process. Since the encoding of such knowledge in the form of rules has to be hand crafted, it also requires a tremendous amount of human effort to encode the knowledge.

3.2 HEBMT: A Hybrid example-based approach to Machine Translation

The HEBMT approach proposed by us, has following features for overcoming many of the problems mentioned in the previous section:

- *Abstraction of the examples to reduce the size of example-base:*

Abstraction of examples is done by identifying the syntactic units in the sentence. Process of abstraction is quite different from the parsing. While parsing a sentence, all possible paths are tried out, and the whole sentence is analyzed to get the phrases of the sentence. On the other hand, the abstraction is a sequential process which identifies the simple bindings between the adjacent words. An examination of Appendix D reveals the importance of the abstraction. As illustrated there, the

Appendix D, that with the help of only 4 example sentences, our system is able to translate 50 sentences. It is possible to translate many more sentences, which differ in the verbs, subjects and objects, provided they have similar abstracted patterns, using these four examples only.

- *Partitioning of example-base, and matching of syntactic units instead of word to word matching to reduce the search-space:*

We divide the entire example-base into several smaller partitions. Each partition is again recursively divided into several sub-partitions to achieve several levels of partitioning. This drastically reduces the search space in which an input sentence is matched. At the first level verb-type is matched, at the second level the number of syntactic units is matched and at the third level the type and ordering of each syntactic unit is matched. If an exact match is not found, the system stops the matching process, and requests the developer (or user) to enter a new example corresponding to that input sentence. It was also observed during the testing that, on average, each sentence has about 3 to 6 syntactic units. Therefore, even if the system tries matching with 500 examples (though this number will be smaller even in a practical system due to the partitioning and abstraction), at most 3000 matches will be required, thus significantly reducing the response time of the system.

- *Using semantics for word sense resolution:*

In case of a word having multiple meanings and therefore, several translations in target language, the correct choice is done by taking into account the semantics of the words within syntactic units or by using MBT [SM90] approach. For example, the syntactic unit '*choṭā beṭā*' is translated as 'younger son', and not as 'small son', by matching their semantic tags. Lexicon entries of these words are given in Appendix C. MBT [SM90] approach uses an example-base of fragments of sentences

for obtaining the correct word meaning. For example, the word '*calanā*' has many meanings like 'run', 'drive', 'operate', 'walk', and these can be resolved by matching with fragments '*dukana calana*', '*kara calanā*', '*topa calāna*', and '*pedala calāna*'.

In HEBMT system, multiple verb meanings are resolved by the example-base itself. Different examples are entered corresponding to different verb meanings, whenever required. A verb of source language maps to different target verb meanings depending upon the type of the subject, the object, and the structure of the sentence itself, and sometimes it also depends upon the tense of the verb.

Every language has a variety of sentences. Each sentence consists of many syntactic units and each syntactic unit is a group of specific kinds of words arranged in a specific order. In general, a sentence differs from another sentence firstly due to its surface level structure, and secondly on the type of inner structures of phrases. Two source sentences, which are the same at surface level, may or may not have same translation at the target level. On the other hand, two sentences which are not the same at structure level, will always have entirely different translation at the target level. This is another main concept used in the proposed HEBMT system.

3.3 System Architecture

Figure 3.2 depicts the overall system architecture of HEBMT system. An input compound sentence is first broken into two or more sentences, which can be simple or complex ¹.

¹In example-based systems, for translating the compound sentences, firstly, the translation of the first part of the sentence is achieved by matching in the example-base. Then, the other part of the compound sentence is matched and translated. Finally, both the translated portions are combined to get the final translation of the compound sentence. In HEBMT system, at the first level itself, the sentence is broken into simple sentences and the information about conjunctions and the order and the number of simple sentences, is stored, and in the end, all simple sentences are joined to get the translation of compound input sentence.

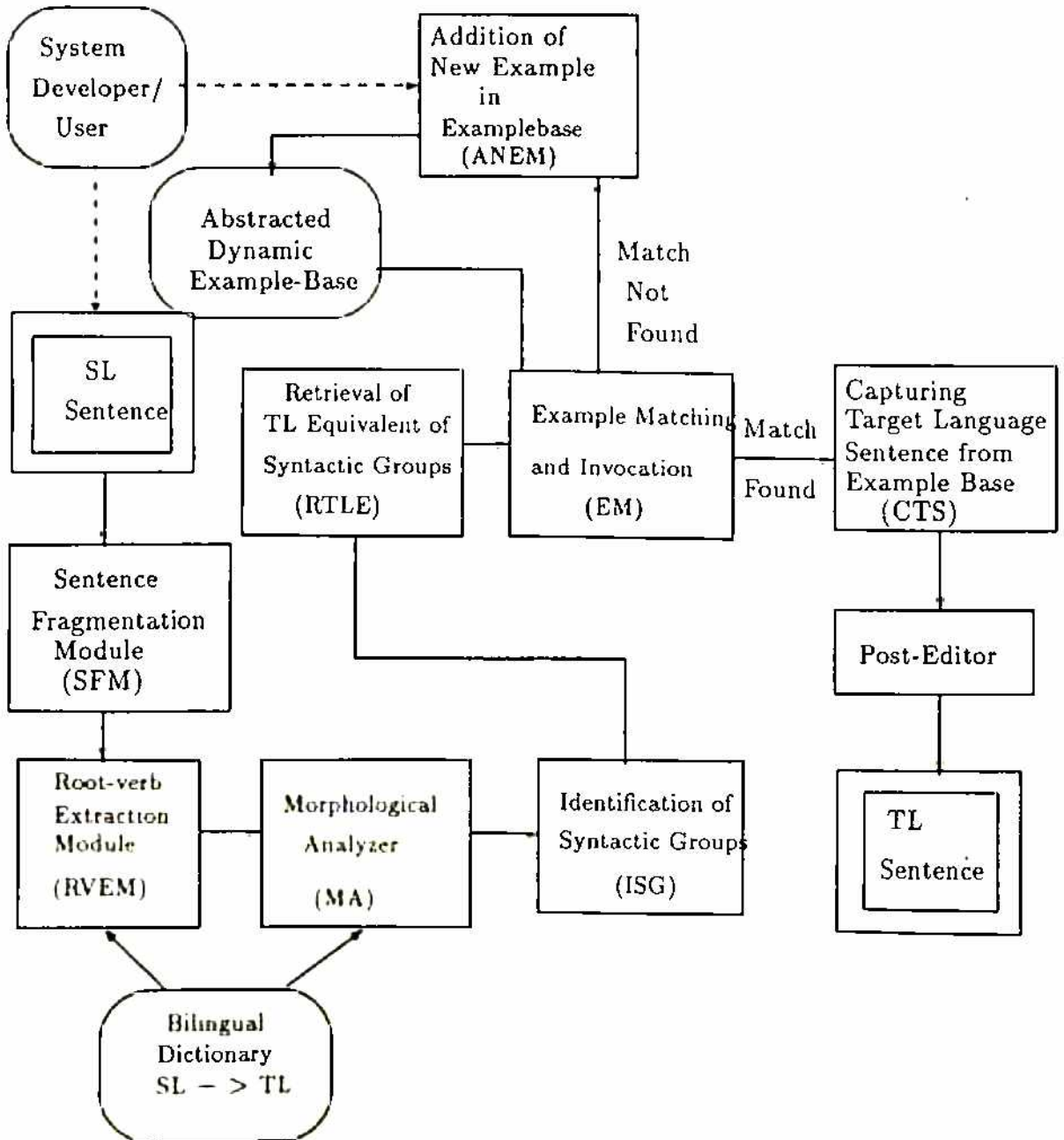


Figure 3.2: System Architecture

The verb-part is extracted using a verb extraction finite state machine. Hindi is a verb final language and verb-part in a Hindi sentence is analyzed from the end of the sentence. After the verb extraction, morphological analysis is done for each word of the sentence, and lexical database is searched to get the syntactic category and the English meaning of each word in the sentence. A finite state machine identifies the syntactic groups in the source sentence and generates the translated syntactic groups. A distance matrix between the input sentence and the sentences from the appropriate partition of the example-base is computed, and the example with the minimum distance is invoked for getting the translation of the input sentence. The main modules shown in Figure 3.2 are briefly explained in the following paragraphs. The details of many of these modules have been presented in the following chapters.

1. Addition of New Example in the example-base Module (ANEM) :

When an input sentence is fed for translation, the system translates it by identifying the best matched example and produces a translation. However, if the proper match is not found (or the user is not satisfied with the produced translation), a new example in the existing example-base is added by this module. A system developer does not have to build an example-base. Example-base gets built by itself, as the developer starts the testing of input sentences. Actually, the developer feeds input sentences from the set of examples on the basis of which he wants to built his example-base. A new example is added to the example-base in the abstracted form with all the syntactic and semantic information, and the developer is asked to enter the pattern of target language example sentence. More details about the creation of and addition to the example-base are given in chapter 5.

2. Sentence Fragmentation (SF) :

This module takes the original input sentence as its input, and checks if the input sentence is a compound sentence. Type of the sentence is identified on the basis of some key words using some heuristics. If the sentence is found to be a compound sentence, it is broken into two or more simple or complex sentences. All the required information about the conjunctions, and the number of simple sentences is stored by this module. Other modules are called to translate these simple sentences, and this process continues till the translations of all simple sentences are obtained. Afterwards, the translated outputs are combined to obtain the translation of the original compound sentence.

3. Root Verb Extraction Module (RVEM) :

Most Indian languages are verb final languages. Therefore, it is simpler as well as more accurate to extract the verb-part of the sentence from the end of the sentence. An expectation driven module (details in chapter 4) has been developed to analyze the verb-part of the sentence. Along with the root-verb and its other morphological information, this module also returns 'verb-type' and 'tense' of the sentence, and the 'number' and the 'gender' of the subject.

4. Morphological Analyzer (MA) :

The input sentence is first passed to a morphological analyzer for analysis. In the input sentence, proper words of the source language are identified, and the target language meaning, syntactic information and the semantic information is retrieved using the lexical database Appendix C prepared for the translation purpose. Details of this module are presented in chapter 4.

5. Identification of Syntactic Groups (ISG):

This module transforms the input sentence into an abstracted form. The module has been implemented as a finite state machine. An input sentence is transformed to an abstracted form by identifying the syntactic units of the sentence. To start with, the finite state machine reads the category of the first word from morphological output, and generates certain expectations. If the category of the next word is not one of the generated expectations, the sentence is rejected and the finite state machine halts. If the expectation is fulfilled, expectations for the next word are generated, and this process continues till the proper syntactic unit is identified. After the identification of the syntactic unit, the finite state machine writes all syntactic and semantic information about that syntactic unit and halts. The finite state machine is then restarted, and the same procedure of identifying the syntactic units is continued till the end of the sentence is found. The details of the finite state machine for Hindi to English translation are included in chapter 6.

6. Retrieval of Target Language Equivalent of Syntactic Groups (RTLE) :

Translation of each syntactic unit in the target language is obtained by this module. There are several possible methods to develop this module. For example, one can have a separate example-base [SM90] to translate the syntactic units. Alternatively, one can have a separate text generator which makes use of a small rule-base to translate the syntactic units. In our implementation, we have used the rule-based method, which has been combined with the finite state machine to increase the efficiency of the system.

7. Example Matching (EM) : After the abstraction of the input sentence,

the abstracted sentence is matched with the abstracted examples of the appropriate partition. Appropriate partition is found on the basis of the main verb-type and the 'number' of syntactic units. In that partition, input abstracted sentence is matched with all abstracted examples, and all examples which exactly match the syntactic pattern, are retrieved. If exact match is not found, a new example is added by invoking the 'ANEM' module. If the match is found, the distances are calculated for all matched examples. To find the distance, syntactic and semantic information of the corresponding syntactic units are used. A heuristic based human engineered distance function has been defined to evaluate the distance. All examples having minimum distance are invoked for translation. The process of distance calculation and example-matching has been explained in more detail in chapter 7.

8. Capturing Target language Sentence (CTS):

Finally, when the best matched example is found, this module produces the final translation by combining the translations of the constituent syntactic units according to the pattern of the abstracted example, as illustrated in more detail in chapter 7.

9. Post Editor (PE):

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 refined 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 kapara kaphi calega*)

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

2a. He/She gave me a pen. (*usane mujhe pena diya*)

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

OR

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

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

3a. This is pen. (*yaha kalama he.*)

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

4a. Servant has broken glass. (*nōkara ne gilāsa tora diya he.*)

4b. Servant has broken the glass. (Perfect 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 also . This is illustrated in the translated outputs shown below.

5a. My father are going to Kanpur. (*mere pitāji kanapura jā rahe hē.*)

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

Please note that the system has produced 'are' due to the presence of '*ja rahe hē*' 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. (*donō bhāi pāsa ke skula mē parhate the.*)

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

In this case, the system gives 'brother' in place of 'brothers', since the Hindi sentence used the singular form of the noun *bhai*. If the word used in Hindi was '*bhāiyō*', we would have got the perfect translation.

There is one more type of post editing where system is not able to choose the proper meaning of a noun, or an adjective or an adverb. In such cases also, alternative meanings are provided and the user picks up the right meaning.

For example,

7a. Doctor has asked patient to come yesterday/tomorrow. (*daktara ne marīja se kala āne ko kahā hē*)

7b. Doctor has asked patient to come tomorrow. (Perfect Translation)

8a. Father listened all matter/thing. (*pitāji ne sarī bāta sunī*)

8b. Father listened all matter. (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.

3.4 Source and Target Language Dependency of the HEBMT Modules

When we examine the proposed architecture for its suitability in a multilingual environment, we find that the following modules are independent of the source and target language. Module 'ANEM' is completely language independent. Only the entry of target language pattern will be target language dependent, but it does not effect the module (i.e., coding in the programming language) at all. Similarly the modules, 'EM', 'CTS', 'PP' are completely language independent. The module 'MA' uses many suffix files which are source language dependent, but the module itself is source/target language independent. Modules 'SF' and 'RVEM' are source language dependent, but these will require only small changes for verb final source languages. However, the modules 'ISG' and 'RTLE' are heavily dependent upon the language pair, i.e., the source language as well as target language, and will have to be rewritten for other source/target language pairs.

Chapter 4

Morphological Analysis

4.1 Features of a Natural Language

The words in any natural language constitute the atomic units which convey information regarding parts of speech, number, person, gender etc., besides their meaning(s). All natural languages have a large vocabulary of words. However, a large number of words are derivatives of the original word. For example, in English, 'pulls', 'pulled', 'pulling' have originated from the root word 'pull', and 'earlier', 'earliest' are the derivatives of the adverb 'early'. Similarly, Hindi also has a large number of derivatives of their corresponding root words. Sometimes, the *parsarga* or *vibhakti* sign is also combined with the word and needs to be detected in order to find the root word. Examples of derivatives for a few root words are shown in the Table 4.1.

A lexical database is a specially organized list of lexical units of a language, supplying specific information for a specific purpose. A lexical database is equivalent to a specialized dictionary. Hence, while developing the lexical database or a dictionary for any language, an efficient way is to store only the root words with

Root word	Part of speech	Derivatives
<i>beca</i>	verb	<i>becanā, becā, beci becakara, becane, becata, etc.</i>
<i>larakā</i>	noun	<i>larake, larakō</i>
<i>acchā</i>	adjective	<i>acche, acchi</i>
<i>usa</i>	pronoun	<i>usakā, usake, usamē, usako usane, usapara, usase, etc.</i>
<i>tuma</i>	pronoun	<i>tumhārā, tumhare, tumhari tumase, tumane, tumako, etc.</i>
<i>laraki</i>	noun	<i>larakiyā, larakiyō</i>

Table 4.1: Example of derivatives of some words

some additional information required for its recognition. This technique of storing only the root words consumes lesser space than what would be consumed if root words were stored along with all their derivatives in the dictionary.

4.2 What is Morphological Analysis?

The task of recognizing a root word from its derivative and retrieving the corresponding information from the lexicon is accomplished by the morphological analyzer. While a large number of derivatives follow systematic synthesis rules, there are exceptions which can not be obtained through morphological analysis. For example, 'went' is the past form of 'go', but it is not derivable from 'go'.

The morphological analyzer must know the basic words of the language in addition to the prefixes and suffixes to avoid incorrect breaking up of the words. In addition, the analysis must be guided by more specific constraints, since every word can not be combined with every suffix or prefix.

In our context, the morphological analyzer takes as its input a word (for convenience, a sequence of characters delimited by spaces or punctuation marks is defined

as word) or a joined_word (joined word is a concatenated word of two or more words), analyzes it, and tries to guess the root word. Then the root word is searched in the dictionary. If found, it retrieves all the relevant information of the root word such as syntactic category, number, person, gender, target language meaning and semantic tag etc. In the case of failure, further analysis is performed. If, even after exhaustive analysis, the root word is not found, there are two alternatives. The word can be assumed to be a proper noun or alternatively, the system may prompt for user intervention.

Words can be of two types, simple and compound words. A simple word is a root or a derivative of the root. A compound word is a combination of two or more words. For example, in Hindi '*kala purjā*' is a compound word, which is the combination of the words '*kala*' and '*purjā*'. Similarly, the word '*mujhako*' is the combination of the words '*mujha*' and '*ko*'. Therefore, in order to analyze the compound words, we need to concatenate the adjacent words as in '*kala purjā*', whereas for some other compound words like '*mujhako*' we need to break it into its constituents.

4.3 Difficulties Encountered in Morphological Analysis

The internal structure of words is complex, and needs to be examined in great detail within the paradigm of structural linguistics. Irregularity of the language and the historicity are two major factors that make the design of a morphological analyzer difficult [Ter83].

Irregularity : It becomes difficult to find out a minimal set of rules to capture the variations of the words. This is because they exhibit a high degree of irregularity

and idiosyncrasy. That is why, another lexicon is needed for such words called an exceptional lexicon.

Historicity : Much of the irregularity comes from the fact that lexical items come into a language relatively freely from other languages, and their normal form reflects the morphological structure of the source language as well as adoption to the new host. For example : the word “destroy” has the corresponding noun form “destruction”, but the word “employ” does not have its noun form as “empluction”. The latin roots “indices”, “schemata”, “cherubium” are all plurals that follow the rules of another language.

Different styles of writing : Some words can be written in different styles. For example: ‘cannot’ can be written as ‘can not’, ‘can’t’, ‘cann’t’. This too makes the morphological analysis difficult.

4.4 Strategies for Implementing a Morphological Analyzer

Broadly, there can be two strategies for implementing a morphological analyzer. One strategy is to have paradigm files for each part of speech with their characteristics. Prefixes are extracted and mapped to these paradigms. The other strategy is to have suffix files for each part of speech with information about the category. For example, the morphological analyzer can deduce ‘wife’ from ‘wives’, ‘nation’ from ‘national’ or ‘nationality’, ‘catch’ from ‘caught’. But the suffix file depends upon the application where the morphological analysis is being used. It may not be economical to analyze all kinds of derivatives. For example, suffixes ‘al’ and ‘ality’ are not very frequent, and so it may be more expensive to analyze such derivatives

than to store them as they are. But plural forms of nouns and derivatives of verbs are commonly found, and it really saves lots of space while creating the lexical database. Therefore, the implementation strategy and the kinds of suffix files needed, depend upon the application of the morphological analysis. There is a trade-off between the reduction in the the size of lexical database and morphological processing required to arrive at the root word. In fact, it is the application to which this lexical database is being used, which dictates the extent of the compromise. Further, if the lexical database is used in a bilingual or multilingual environment, it is important that the morphological synthesis be possible at the target level in a similar fashion.

4.5 Morphological Analyzer for Hindi

4.5.1 Analysis of the Verb-Part

In all languages, verb phrase has a specific place in the sentence and plays an important role to correctly understand the sentence. Hindi is a verb final language, where verb phrases have exocentric close-knit construction. In Hindi, verb phrases are combinations of the main verb and the auxiliary verb. On the contrary, English verb phrases are combinations of auxiliary verb and the main verb. Broadly, in Hindi, the main verbs can be classified as simple verbs, complex verbs, and the compound verbs [Sin85]. A complex verb is a combination of a noun and a simple verb or an adjective and a simple verb.

Complex verb = noun + 'kara/ho/de',

or

Complex verb = adj + 'kara/ho',

For example, *intajāra karana*, *sajā de*, *bimāra ho* etc. are complex verbs.

A compound verb is a combination of two simple verbs.

Compound verb = simple verb + simple verb,

For example, '*a jānā*', '*le jānā*' etc. are compound verbs.

While reading a complex verb, the system will recognize the first word as a noun or adjective. However, in order to identify the correct part of speech, further analysis will be needed. Actually, in Hindi, verb analysis becomes quite complicated and, at times, ambiguous because of various types of verbs and modifications that are possible in verb phrases. After analyzing different kinds of verbs in different types of sentences, we found that many ambiguities can be resolved, and verb analysis can be made more systematic, if we analyze the verb in a Hindi sentence from the end of the sentence. An expectation driven routine reads the last word of the Hindi sentence and accordingly moves to possible paths to find the root verb. Best-first-search has been implemented to get the root verb. A suffix file to get the root word from the past form of the verb has been used. For example : *pī* from *piyā*, *khā* from *khāyā*, etc.

One exception-paradigm file is used to take into account the exception forms of some verbs like the past form '*gayā*' of the the root '*ja*'. Another paradigm file generates the possible information about the 'number' and 'gender' of the subject of the sentence, according to the different verb-endings. For example, if the verb-ending is "*thī*", then the expected number of the subject is 'singular' and expected gender is 'female'

Another file defines the mapping of the 'tense' of the main verb to the corresponding translation pattern in English. For example, if 'tense' is 'prpa' (present perfect active) then,

translation_pattern = has/have + past_participle_form of the root verb.

Thus, the verb-part synthesizer module finds the main root verb of the input sentence, along with information about the 'number' and 'gender' of the subject of the sentence, the 'tense' of the verb and the corresponding verb form in target language.¹

4.5.2 Analysis of the Remaining Part of the Sentence

After identifying the main verb of a given input sentence and associated sentence properties dependent upon the verb, the rest of the sentence is passed to another sub-module which identifies the root words of noun, adjective and pronoun. Different suffix files are used for different parts of speech.

For example, if the word is

'larakiyō'

then the root word is extracted as follows:

1. Remove *'iyō'* from the word *'larakiyō'*.
2. Add *'ī'* to it from the suffix file.
3. Search for *'larakī'* in the electronic lexical database.
4. If found, retrieve all related information, and also pass the extra information of *'larakī'* being 'plural' in the input sentence.

In this way, at the time of translation, *'larakiyō'* is translated as 'girls'.

4.5.3 Analysis of a Hindi Sentence

Following steps are used in the Morphological Analysis of a Hindi sentence:

1. Read the input Hindi sentence S.

¹For convenience, it has been assumed that there are five forms of verbs. If verb is 'eat' then the five forms are 'eat', 'eats', 'ate', 'eaten', 'eating'.

2. Identify the words (sequence of characters separated by delimiters)

$w_1, w_2, \dots, w_{n-1}, w_n$ as an array W of words.

3. Call to verb-part synthesizer module

(analyzes the verb-part of the sentence)

Input : word array W

Return value: remaining number of words m .

(let there be m words left in the word array)

4. $i = 1$

while($i \leq m$)

{

join three words (w_i, w_{i+1}, w_{i+2}) to make one `joined_word`

call to noun-part synthesizer

(analyzes the plural form of nouns and adjective)

(Input : word, Return value : found flag)

if (found-flag = TRUE)

 modify i and return to while loop

else

{

join two words (w_i, w_{i+1}) to make `joined_word`

call to noun-part synthesizer with `joined_word`

if (found-flag = TRUE)

 modify i and return to while loop

else

{

call to noun-part synthesizer with single word w_i

```

if (found-flag = TRUE)
    modify i and return to while loop
else
    {
    break the word  $w_i$  into two words as
     $w_i =$  reduced word + post-position (using post-position file)
    call to noun-part synthesizer
    if (found-flag = TRUE)
        modify i and return to while loop
    else
        enter the word  $w_i$  as a proper noun in on-line-lexicon
        modify i and return to while loop
    }
}
}

```

Example: Let the input to morphological analyzer be the Hindi sentence :

H! *rama k̄ara ke kala purje th̄ika karane ke lie mujhase s̄o rūpae le raha*
he.

The analysis will be carried out using the above procedure as follows:

1. Read the sentence 'H!'
2. Words *rama, k̄ara, ..., le, raha, he* are identified as $w_1, w_2, \dots, w_{13}, w_{14}, w_{15}$.
3. verb-part synthesizer takes the array W as its input, and returns the following information:

3a. Words from w_1 to w_{12} .

3b. Root-verb : 'le', Tense : 'prca'(present continuous active).

3c. English meaning of 'le', number of the subject : 'singular', and gender : 'masculine'.

4. Remaining-part synthesizer takes the remaining word array as input, joins the words and calls noun-part synthesizer.

The word *rama* is identified as 'proper noun' because '*rama*' is not in lexicon.

To identify '*kala purje*', both words are joined and then from the noun suffix file 'e' is changed to 'ā', and '*kala purjā*' is identified as the root word. Similarly, '*thika karane*' is analyzed using the verb suffix file. However, the word '*mujhase*' is broken into '*mujha + se*', where '*mujha*' is a pronoun and '*se*' is a post-position, and two different entries are made in the on-line lexicon file.

4.5.4 Verb-part Synthesizer

The verb-part synthesizer uses the following steps for the example sentence:

1. Matching of the last word *hē* with the auxiliary verb suffixes.

2. A match is found, and the type of the suffix indicates that sentence is in present tense.

3. Now the next word w_{14} (*rahā*) is matched, and it generates a possibility of being the constant part of the present continuous tense.²

4. Now, if '*rahā hē*' is recognized as the auxiliary part of present continuous tense, expectation is generated for the word w_{12} as root verb.

5. But since the root verb can be a complex verb or compound verb, therefore at first, words w_{11} , w_{12} , w_{13} are combined and checked in verb lexical database. Since

²The word '*rahū*' is past form of the root verb '*rahu*', so there exists another path also if not successful in the first path.

these are not found there, only w_{12} , w_{13} are joined and again checked. Since these are again not found, only w_{13} , i.e., 'le' is searched in verb lexical database. 'le' is found in the verb lexical database, therefore all related information is retrieved. The tense is identified as 'present continuous tense', gender as 'masculine', and number as 'singular'.

6. The root verb 'le' is found this way, and the count of remaining words is passed to remaining-part-synthesizer module.

4.5.5 Noun-part Synthesizer

The noun-part synthesizer uses the following steps for the example sentence:

1. The words w_1 , w_2 , w_3 are joined and search_module is called. Search_module searches the word in lexical database.
2. The joined_word 'rāma kala purje' is not found, so analyzer_module is called. Analyzer_module uses different suffix files to analyze the word.³
3. Since the joined word is not found, therefore, words w_1 , w_2 are joined and again the joined_word 'rāma kala' is searched and analyzed.
4. Since the joined word is not found, therefore, the word *rāma* is searched and analyzed.
5. Since the joined word is not found, therefore, an attempt is made to break the word using a post-position file.
6. Since the word can not be broken, the word *rama* is identified as a proper noun.
7. Now the words w_2 , w_3 , w_4 are joined, searched and analyzed.

³Whole lexical database has been divided into smaller files like noun database, verb database and pronoun database. And, to minimize the searching time searching of root word is done in the following order of database i.e. Noun database, pronoun database, post-position file, and verb database.

8. Since the joined word is not found, therefore, the words w_2 and w_3 are joined and searched and analyzed. Analyzer_module changes the last suffix 'e' of '*kala purje*' to 'a', and '*kala purja*' is identified as a root word, and all needed information is retrieved. Note that the words, '*kala*' and '*purja*', separately also, are two different valid words.

9. Similarly, '*thika karane*' is analyzed using verb suffix file. But the word *mujhase* is broken into '*mujha + se*', where '*mujha*' is a pronoun and '*se*' is a post-position, and correspondingly two different entries are done. Similar process continues till all the remaining words are exhausted.

Most Indian languages are verb final languages, and the implementation strategy used in Hindi morphological analyzer can be used for all those languages. Only suffix files will be changed according to the source language.

Chapter 5

Creation and Arrangement of Example-Base

An example-base for an EBMT system, consists of examples of translation pairs. A translation pair consists of a source language example, its translation, and a segment mapping which defines segment level correspondence between the source language sentence and the target language sentence [H.K93a]. The examples for the example-base are taken from various sources, e.g., government departments, publishers etc. However, one major problem in creating an example-base is the selection of examples for the example-base. On one hand, we want the example-base to contain enough examples so that for every input sentence to be translated, a suitable match can be found in the example-base. On the other hand, we want the size of example-base to be as small as possible, to reduce the space required to store the example-base, as well as to reduce the time needed to search the suitable example. In other words, we want the example-base to contain only those examples which are really used for translation, without repeating the examples which are similar in type and structure to other examples in the example-base. In our implementation of the

Hybrid Example-based Approach, we have tried to fulfil these requirements by using the concept of dynamic creation of example-base.

5.1 Concept of Dynamic Creation of Example-Base

A developer of EBMT, will have a number of sentences for which translation from source to target is available. These examples themselves can form the example-base for the EBMT system. However, all of these examples may not be really required to be kept in the system as they may be similar to each other and the system may be able to derive the translation from other examples. Therefore, a dynamic approach is more appropriate for the growth of the example-base. Using this approach, we assume that the system developer starts with an empty example-base. First example sentence is entered as it is. After that, whenever a new sentence arrives for translation, it is first matched with examples in the example-base. If the system fails to give a satisfactory translation, it is assumed that this input sentence is different from the examples in the example-base. In that case, the input sentence is added to the example-base leading to the growth of the example-base. This way, it is possible to contain the size of the example-base to the minimum possible. Obviously, this approach is very much dependent on the way the similarity has been defined. Figure 5.1 depicts the algorithm used for creation of abstracted example-base.

5.2 Creation of Abstracted Example-Base

In HEBMT system, we have used an approach of creating the example-base which is different from other example-based approaches. The HEBMT system example-base

```

i=1;
get_from_corpus(SLES[i]);
do_morph_analysis(SLES[i]);
get_syntactic_groups_of(SLES[i]);
get_translation_of_syntactic_groups;
get_from_corpus(TLES[i]);
enter_target_pattern; \* Developer enters the target pattern
                        looking at the TLES[i] and the
                        translation of syntactic units*\
add_to_example-base(TLES_pattern); \*SLES pattern and syntactic
                                    and semantic information
                                    will be automatically added*\

while(not end of corpus)
{ i=i+1;
  get_from_corpus(SLES[i]);
  do_morph_analysis(SLES[i]);
  get_syntactic_group_of(SLES[i])
  j=1;
  while(j<i or flag ==0)
    {match_verb_type_of(SLES[i],SLES[j]);
     if MATCHED
       {compare_no_of_syntactic_units(SLES[i],SLES[j]);
        if EQUAL
          {match_syntactic_unit_types(SLES[i],SLES[j]);
           if MATCHED
             {get_translation_of_(SLES[i]);
              (using target pattern of SLES[j])
              ask_developer's_satisfaction;
              if SATISFIED flag =1;
             }
          }
        }
     }
    j = j +1;
  }
  if(flag ==0) add_to_example-base(TLES[i]_pattern);
}

```

Where

SLES[i] -- ith source language example sentence
TLES[i] -- ith target language example sentence

Figure 5.1: Algorithm to create example-base

consists of abstracted examples, instead of raw examples. Initially, one example Hindi sentence is entered for translation, and the finite state machine transforms the Hindi sentence into an ordered set of syntactic groups (refer chapter 6). Now the pattern of the abstracted Hindi sentence is searched in the abstracted example-base. If a match is found, the system translates it into English according to the abstracted English example. If a match is not found or if the translation is not upto the satisfaction of the developer, the system requests the developer to enter the target language pattern. The abstracted Hindi sentence along with the target language pattern given by the developer, is then entered in the example-base. Note that the abstracted English example will contain the syntactic groups and some other extra features of English language, if needed. As an example, in the following, 1(a) to 1(e) show the process of abstraction for the Hindi sentence H1!.¹

H1!.	<i>sudhira gitā ko pyara karatā hē.</i> {Input Hindi Example Sentence }.		
1(a).	<i>(sudhira)</i>	<i>(gitā ko)</i>	<i>(pyara karatā hē)</i> {Syntactic grouping in Hindi Sentence. }
1(b).ES!	<snp,1>	<snpk2,2>	<mv> {abstracted example in Hindi(SL)} ²
1(c).	<i>(Sudhir)</i>	<i>(Gita)</i>	<i>(loves)</i> {English translation of syntactic groups.}
1(d).ES!	<snp,1>	<mv>	<snpk2,2> {Abstracted example of English}
1(e).	Sudhir	loves	Gita. {Translation after substitution.}

¹Only those parts, which are marked with ES!, are actually entered in the example-base.

Similarly, 2(b) to 2(e) show the abstraction corresponding to the Hindi example sentence H2!.

H2!.	<i>jona ko bukhara he.</i> {Input Hindi Example Sentence }.		
2(a).	<i>(jōna ko)</i>	<i>(bukhara)</i>	<i>(he)</i> {Syntactic grouping in Hindi Sentence. }
2(b).ES!	<snpk2,1>	<snpk,2>	<av> {Abstracted example in Hindi.}
2(c).	<i>(John)</i>	<i>(fever)</i>	<i>(is)</i> {English translation of syntactic groups.}
2(d).ES!	<snpk2,1>	{has/have}	<snpk,2> {Abstracted example in English.}
2(e).	John	has/have	fever. {Translation after substitution.}

Now, the following Hindi sentence H3! will be translated using the above example-base following the steps 3(a) to 3(d).

H3!.	<i>jōna kā barā bhāī merī chotī bahana ko maratā he.</i> {Input Hindi Sentence}		
3(a).	<i>(jōna kā barā bhāī)</i>	<i>(merī chotī bahana ko)</i>	<i>(maratā he).</i> {Obtained after syntactic grouping}
3(b).	<i>(John's elder brother)</i>	<i>(my younger sister)</i>	<i>(beats)</i> {English meanings of syntactic units.}
3(c).	<snpk,1>	<snpk2,2>	<mv> {Abstracted input Hindi Sentence}

This input Hindi sentence will be matched with the first example of the example-base i.e. with 1(b). Then input sentence H3! will be translated using the target pattern 1(d) as follows:

3(d). *John's elder brother beats my younger sister.*

Consider another example of input Hindi sentence H!4,

H!4. *mere dostā ko bahuta teja bukhāra hē.* {Hindi Input Sentence}

- | | | | |
|-------|------------------------|------------------------------|-------------------------------|
| 4(a). | <i>(mere dostā ko)</i> | <i>(bahuta teja bukhāra)</i> | <i>(he)</i> |
| 4(b). | <snpk2,1> | <snpk,2> | <av> |
| | | | {abstracted example in Hindi} |
| 4(c). | <i>(my friend)</i> | <i>(very high fever)</i> | <i>(is)</i> |

This input Hindi sentence will be matched with the first example of the Example-Base, i.e., with 2(b). Then input sentence H4! will be translated using the target pattern 2(d) as follows:

4(d). *My friend has/have very high fever.*

But, when the input sentence "*sītā somavāra ko jā rahi hē*" is fed for translation, it is matched with the first example H!1 of Hindi and the system translates it as "*Sita is going Monday*", whereas an acceptable translation would be "*Mary is going on Monday*". In this case, we enter another example where the English example will be different from the first one and will be of the form:

<snpk,1> <mv> {on} <snpk2,2>

Now when a new input sentence of similar kind comes, it is matched with both the examples but due to syntactic and semantic differences, distances are different and example with minimum distance is invoked for translation. After an exposure

to a variety of sentences, the example-base attains a level of maturity, wherein the additions are not frequent.

5.3 Partitioning of the Example-Base

In order to reduce the search time, the example-base is partitioned on the basis of type of main verb of the sentence and on the number of fragments in the source language sentence.

Verb plays an important role in any sentence translation and in sentence parsing. Two sentences may look similar if their verb parts are ignored, but if verb parts are taken into account, sentences become entirely different. For example:

H5!. *vaha ladakā hē.* {Hindi Sentence}

E5!. He is a boy. {English Translation}

H6!. *vaha ladakā jā raha hē.*

E6!. That boy is going. {English Translation}

In the above sentences, if the verb parts are taken out, the sentences look exactly similar. In the above sentences, grouping of the words will be different due to different verb-types, and so the translation will also be different.

Let us take another example, where the translation is different even though the grouping of the words is similar.

H!7. *vaha mera Gara hē.* {Hindi Sentence}

E!7. That is my house. {English Translation}

H!8. *vaha mere Gara ja raha hē.* {Hindi Sentence}

E!8. He is going to my house. {English Translation}

Therefore, we can see that it is very important to distinguish between the

sentences on the basis of verb-type. Verb-type has been used as a primary key to partition the example-base.

Sentences containing different number of syntactic units differ in their translation patterns also. That is why, number of fragments has been used as a secondary key to partition the example-base.

Given, abstracted examples of the following six types are put in three different partitions.

1. (snp) (snp) (mv)
2. (snp) (snp) (mv)
3. (snp) (snp) (av)
4. (snp) (mv)
5. (snp) (snpk2) (mv)
6. (snp) (sadv) (av)

Examples 1, 2 and 5 are put in partition 1, examples 3 and 6 are put in partition 2, and example 4 is put in partition 3. It should be noted that examples 1 and 2 have the same SL syntactic pattern but their syntactic and semantic information and target pattern will be different. All the nine "snp" in six examples may be entirely different as far as their STATUS (animate, inanimate), GNP(gender, number, person), SEMANTIC-TAG are concerned. Some sample entries from our example-base are shown in Appendix F.

Chapter 6

Identification of Syntactic Units in a Sentence

As discussed in earlier chapters, the example sentences are stored in the example-base in an abstracted form as a sequence of syntactic units. Similarly, the input sentence has also to be analyzed for identifying syntactic units in it. We use a finite state machine, which we call as HFSM (Hindi Finite State Machine) [JRJS95] for identifying the bindings between the words of a Hindi sentence, and for replacing them by groups. These groups act as syntactic units in the sentence. The identification of the syntactic units is the first step in the direction of understanding any language and for translation from that language to any other language. In this chapter, our discussion will be mainly focussed on the finite state machine for Hindi language only. However, most of the discussion can be extended to other languages also.

Although Hindi is said to be a relatively free word order language, it is primarily true only for the syntactic units and not for individual words. The analyzer joins those words of the sentence which can not move independently in the sentence but,

as a group can move around in the sentence, to a large extent, without affecting its interpretation. The lexical category and the postpositions provide the essential clues for the identification of syntactic units.

[GD93] have presented the issues involved in the design of a computational base grammar for Hindi sentences. It has been shown that operation of grammar rules depends on the priority of their applicability and grammar rules have been augmented using LFG (Lexical Function Grammar). [RC93] have designed an interface to correct certain kinds of errors in Hindi sentences. [OSV93] have designed a Hindi sentence analyzer using semantic tag based object oriented design. [S.S87] has presented the features of equational sentences.

We have developed a Hindi sentence analyzer in the form of a finite state machine. We are using only broad lexical categorization of words and very little semantic information to analyze a wide variety of Hindi sentences. The finite state machine offers a convenient instrument for dealing with a variety of sentences which may be encountered during the process of development of a system.

In order to obtain the translation of a sentence, the syntactic units have to be translated in the target language after their identification. It is possible to have a separate target language text generator to translate the syntactic units. In that case, each syntactic unit has to be synthesized and then translated. Here, we have clubbed the synthesis part with the analysis part to increase the efficiency of the translation system.

6.1 Main Features of Hindi Language relevant to the Identification of Syntactic Units

Hindi language has S.O.V (Subject-Object-Verb) sentence structure. A Hindi sentence can be broken into two parts as follows.

Main-Part + Verb-Part,

where Main-Part is a combination of noun phrases, adjective phrases, adverb phrases etc., and Verb-Part is the combination of main-verb and auxiliary verb. In Hindi, the main verbs can be simple, complex, compound or of some special type. A large number of verbs are derivatives of nouns and adjectives, which appear in conjunction with auxiliaries. These nouns and adjectives constitute part of the verb, therefore for the analysis of the sentence they are not treated as nouns or adjectives.

For Example¹:

H1. *vaha garibō ki sevā karatā hē.*

E1. *He serves the poor.*

In the sentence H1, the word 'sevā' will have a lexical entry as a "noun" with English meaning "service". However, the auxiliary part 'kara' following 'sevā' makes 'sevā' to be interpreted as a 'verb' whose meaning in English is 'serve'.

Even the auxiliary portion of the Verb-part has many variations depending upon the gender, number and person of the subject in the sentence. According to the Indian traditional Paninian approach, the verb plays a central role in the analysis of a sentence. All "karak" relationships depend upon the verb. The verb generates expectation for the "karaks". The post-positions ("parsarg" or "Vibhakti" sign) provide clues to the plausible "karak" relationships.

¹All the sentences numbered with "H" are Hindi sentences and all sentences numbered with "E" are English sentences.

SN	Karak	Case Marker(Post-position)
1.	<i>karata</i> (Subject)	<i>ne</i>
2.	<i>karma</i> (Object or Destination)	<i>ko</i>
3.	<i>karana</i> (Instrument)	<i>se</i>
4.	<i>sampradana</i> (Purpose or Effect)	<i>ke lie</i>
5.	<i>apādana</i> (Source)	<i>se</i>
6.	<i>adhikarana</i> (Location or Destination)	<i>mē, para</i>

Table 6.1: Karaks and Post-Positions

According to Indian traditional Paninian frame work, Hindi has six karak relationships and a number of nonkarak relationships. These karak relationships are akin to cases used by the western grammarians. In Hindi, there are a number of post positions called “parsarg” or “Vibhakti sign”, which give clues about these karaks and non-karaks relationships. These signs are not unique and only generate expectation for analysis.

The Table 6.1 enumerates the six karaks and the corresponding post-positions used and Table 6.2 enumerates the post-positions and the corresponding karaks and non-karaks.

6.2 HFSM : A Finite State Machine for Identifying Syntactic Units in Hindi Sentences

HFSM takes as its input, the output of the morphological analyzer (refer chapter 4). Morphological analyzer provides all possible syntactic categories, all possible meanings, and other necessary syntactic and semantic information about every word from the input sentence. HFSM is started by reading the first syntactic category of the first word, and it generates the expectations for the syntactic categories of

SN	Post-positions	Karak or Non-karak relation
1.	<i>ne</i>	<i>karatā</i> (Subject)
2.	<i>ko</i>	<i>karama</i> (Object)
3.	<i>se</i>	<i>karana</i> or <i>apadana</i> (Instrument or Source)
4.	<i>mē</i>	<i>adhikarana</i> (Location or Destination)
5.	<i>para</i>	<i>adhikarana</i> (Location or Destination)
6.	<i>kā, ki, ke</i>	Non-karak (Showing relation)
7.	<i>ke bada</i>	Non-karak (Showing time)
8.	<i>ke pasa</i>	Non-karak (Showing place)
9.	<i>ki ora</i>	Non-karak (Showing direction)

Table 6 2: Post-Positions and Karaks / Non-Karaks

the adjacent word. For example, if the first symbol is a 'possessive-case' (like *merā*, *mere*, *usakā*, etc.), HFSM will generate the following six expectations:

'common noun',

'non-karak' (*he*, *bāda*, etc),

'verb-noun' (*girana*, *lene*, etc),

'adjective',

'adverb',

and

'end of sentence',

since words (or phrases) belonging to any of these six categories can follow a possessive-case. If the next word is not one of the generated expectations, HFSM tries to find out if the original word has any other syntactic category also. If the word has any other category also, the HFSM tries out expectations according to this new syntactic category. Otherwise, the HFSM halts and concludes that the input sentence is syntactically incorrect. This process continues till the proper syntactic

unit is identified. After the identification of syntactic unit, the finite state machine writes down its type, its target language meaning, and other syntactic and semantic information, and then continues processing the rest of the sentence.

6.2.1 Illustrations

In this subsection, we will examine few entries from the Hindi-English example-base to illustrate the working of the HFSM.

Example1:

Let the entry be:

H2. *jōna ne cora ko pakara liyā hē.*

E2. *John has caught the thief.*

In the sentence H2, the verb-part is '*pakara liyā hē*', and the remaining part is '*jōna ne cora ko*'. The verb-part is analyzed by verb-part synthesizer (refer Chapter 4). In the remaining part, '*jōna*' is recognized as 'noun' and '*ne*' as a "Vibhakti" sign (case marker) (refer Table 6.2) of type 1. Therefore, '*jōna ne*' becomes one syntactic unit, and it is named "nPK1" (please refer Appendix B). 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 '*cora*' which is 'noun', and is followed by '*ko*', which is a case marker of type 2. Therefore, '*cora ko*' is identified as "nPK2". Thus the final pattern of H2 is:

<nPK1> <nPK2> <mv>

Here "mv" denotes the main verb.

Example 2:

Let us examine the entry:

H3. *jōna ke bare bhāī ne merī ke sabase acche dostā ko mārā thā.*

E3. *John's elder brother had hit Mary's best friend.*

In the sentence H3, 'māra thā' is recognized as main verb, 'jōna ke bare bhai ne' is recognized as "npk1", because non-karak 'ke' follows the noun symbol 'jōna', 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, 'mēri ke sabase acche dostā ko' is identified as "npk2".

Note that, after the identification of syntactic groups, sentences H2 and H3 look exactly the same at the surface level. It also shows that 'jōna' can be mapped to 'jōna ke bare bhāi' and 'cora' can be mapped to 'mēri ke sabase acche dostā'.

Example 3:

Let the entry be:

H4. *saba bacce gurū se dharmā kī shīkshā le rahe hē.*

E4. *All the children are learning about religion from the teacher.*

In sentence H4, 'shīkshā le rahe hē' is identified as the verb-part, and in the remaining part, 'saba bacce' as <np>, 'gurū se' as <npk3> and 'dharmā kī' as <npk6>.

Therefore, the final pattern of the sentence H4 looks like this:

< snp > < npk3 > < npk6 > < mv >

In Table 6.3, some Hindi sentences along with their English translation have been listed, and Table 6.4 shows the corresponding syntactic units identified by the HFSM. Please note that the English sentences have been given just for illustration, and will not be stored in the actual example-base.

When the above Hindi sentences were fed as input to HFSM, the outputs obtained in the form of syntactic units are shown in Table 6.4.

SN	Hindi Sentences	Corresponding English Sentences
1.	<i>vaha larakā he</i>	He is a boy
2.	<i>vaha larakā jā rahā he</i>	That boy is going
3.	<i>jōna mēri ke ghara ke pāsa rahā karatā thā</i>	John used to live near Mary's house
4.	<i>ve larakē mere ghara rote hue āe the</i>	Those boys came to my house crying
5.	<i>jōna kī bahana kī shadi mere cacere bhāī ke sātha kanapura me ho rahi hē</i>	John's sister's marriage is being performed in Kanpur with my cousin brother

Table 6.3: Input Hindi sentences

SN	Output of HFSM
1.	<snp> <snp> <av>
2.	<snp> <mv>
3.	<snp> <npk7ank> <mv>
4.	<snp> <snp> <vp_adv> <mv>
5.	<snp> <npk7nk> <npk4> <mv>

Table 6.4: Identified Syntactic Units of Hindi sentences

6.2.2 State Transition Graphs of the HFSM

Figure 6.1 and Figure 6.2 show the state transition graph (STG) of finite state machine for two different Hindi sentences H5 and H6.

H5. *jōna ke bare bhāī kā betā mere dostā ke larake ko mara raha he.*

E5. *John's elder brother's son is hitting my friend's son.*

H6. *kala jōna mēri ke satha dillī ghumane jāyegā.*

E6. *Tomorrow John will go to see Delhi with Mary.*

6.3 Translation of Syntactic Units in Target Language

As mentioned above, the HFSM also produces the English translation corresponding to the identified syntactic units. This eliminates the need of a separate 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 following words and the features of verb-part are used to identify the correct syntactic unit and the correct meaning of that syntactic unit. Semantic knowledge of the constituent words of the syntactic category is used to pick up the most suitable meaning of those words.

In Hindi, most of the pronouns and possessive-case 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 jā rahā hē.

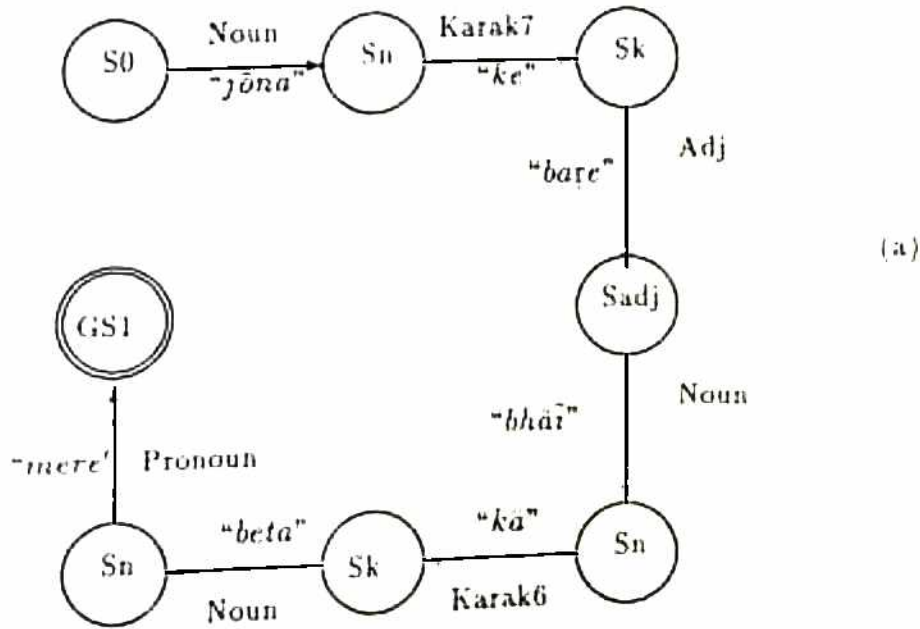
He is going.

vaha pera hē.

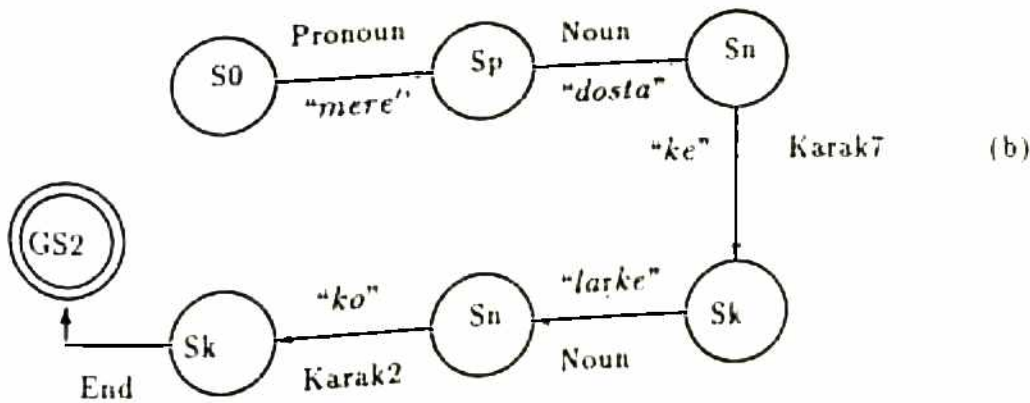
That is a tree.

Input Sentence :

"jōna ke baṛe bhāī kā beta mere dostā ke laṛke ko mara raha he"



"jōna ke baṛe bhāī kā beṭā" is Identified as < snp >



"mere dostā ke laṛke ko" is Identified as < npk2 >

THE FINAL PATTERN IS < snp > < npk2 >

Figure 6.1 State Transition Diagram for the sentence H5

Input Sentence :

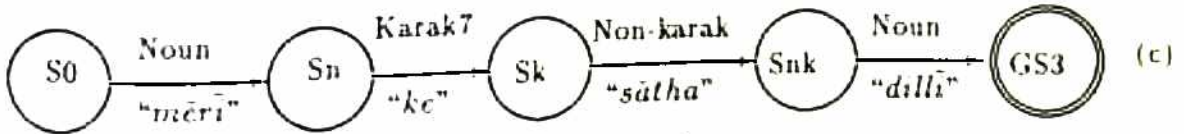
"kala jona meri ke satha dilli ghumane jayega."



"kala" is identified as < sadv >



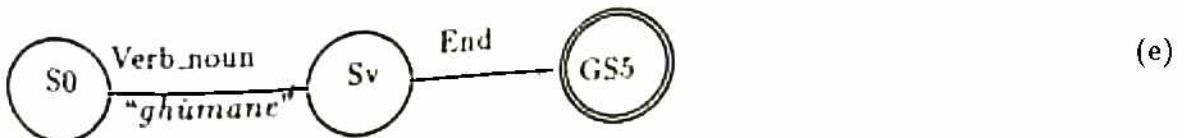
"jona" is identified as < np >



"meri ke satha" is identified as < npk7nk >



"dilli" is identified as < np >



"ghumane" is identified as < vnp >

FINAL PATTERN IS < sadv >< np >< npk7nk >< np >< vnp >

Figure 6.2: State Transition Diagram for the sentence H6

<i>vaha ladra ka ja rahā he.</i>	That boy is going.
<i>vaha kānapura ja rahā he.</i>	He is going to Kanpur.
<i>vaha khana khā rahi hē.</i>	She is eating food.
<i>vaha peṛa lambā he.</i>	That tree is long.
<i>vaha mera dostā he</i>	He is my friend.
<i>vaha cora bhaga gaya</i>	That thief ran away.
<i>vaha cora lagatā hē</i>	He looks like a thief.
<i>vaha cora kā bhāī hē</i>	He is thief's brother.
<i>vaha mere sātha rahatī he.</i>	She lives with me.

Similarly, the various roles of possessive-case (*mere, merī*) are illustrated in the following examples.

<i>mere pāsa cāra kitābē hē</i>	I have four books.
<i>ye kitābē merī hē</i>	These books are mine.
<i>pitāji mere lie kitābē lāye hē</i>	Father has brought books for me.
<i>mere pitāji adhyāpaka hē</i>	My father is a teacher.

From the above Hindi sentences, it is evident that knowledge of sentence structure and of the characteristics of verb-part, are necessary to disambiguate the correct role and correct meaning of most of the pronouns and possessive-cases. The HFSM has been designed to take care of such contextual knowledge. The finite state machine is 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.

Chapter 7

Example Matching and Target Language Sentence Generation

For matching examples of source language, the system uses the type of main-verb, the number of syntactic units, the type and ordering of each syntactic unit, and the associated syntactic and semantic tags. While the syntactic tags are well understood and clearly defined, it is not so for the semantic tags. The success of finding the best match greatly depends upon the way these tags are assigned and classified, and their corresponding hierarchy. In the following section, we examine the aspects concerning the semantic tags. In section 7.2, we present the layout of the lexical database and the disambiguation of multiple meanings using semantic tags. In section 7.3, we present the process of matching and the design of the distance function and its implications.

7.1 Semantic Tags

Most of the time, we can disambiguate words [J.A87] that are ambiguous between syntactic classes by syntactically analyzing the sentence that contains them. For

example, given the sentence 'The bottle dropped and broke', the word 'drop' has been used here as a verb, and it is clear from the sentence structure that it cannot be a noun. However, the word is still ambiguous between its meanings 'fall by accident', 'become less', 'give up', 'lose' etc. This ambiguity can be resolved only by semantic information. By combining knowledge about the case structure of verbs with selectional restrictions on what types of objects can fill those cases, the interpreter picks the 'fall by accident' sense.

All words sharing some common features of meaning form a semantic field [Wei63] and [Sin93]. All words belonging to a semantic field are linked to each other through some semantic relationship. For example, all words denoting day (Monday, Tuesday, Sunday etc.) would form one semantic field. Similarly, words expressing some feeling will form another semantic field. Semantic field establishes a relationship between the linguistic and extra-linguistic components, and links the linguistic form with the world of experience. It has been agreed by semanticists that the meaning of a lexical item cannot be brought out precisely and effectively as long as its inter-relationship with its synonyms, antonyms or other related words belonging to the same semantic field is not known.

However, as the human mind is able to accumulate all semantic relationships without all explicit explanations, it becomes very difficult to list all relationships while developing a dictionary or any lexical database for NLP applications. 'Actually, it was found that it is not necessary to incorporate all semantic relationships while developing a dictionary to bring out the meaning of each word. Therefore classification of objects is done using the hierarchy of semantics.

Any list of semantic tags and the hierarchy cannot be stated as necessary and sufficient. It is very difficult to put a limit on the number of levels to which this

hierarchy can grow. The ultimate limit is the 'word' itself. Therefore, every system designer decides, based on experiences gained by analysis of the language pair, a certain set of semantic tags and their hierarchy which works in most of the cases.

A list of semantic tags (see Appendix A) and their hierarchy has been designed for the HEBMT system. It has been implemented using tree data structure. This semantic tag tree has been designed keeping in mind the application of machine translation. Semantic tag hierarchy has been shown in the form of a tree in Figure 7.1.

We can neither prove that it is sufficient or necessary, nor claim that it is perfect from a linguistic point of view. However, it has been tested for lots of input sentences in common usage and, to a large extent, has been found to be adequate.

7.2 Lexical Database used for HEBMT and Multiple Meaning Disambiguation

All the information used in evaluating the distance between the syntactic units of an input sentence and the example sentence, is stored with the words while developing the lexical database. Lexical database is a dictionary plus a knowledge base with grammatical, semantic and syntactic information. Some of the entries from the lexical database of HEBMT system have been listed in Appendix C. As can be seen from these entries, semantics help us in distinguishing between the words having same syntactic category.

In HEBMT system, semantic tags have been used for sense disambiguation and for distinguishing one example from another. Note that the words which belong to adjective category have been given noun tags. Actually, these noun tags help in

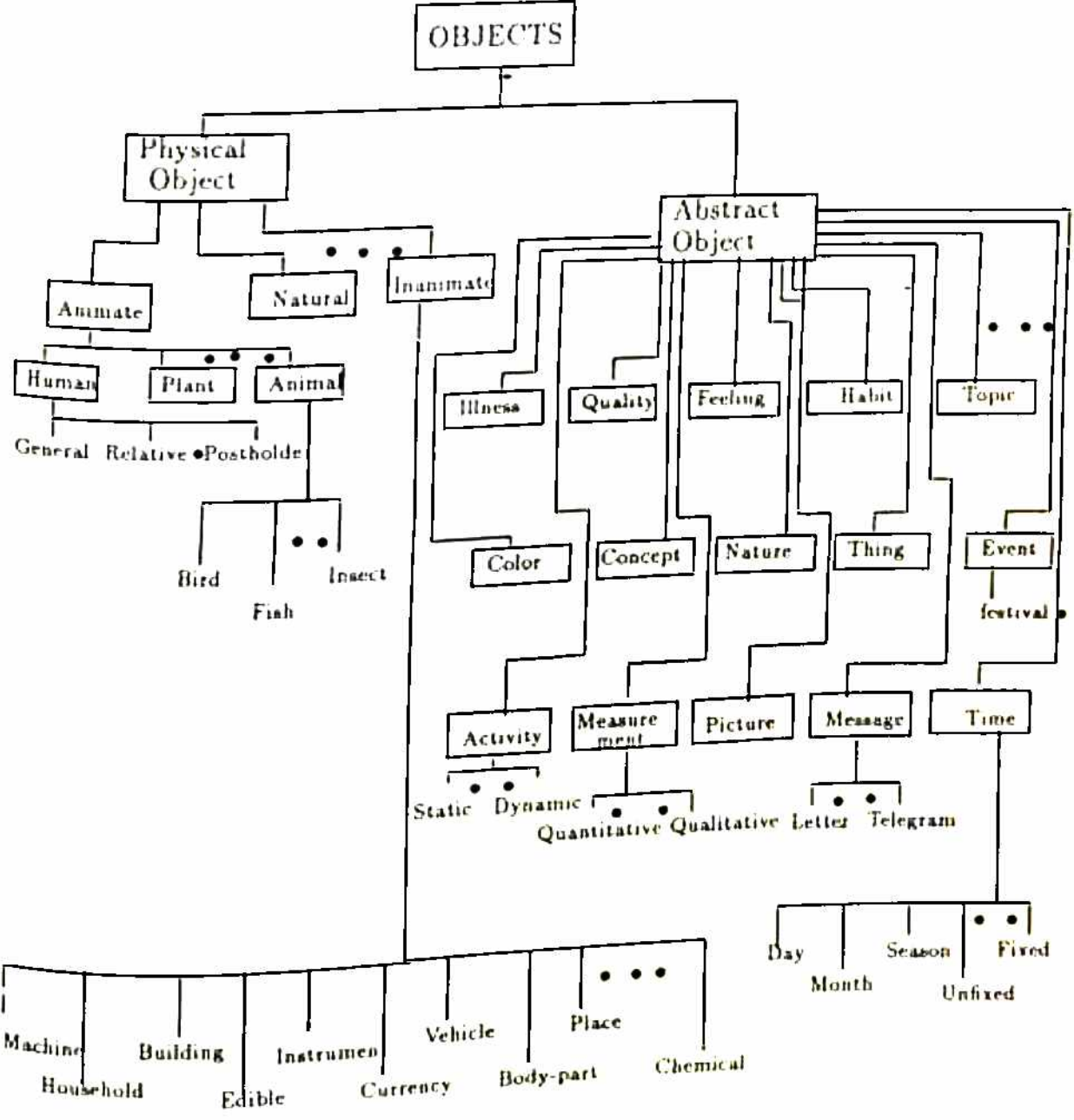


Figure 7.1: Semantic Tree

picking up the right meaning of adjectives and nouns. For example, the syntactic unit '*rama ka chotā larākā*' was translated by our system as 'Ram's younger son' with the help of the semantic tags (and not as 'Ram's small son').

It is well known that a complete disambiguation requires a thorough understanding of the language. However, here we have not attempted to create and store such an understanding in the machine. Our attempts are limited to discovering sentence surface structures which enable us to disambiguate meaning. An attempt at the disambiguation of verb meanings was made in this direction, and it was realized that it is better to have it in implicit form. During the analysis, it was found that for some verbs it was possible to assign noun tags according to verb requirements for meaning disambiguation. However, there were verbs for which the knowledge of semantic tags was not enough. Most approaches require a complete analysis of the sentence, by identifying subject and object. Here, we are not attempting to generate complete parse of the sentence. The only attempt is to match at the surface.

In HEBMT system, verb meanings have been resolved from the example-base itself. This approach seemingly increases the size of the example-base, but the developer does not have to provide knowledge in its explicit form, and it also saves a lot of human effort. The task of assigning the semantic tags for verbs requires expert linguistic knowledge, and needs a system developer who can represent the knowledge in the form of proper semantic tags. An expert has to examine all possible meanings a verb can have, and also has to help the developer to find the proper classes of nouns which will represent them. It is possible that the developer is unable to define unambiguous tags, even after an exhaustive analysis. Therefore, HEBMT system makes use of its example-base wherever a verb has multiple meanings. It provides the developer or user the facility to enter the meaning of his own choice. It

also helps the user in picking up the right meaning by providing the list of possible meanings.

Though we have tried to resolve multiple meanings of nouns and adjectives using semantic tags, there are still many cases where we can not resolve the meanings. In those cases, we have either given the option for post-editing or such a meaning has been chosen, which may not be the most appropriate meaning at that place, but it appropriately conveys the message.

7.3 Matching Process and Design of Distance Function

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. Then the finite state machine identifies the syntactic units using the retrieved information of words. The finite state machine also translates those syntactic units in the target language. The pattern of all syntactic units and other related information is passed to matching module. The example-matching module first looks at the verb-part of the input sentence and on the basis of the type of verb-part, chooses the appropriate partition for input sentence to match. Then the syntactic units of the input sentence are counted for entering into the next level of partition. After reaching the correct sub-partition, exact pattern matching is performed.

The matching process then retrieves all examples from the appropriate partition, and tries for matching with the input sentence, in the order and type of each syntactic group. For all such examples, distance is found with the input sentence using a

distance function. A simple computational measure of semantic closeness between two words is their distance from each other in the type hierarchy. The distance function to retrieve the best example for a given input sentence has an important role in determining the success of HEBMT or any example-based system. In our present implementation of the HEBMT, the distance 'd' between IS(input sentence) and ES(example sentence) is defined as follows:

$$d(IS, ES) = \sum_{p=1}^n d_p(ISG, ESG) + d_v(ISV, ESV)$$

where n is the number of noun syntactic groups in the source language sentence and ISG and ESG are 'input sentence noun syntactic group' and 'example sentence noun syntactic group' respectively, and ISV and ESV are Input and Example sentence verb groups.

$d_p(ISG, ESG)$ – distance between p th noun syntactic groups of input and example source language sentence

$d_v(ISV, ESV)$ – distance between verb group of input and example source language sentence

$$d_p(ISG, ESG) = (w_1 \times ATD + w_2 \times ASD)/(w_1 + w_2)$$

$$ATD = (q_1 \times SD + q_2 \times (GD + ND + PD))/(q_1 + q_2)$$

$$d_v(ISV, ESV) = VCD$$

where ATD, SD, GD, ND, PD, ASD and VCD are attribute difference, status difference, gender difference, number difference, person difference, additional semantic difference, and verb category difference between example sentence and input sentence. w_1, w_2, q_1, q_2 , are weights 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.

The values assigned to weights and different distance parameters are: $w_1 = w_2 = 1$, $q_1 = 2$, $q_2 = 0.5$

$$SD = \begin{cases} 0 & \text{if STATUS(animate, inanimate)} \\ & \text{in both the groups is same} \\ 0.5 & \text{otherwise} \end{cases}$$

$$GD = \begin{cases} 0 & \text{if GENDER(masculine, feminine, neuter)} \\ & \text{in both the groups is same} \\ 0.2 & \text{otherwise} \end{cases}$$

$$ND = \begin{cases} 0 & \text{if NUMBER(singular, plural)} \\ & \text{in both the groups is same} \\ 0.2 & \text{otherwise} \end{cases}$$

$$PD = \begin{cases} 0 & \text{if PERSON(first, second, third)} \\ & \text{in both the groups is same} \\ 0.2 & \text{otherwise} \end{cases}$$

$$ASD = \begin{cases} 0 & \text{if SEMANTIC.TAG(relation, profession, liquid, etc.)} \\ & \text{in both groups is same} \\ 0.3, 0.5, 1.0 & \text{depending upon the level of matching in semantic tree} \end{cases}$$

$$VCD = \begin{cases} 0 & \text{if verb class(transitive, intransitive, etc.)} \\ & \text{in both the groups is same} \\ 0.5 & \text{otherwise} \end{cases}$$

Above values are assigned based on examination and observation of a variety of source sentences. However one may attempt to design a system which learns these weights through examples. This is itself a complex task, and has not been addressed in this thesis. In this sense, values of the weights assigned are ad-hoc, but provide reasonably good matching, as can be seen from the translated outputs shown in Appendix D and Appendix G, and results presented in chapter 8.

7.4 Illustration of Distance Procedure

Procedure of distance evaluation has been explained by taking some example sentences and some input sentences. Example sentences have been denoted by 'E' 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:

E1. *jōna elisā ko paṛhāta he.*

E2. *merā dostā itavāra ko āyega.*

Both of these examples have source pattern as '<snp> <nkp2> <mv>'.
'

Target pattern corresponding to example E1 is '<snp> <mv> <nkp2>'.
'

Target pattern corresponding to example E2 is '<snp> <mv> {on} <nkp2>'.
'

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. *merī somavāra ko jā rahī he.*

I2. *sudhīra mere bhāī ko pīta rahā hē.*

I3. *merī kulhārī kīsī bhī padārtha ko kṛta sakatī he.*

I4. *merā choṭā bhāī mujhako pyāra karatā he.*

I5. *usakā bhāī itavāra ko vāpisa āyegā*

Table 7.1 shows the individual distances between corresponding syntactic units of sentences E1 and I1.

Therefore, the distance between I1 and E1.

Example Sentence Group	Input Sentence Group	Distance
<i>jōna</i> (snp,[ani,mas,sing,third],[human])	<i>meri</i> (snp,[ani,fem,sing,third],[human])	0.03
<i>elisā ko</i> (npk2,[ani,fem,sing,third],[human])	<i>somavara ko</i> (npk2,[inani,neu,sing,NA],[day])	0.73
<i>parhātā he</i> (mv,[transitive])	<i>jā rahi he</i> (mv,[intransitive])	0.5

Table 7.1: Distance Calculation

$$D(I1,E1) = 0.03 + 0.73 + 0.5 = 1.27;$$

Similarly, the distances were evaluated for other input sentences with both example sentences. These distances are listed below:

$$D(I1,E2) = 0.03 + 0.0 + 0.0 = 0.03;$$

$$D(I2,E1) = 0.25 + 0.28 + 0.0 = 0.53;$$

$$D(I2,E2) = 0.03 + 0.73 + 0.50 = 1.27;$$

$$D(I3,E1) = 0.73 + 0.73 + 0.0 = 1.47;$$

$$D(I3,E2) = 0.73 + 0.50 + 0.50 = 1.73;$$

$$D(I4,E1) = 0.25 + 0.32 + 0.0 = 0.57;$$

$$D(I4,E2) = 0.18 + 0.73 + 0.50 = 1.42;$$

$$D(I5,E1) = 0.28 + 0.73 + 0.50 = 1.52;$$

$$D(I5,E2) = 0.0 + 0.0 + 0.0 = 0.0;$$

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 7.2.

It can be seen from the above examples, that the system is able to pickup the right example, though there may be incidents when, due to inappropriate distance, a wrong example is picked up. In HEBMT system, all distances and weights have been

SN	Translated English output	Matched Example No.
I1.	<i>Mary is going on Monday.</i>	E2
I2.	<i>Sudhir was beating my brother.</i>	E1
I3.	<i>My axe can/may cut any material.</i>	E1
I4.	<i>My younger brother loves me.</i>	E1
I5.	<i>His/her brother will/shall come back on Sunday.</i>	E2

Table 7.2: Example Matching

defined on the basis of heuristics, which can be an adhoc method to do so. A linear programming scheme has to be used to correctly adjust the weights and distances with the growth of example-base. However, in the current implementation, we are not using any optimization method, but still we are getting quite good results, which supports the suitability of our distance function.

7.5 Target Language Sentence Generation

After finding the example with minimum distance, the target language sentence is generated using the target pattern of that example. Let us illustrate this by taking the input sentence I1 of Section 7.4, which matches with example sentence E2. Steps 1(a) to 1(f) below, show the process of translation.

- 1(a). *meri somavara ko ja rahi he* {Input sentence}
- 1(b). <snp> <npk2> <mv> {Syntactic grouping}
- 1(c). (*Mary*) (*Monday*) (*go*) {English translation of syntactic groups.}
- 1(d). <snp> <mv> {on} <npk2> {Target pattern of example E2}
- 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 syntactic units identified by the finite 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. Different files for tense mapping and for verb form mapping are used. For example, in the above case, root verb was 'go', tense was 'present continuous', number was 'singular', gender was 'feminine', and the root verb 'go' was translated as 'is going'.

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

Chapter 8

Implementation and Experimentation

The block schematic diagram of the HEBMT system is shown in Figure 3.2 of Chapter 3. Every module shown in the Figure 3.2 has been implemented separately. The implementation was carried out in a top down fashion. As mentioned in earlier chapters, we have developed a Hindi to English translation system as a case study using the HEBMT approach. At the time of design, we have separately identified the modules and submodules which are source language dependent, target language dependent, language pair dependent and, totally source/target language independent. The functions of each module and interdependency among them was also identified in order to make them compatible with each other. The inputs and outputs of each module were examined carefully. The separation of data and program is provided to enable easy experimentation.

Before starting the implementation, we examined various programming languages keeping in view the available man power skills who will maintain the system and also the easy availability of required hardware and software needed to install the

system. The availability of debugging tools and other facilities were also considered while selecting the programming language. We have used C language in UNIX based environment for system development, which is one of the the most commonly used programming language fulfilling above mentioned requirements.

8.1 Implementation of Modules of the HEBMT System

The modules to extract the root-verb (RVEM) and to identify the syntactic units (ISG module) in an input sentence have been implemented as finite state machines. A finite state machine uses the expectation driven analysis. After reading a symbol certain expectations are generated. Since a word can have more than one category and more than one meaning, the finite state machines for these two modules have been implemented as two way machines, which move in backward direction after failing in forward direction, thus trying out all possible paths before halting. The finite state machines use best first search strategy for trying out all possible paths. The implementation details of main modules of our system have been presented in the following paragraphs.

Implementation of RVEM:

The input to the RVEM module is a source language sentence. The module extracts the verb part and the rest of the sentence is passed to the morphological analyzer module. Main submodules and data files used in the implementation of the RVEM are shown in Figure 8.1.

As shown in Figure 8.1, the module uses two mapping files `verb_form.c`, `tense_map.c`, and a file `verb_lexicon.h` which contains lexicon entries for the root verb. The file

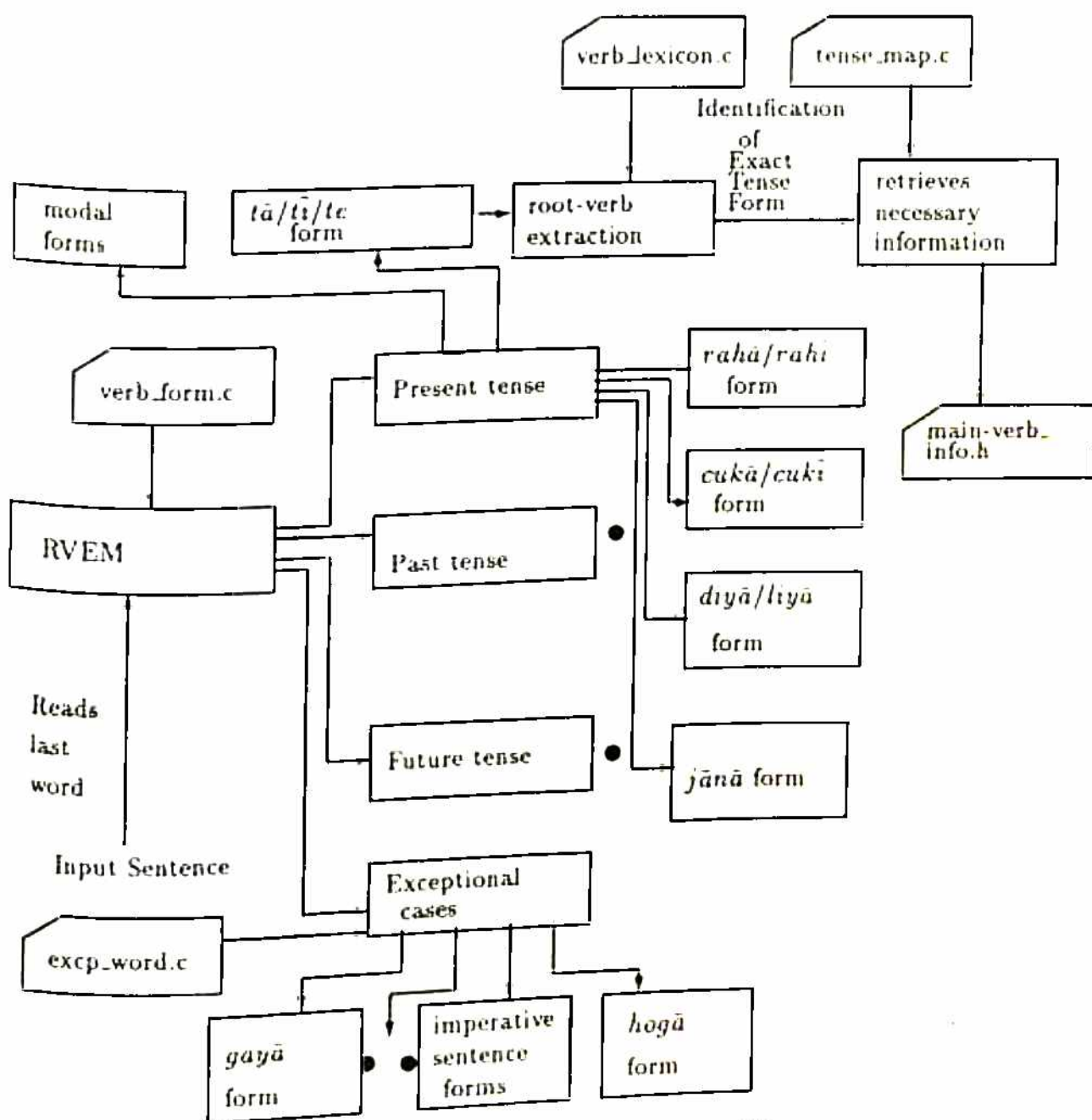


Figure 8.1: Information Flow Diagram of RVEM

verb_form.c defines the mapping from different Hindi verb-endings to the verb-forms in English. For example, if the Hindi verb-ending is 'hê', the mapping states that the sentence is in 'present tense', 'number' of the subject is 'plural', and 'gender' of the subject is 'not known'. However, the exact tense form¹ is identified after further analysis. Figure 8.1 shows some of the modules which are used to identify the present indefinite active tense (pria).

Another file excep_words.c is also used, which contains the list of exceptional verb endings like 'gayā, 'hogā, etc.'. The file tense_map.c defines the mapping between the tense of the verb to the corresponding auxiliary part, and the type of main part of the target language. For example, if after the analysis, the tense is identified as 'prpca (present perfect continuous active)', and the subject is 'singular', then the auxiliary part is 'has been' and the main part is the 'past participle form' of the root verb.

Implementation of ISG Module:

Figure 8.2 shows some of the main submodules and datafiles used by the ISG module. A separate sub-module has been designed for each syntactic category which performs the necessary analysis and generates expectations for the next word. For example, S_v , S_n , S_p , S_{ad} , S_{adj} , S_{pos} modules analyze the verbs, nouns, pronouns, adverbs, adjectives, and possessive cases respectively.

The input to the ISG module is in the form of three flat files, word_array.h, category_array.h, and meaning_array.h which are output files of morphological analyzer (MA) module, and main-verb_info.h which is an output file of the root-verb

¹For example, the present tense has seven forms: present indefinite active, present continuous active, present perfect continuous active, present indefinite passive, present continuous passive, present perfect passive, and some modal forms.

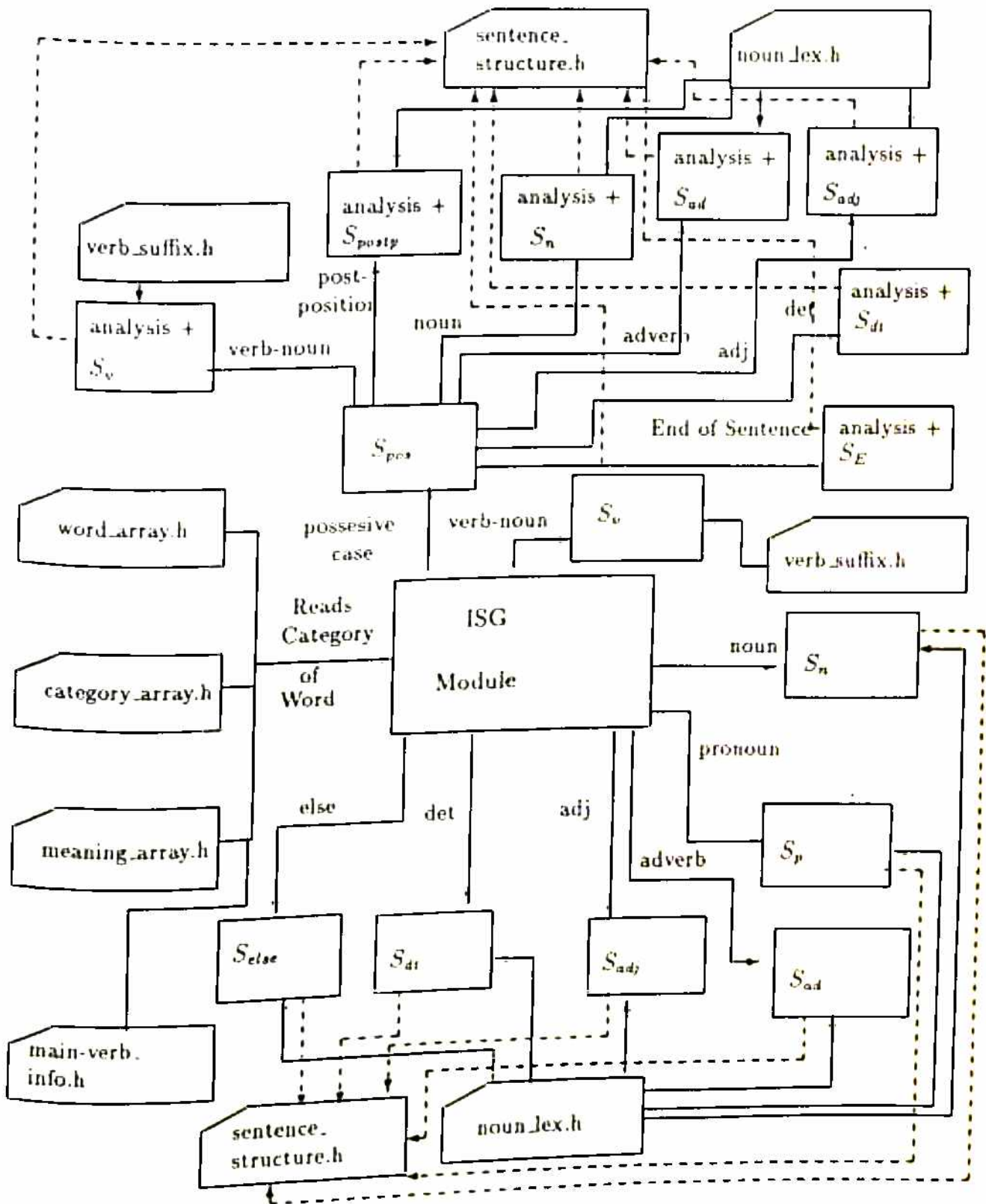


Figure 8.2: Information Flow Diagram of ISG module

extraction module (RVEM). Since a word can have many categories, and for each category the word can have many meanings. Therefore, the record for each word in the lexical database is a variable length record. The morphological analyzer breaks the record of each word into three parts so that the storage can be minimized and the required information can be retrieved easily. The file `word_array` contains the root word and the number of categories that word has. The file `category_array.h` has information about all categories, and the number of meanings each category has. The third file `meaning_array.h` contains all meanings of a word, and other semantic information attached to that meaning. The output of the ISG module is another flat file indicating the number of syntactic units in a sentence, their types, and their order in the input sentence. The finite state machine (ISG) has been designed in a modular form so that it is easy to make changes, whenever required. During the system development and testing, many times we had to make changes after discovering new patterns which were not initially included in our module. The modifications to include these new pattern could be made easily due to the top down strategy used during the design.

Implementation of Morphological Analyzer Module:

Figure 8.3 shows the modules and different data files used in implementing the morphological analyzer.

The main database used by the morphological analyzer is the lexical database which has entries of source language words in their root form with their translation in the target language and the other syntactic and semantic information (Appendix C). Besides this, the other files used by the module are the suffix and extension files to retrieve the root word.

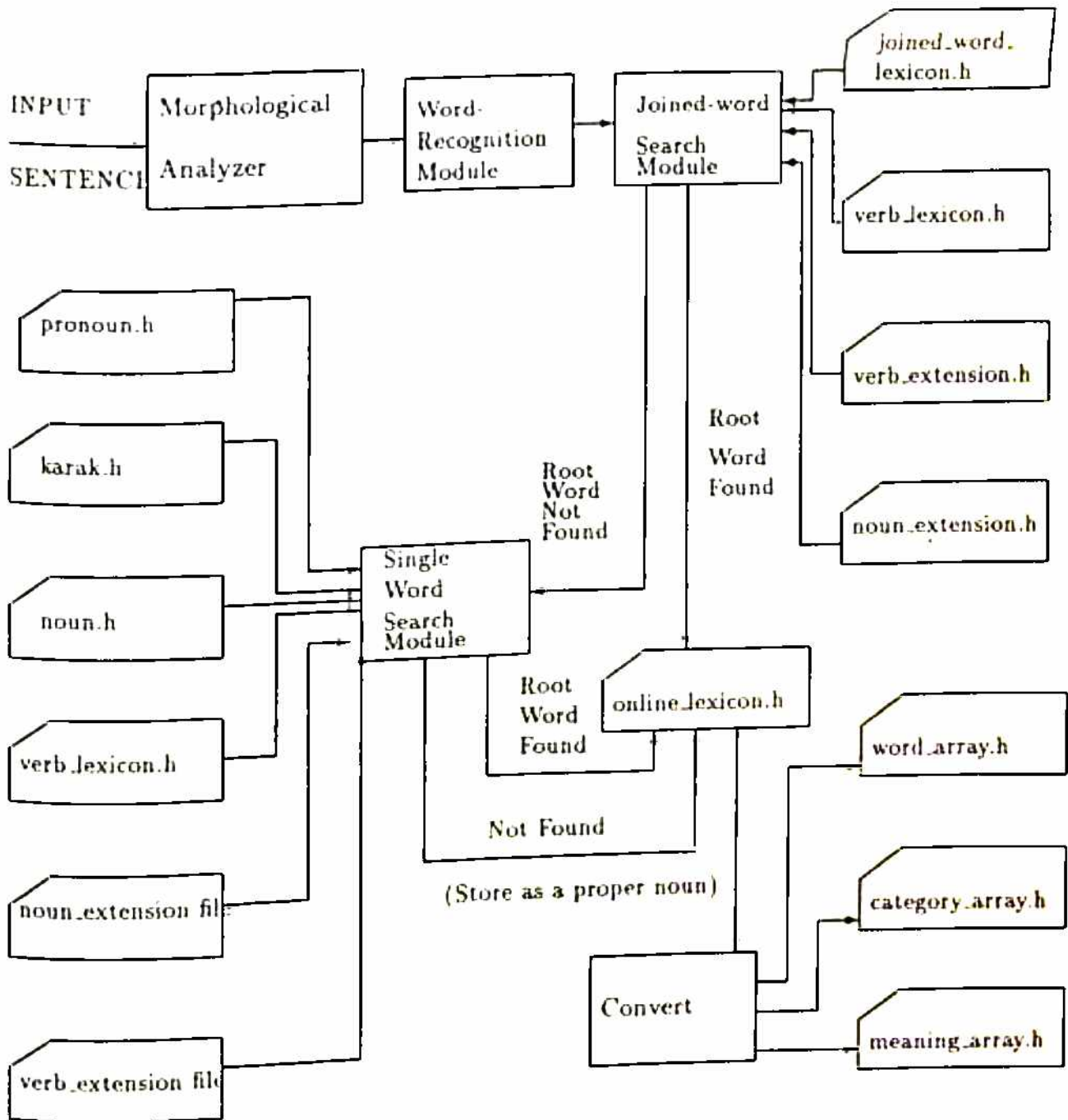


Figure 8.3: Information Flow Diagram of MA module

The lexical database has been currently implemented as a flat file and it has been divided into smaller flat files to improve the efficiency of the system. Heuristics are used for efficient searching and identification of the root word. The MA module produces all necessary information about each word in a organized way such that the ISG module can use it as its input. Each suffix file has been implemented as a separate structured array. The file `verb_extension.c` contains all possible extensions or suffixes which a verb can have, when used in the middle of the sentence. For example, '*kiyā huā, karatā huā, karane, karana, etc.* ', are some of the forms which a root verb '*kara*' can take. Similarly, the file `noun_extension.c` contains possible plural suffixes for nouns.

Implementation of modules for creation of example-base, example matching and translation:

The block schematic of the modules for example-base creation, example matching, and translation process is shown in Figure 7.

The information flow diagram of the modules for example-base creation, example matching, and translation process is shown in Figure 8.4. We have shown all three modules in a simple figure since these three modules are very much interrelated. The example-matching module (EM) uses example-base for matching. The example-base has been partitioned and flat files have been used to store the example-base. The EM module first reads the verb-type of the input sentence, and tries to match with existing types in `top_index.h` file. The file `top_index.h` represents the first level or top level index (refer to Appendix F, and `verb_type` is used as the primary key for searching). If a match is found, it retrieves the address of the next level index, where the number of syntactic units of the input sentence is the 'key' for matching. The

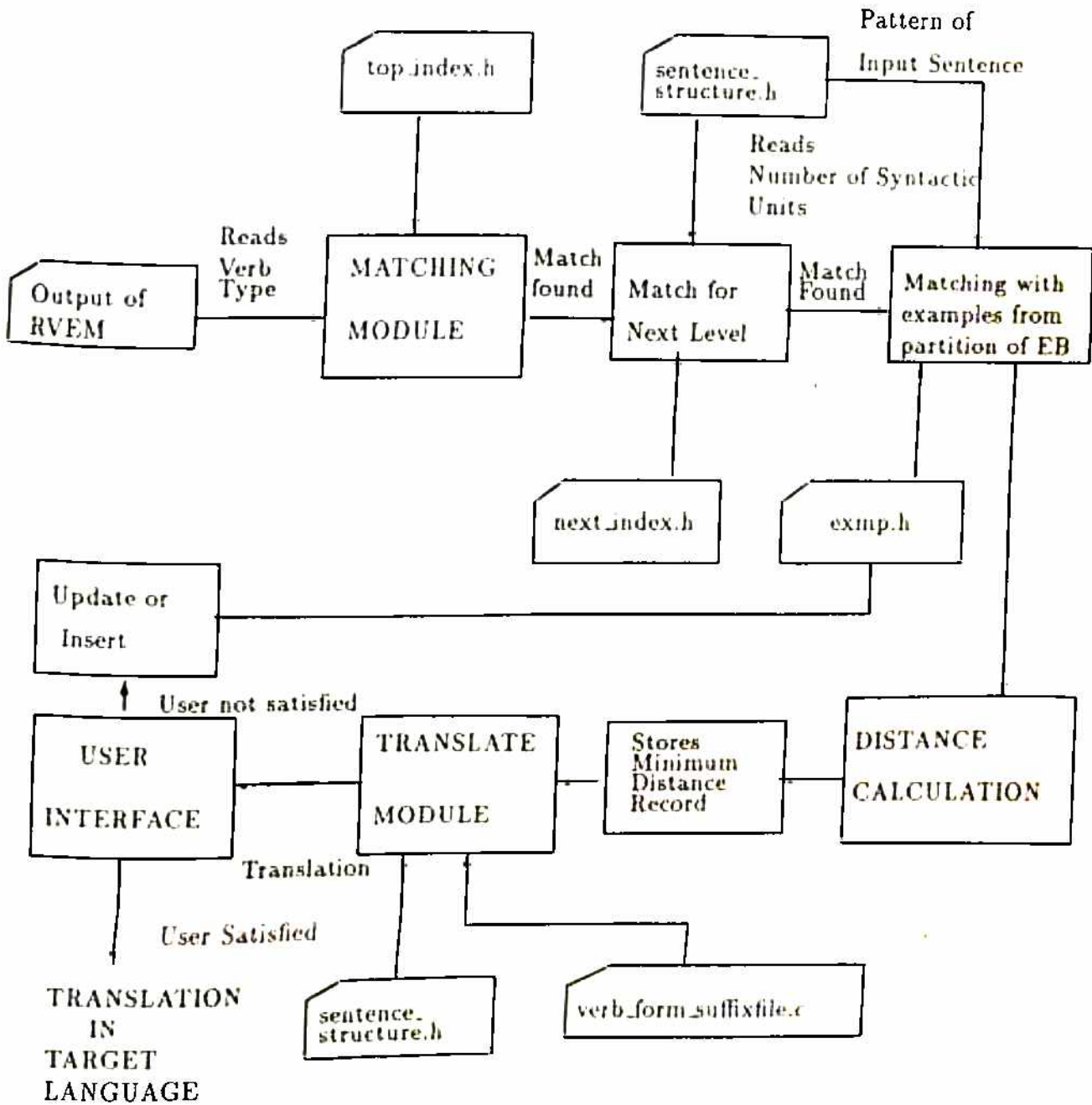


Figure 8.4: Information Flow Diagram of EM module

output of the ISG module provides the information about the number of syntactic units in the input sentence. If a match is found, it retrieves the address of partitioned example-base, which contains the abstracted examples similar to the input sentence. Whenever a match is not found, the input sentence is entered as an example in the example-base of appropriate partition. If such a partition does not exist, the system creates new partition on its own.

After reaching an appropriate partition, the pattern of the input sentence is matched with all examples of that pattern. Whenever an exact match is found, the distance between the input sentence and the example is evaluated using the distance function. The distance function makes use of syntactic and semantic information of each syntactic unit to calculate the closeness between the input sentence and the example as explained in Chapter 7. The translation module translates the input sentence according to the example which has been found closest to input sentence. It makes use of translation of syntactic categories, and finds proper translation of verb part using the `verb_form_suffix_file`².

If the exact pattern is not found, a new example is inserted into the example-base. At the time of insertion of a new example in the example-base, the system needs developer/user's help for entering the target language pattern. The system provides the instructions for entering the target pattern. Rest of the information is extracted by the system itself from the input sentence.

All modules have been developed in C. Even though the system has not been written in an object oriented language, we have attempted to make use of hierarchical modular design to facilitate easy diagnosis and upgradation.

²This file provides the information about the suffixes to be removed or to be added to change the target root verb to past, past participle form, or continuous form.

8.2 Experimentation

Whole process of experimentation was divided into four phases. In the first phase of experimentation, we wanted to create a sample example-base, and find the degree of compression obtained using the process of abstraction. A variety of input Hindi sentences were taken for testing the system. The Hindi text was taken from three different sources. To begin with, we wanted to test a wide variety of Hindi input sentences, so that the system can have an example-base for different kinds of sentences. Therefore, we selected a book [Sin85] by Dr. S.B.Singh as the source of example Hindi sentences. The book has a variety of Hindi sentences having different patterns. We started building our example-base with the sentences which were given in the Appendix 1 of the book. There are 92 sentences, and out of these, our system was not able to handle the sentence '*ye rahe apake rūpaye*'. In the list of these sentences, some of the sentences are very peculiar like '*āpa kahā ke navāba āe*', and '*āpa to sahāba ṭhhare*'. The System is able to translate these sentences as 'You were nawab of what place' and 'You remained master of course'. Out of these 92 input sentences, 55 sentences were entered as examples.

The second source of examples was a short story from a story book. Story had 48 sentences and out of them 34 sentences were entered as example sentences. As it can be seen from the Appendix G, all sentences of the story were long and had more number of syntactic units in comparison to the first set (sentences from book [Sin85]) of input sentences. Therefore, more sentences were needed to be entered as examples for the story.

The third set was taken from a CSIR science book [B.P92]. The CSIR book was selected because of two reasons. Firstly, it has sentences which differ in nature from

the earlier two sources. Secondly, both Hindi as well as English versions of the book were available and it was easy to enter the target example patterns. 90 sentences were translated from the CSIR book, and out of that 33 sentences were entered as examples. It is evident from the above data that after getting acquainted with a variety of sentences in structure and in number of syntactic units, the system starts obtaining maturity.

In the second phase of experimentation, most of the sentences from part 2 of Dr. Suraj Bhan Singh's book [Sin85] were tested. These sentences cover a wide variety of structures, which Hindi sentences can have. Our system translated 260 sentences quite appropriately, while for some sentences system faced problems. These problems have been discussed in chapter 9 in detail. Out of these 260 sentences, 60 sentences were entered as example sentences and rest of the sentences were translated using the existing example-base. Thus the rate of the growth of example-base was much lower in comparison to the first phase. Figure 8.5 shows the relationship between the total number of input sentences and the number of sentences entered as examples. It can be seen from the figure that the growth of example-base is decreasing with increase in the number of examples. Therefore, we can expect that after sometime the example-base will attain maturity, i.e., rarely a new example will have to be added in order to translate a new sentence, and that is the ultimate goal of any example-based system.

In the third phase of experimentation, we asked some of our friends to make new sentences using the existing lexical database. They provided 30 sentences to us and out of these 30 sentences, the system was able to translate 27 sentences appropriately. The three sentences which could not be translated properly are :

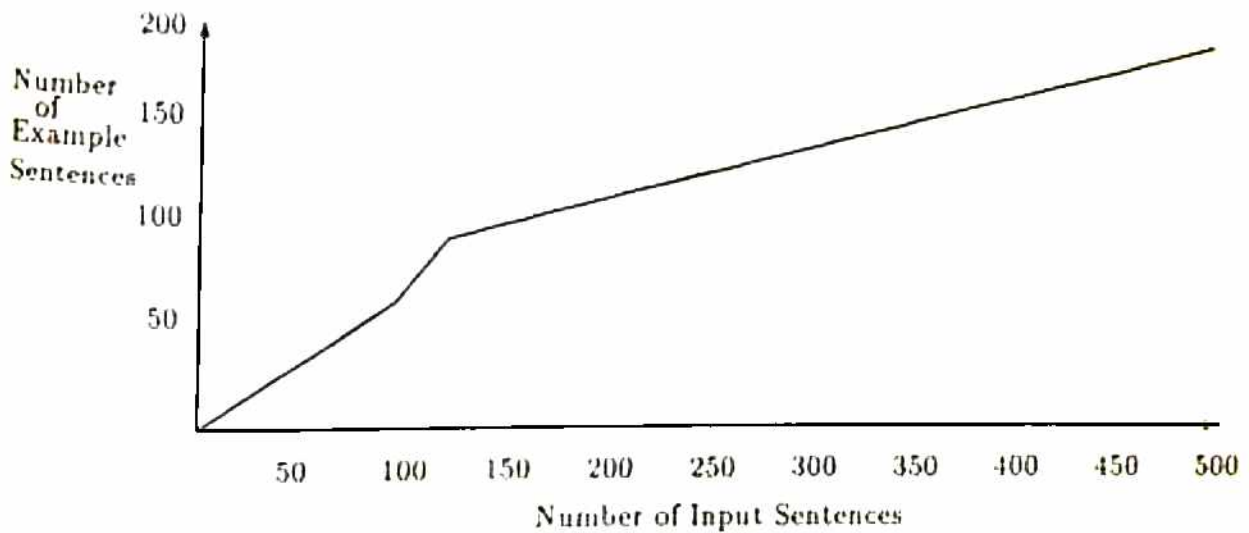


Figure 8.5: Rate of Growth of Example-base

1. *kāma karane mē bahuta mushkila ā rahī hē.*

The system translated it as 'It is becoming very difficult in doing work/function'.

2. *samaya se daphtara jānā cāhiye.*

The system translated the above sentence as 'Office should/ought to go by time'.

However, the sentence *hamako samaya se daphtara jānā cāhiye.* was translated perfectly as 'we should/ought to go to office in time'.

3. *tuma pēse bahuta kharca karatī ho.*

This was translated as 'You spend very money/paisa'. However, the sentence '*tuma bahuta pese kharca karatī ho*' was translated as 'You spend lots of money/paisa'.

It can be noticed from the above sentences that the sentences for which the translation was not satisfactory, were themselves not perfect from the grammatical point of view. However, such sentence constructions are often used in our daily routine. At the time of system development, mostly grammatically correct sentences were examined, and the finite state machine was designed accordingly. We can translate such sentences by developing a separate raw example-base, where sentence

analysis is not done and translation is achieved by matching words of the input sentence with the words of the example sentences.

In the fourth phase of experimentation, we again translated all those sentences which we had translated in phase 1 and in phase 2. This phase of experimentation was done to check the reliability of the distance function. We wanted to see whether we get the same translation as we got at the time of example-base development or does it create ambiguities. It was possible that during the second run, the input sentence could have matched with another example with smaller distance (as discussed in Chapter 3 while explaining the shortcomings of an example-based system), and may result in totally wrong translation. It was found that for only two sentences we obtained the translations which were different from what was during the time of development. The input sentence '*donō bhāī bahuta dukhi the*' was translated as 'Both brother had very sad', instead of 'Both brothers were very sad'. Another input sentence '*sudhira ko unaki ādata pasanda āī thī*' was translated as 'Sudhir had liked a their habit' instead of 'Sudhir had liked their habit'.

These results indicate that most of the time the distance function is able to pick up the right example from the example-base. Although, the design of distance function is based on the heuristics and on the observations made during the analysis of translation process, the results are very encouraging. We can expect even better results by dynamically adjusting the weights with the growth of example-base.

Chapter 9

Conclusions and Discussions

In this thesis, we have proposed a practical and comprehensive model (HEBMT) for machine translation. Effort has been made to capture the best of traditional models and example-based model. A prototype to translate from Hindi to English has been developed as a case study.

9.1 Features of HEBMT Approach

In our approach, we have refrained from emphasizing complete analysis and developing its understanding. Most of the knowledge is incorporated in the FSM (finite state machine) which has been designed by examining large number of examples of the language pair. As the task of a true faithful translation, based on complete analysis and understanding, is very complex and may not be feasible with limited resources, a practical approach is to generate an acceptable translation with limited knowledge incorporated in the system. The HEBMT system is designed to cater to this philosophy. Our aim has been to design a system which automatically generates 90% acceptable translation and only in 10% cases post-editing is required.

It may appear from the methodology that one of the limitations of the HEBMT

system is that the translation in the target is obtained for each syntactic unit independent of the other. However, our implementation caters to a large number of situations which require examination of other syntactic units specially in the case of pronouns and verbs. The finite state machine is designed to take care of such contextual information based on expectations generated by the sequence of individual syntactic units. If a syntactic unit is expected to have multiple meanings, which cannot be resolved by this process, alternatives are provided for post-editing (for details please see discussions on the PE module in section 3.3).

It is known that the verbs play the most important role in any sentence. The same verb in the source language may map to different verbs in the target language depending upon the nature of other constituents of the source language sentence. A detailed analysis was done to examine the behaviour of verbs of Hindi when translated in English. For only a very few verbs, we were able to make disambiguation rules based on semantic tags. In most of the other cases, either it was not possible for us to make rules (represent knowledge in explicit form), or in some cases, more than one rule was needed. Therefore, we decided to disambiguate verb meanings through the example-base. This feature is expensive as many times we have to enter similar examples due to multiple meanings of a verb. However, if the verb of source sentence has only one meaning, then the system captures the meaning from the lexical database instead of the example-base.

Another feature of the system is that very rudimentary analysis is being done on the input sentence. In traditional approaches, an input sentence is parsed by trying out all possible combinations and analyzing the whole sentence. This job of parsing in itself is quite complex and time consuming. A good portion of time needed for translation is taken by the parser. Most of the time, the system recursively analyzes

the input sentence to get the possible groupings of the words in a sentence. This is a time consuming process. On the contrary, in our approach, the analysis based on finite state machine is fast. However, the coverage depends upon the examples considered at the time of design.

Similarly, in traditional approaches, a text generator has to generate proper connectives of target language based on the properties of its constituents. For example, if the target language is English, then it is almost impossible to construct rules for preposition disambiguation. Though for most of the cases, each language has specifications for its connectives based on the subject, object, and the type of the sentence, it is very difficult to encode that knowledge in the form of rules. A deep analysis is required to correctly identify the subject and object of the sentence. The HEBMT system captures the required knowledge from the example-base. The developer does not have to collect, analyze and encode knowledge in the system. The system keeps getting knowledge implicitly through examples, and attains maturity after having a good number of examples in its example-base.

9.2 Limitations and Suggestions for Future Work

Every natural language has some very peculiar type of sentences which are translated in the target language in a fixed manner as used by the native speakers of that language. If the system tries to analyze such sentences it may sometimes fail to yield desired translation. However, these sentences can be better translated either by adding a raw example-base where a word to word matching is performed or by making special entries in the lexical database or by post-editing. Some of the types of the sentences which our system as it is, is not able to translate are discussed

below.

1. *vaha rata bhara soyā rahata hē.*

Our system finds the main verb of the above sentence as 'raha' and not 'so'. However, our system can handle it if we incorporate 'rahata form' as a constant and generate the expectation of verb forms which have the extensions of the form of 'yā' form. This can also be handled by incorporating 'soyā raha' in lexical database or by adding it to raw example-base.

2. *unakī pahūca bahuta dura taka hē.*

This is translated as 'Their reach is to/till very far away' by our system. A better translation can be obtained by post-editing or by adding raw example in this case.

3. *sudhīra ko yaha bata pata hē.*

Our system is unable to identify that the word 'patā' is behaving like a verb here and 'patā ho' is the main verb of the sentence. Even if we had the entry for the root verb 'patā ho' and added the feature so that system identifies the root verb, the problem could not be solved. Because the system is not able to differentiate it for other sentences where 'patā' is behaving like a noun, for example 'yaha rama kā pata hē'. The root verb 'ho' is a very typical verb and in many cases it behaves like an auxiliary verb and in many cases it behaves like a main verb. We faced lots of difficulty in dealing with 'ho type' verbs. These types of sentences can be translated by adding similar sentences as raw examples.

4. *yaha kitāba merī likhī huī hē.*

Our system as it is, translates it as 'This book is written by mine', though it is able to translate correctly 'yaha kitāba mere dvārā likhī huī he' into 'this book is written by me'. This can be corrected by post-editor.

5. *vaha āpakā bhāī raha.*

The above sentence is translated as 'He was your brother'. Similarly, the system translates the sentence as '*vaha apaka ghara raha*' as 'He was your house'. While, if we translate these sentences through raw example-base, we can obtain the perfect translation.

It is well known that pronoun referencing is one of the major sources of ambiguities faced by a machine translation system. Hindi is very liberal with pronouns and only in a few cases differentiates between genders. On the contrary, in English, most of the time, a pronoun clearly identifies the gender of the person. In our system, knowledge has been incorporated in the finite state machine for proper disambiguation of the pronoun referencing within a sentence so that it can properly translate Hindi pronouns into English pronouns.

In order to provide knowledge to the finite state machine, a large number of example sentences were examined by us. But during the experimentation, the finite state machine encountered some of the situations which it was not able to handle.

For example, in our system, the sentence '*vaha acchī sāṛī becatā hē*' is correctly translated as 'He sells good sarees'. However, the sentence '*vaha sāṛī acchī becatā hē*' is translated as 'that saree sells good'. Here, the system does not have the knowledge that the act '*becatā*' will be done by human only and '*vaha*' should be translated as 'he'. However, this knowledge can be incorporated in the finite state machine. The system treats the above sentence equivalent to the following sentences.

'vaha sāṛī acchī dīkhatī hē', or "*vaha sāṛī acchī laga rahī hē*" etc.

Some of the sentences were not translated because the word '*vāla*' was attached to an adjective like '*motā vālā, lāla vālī, etc.*' The system can handle these sentences if the complete phrases '*motā vālā*' or '*lāla vālī*' are entered in the system as lexicons. However, if such an entry is done for every adjective, it will triple the

adjective entries. Therefore, a mechanism to get root word from such adjectives has to be included in the system. This is not a difficult task to incorporate. However, in the current implementation we have not added this feature.

In the above discussion, we have highlighted some of the limitations of our system and a few suggestions to overcome them. However, our work can be extended in different directions some of which are listed below.

- Suppression of Adverbs

While building the abstracted example-base, we found that many a times, we had to enter a new example because in input sentence position of adverb was different from the example sentence or structure of the input sentence was exactly same as the example sentence if we had removed the adverb from the input sentence.

For example, the following sentences

1. *vaha kanapura jā rahā hē.*
2. *kala vaha kānapura jā raha hē.*
3. *vaha kala kānapura jā rahā hē.*
4. *vaha kānapura kala jā rahā hē.*

are exactly same in structure and they differ only by the position of an adverb.

To translate these four sentences we have to enter four example sentences. At this point, we examined the movement of all type of adverbs from the source language Hindi to the target language English. We also found that in many cases adverbs could be suppressed at the time of example matching, and at the time of translation they can be reintroduced. However, for some adverbs it was not possible to suppress them. So a detailed analysis of movements from source to target is needed and also feasibility has to be judged in terms of processing time and effect on the quality of translation.

- Consideration of Interrogative and Negative sentences

In the current implementation, we have considered only affirmative sentences, however, the ultimate goal is to translate all types of sentences. It is also possible to convert other kinds of sentences into affirmative and the translated version could be transformed into corresponding interrogative/negative sentences.

- Exploiting parallelism in HEBMT architecture

As can be seen from the architecture of HEBMT, there is lot of scope for incorporating parallelism. [EKH+93] found that EBMT systems provide better performance when implemented on massively parallel computers. The major source of inefficiency in example-based systems is attributed to large search time and the time required to find the best match. Both of these processes could be parallelized. Furthermore, the morphological analysis, which is also time consuming can be parallelized by invoking different paradigms through different processors.

- Exploration for other Indian languages

Our current implementation is for translation from Hindi to English. As discussed in Chapter 3, many of the modules of our system are language independent, therefore, this approach can also be explored for other languages. In our system, since most of the analysis is done only on the input sentence, very few changes will be required if we change the target language. Changes will be mostly required in the finite state machine while generating the target language equivalent. Most of the Indian languages are similar in structure and are verb-final languages. Only minor changes will be required if the source language Hindi is replaced by another Indian language which is similar in structure. In this case, mainly suffix files in morphological analysis will be changed and some changes will be needed in the finite state machine.

- Statistical resolution of meanings

As pointed out earlier, usually the meanings of the words with multiple senses get resolved using context within a sentence. The statistical information on word bi-gram, tri-gram, n-gram etc. could be used to generate appropriate lexical entries leading to a probable sense resolution.

Though the system has not been tested on a large scale as it required large bilingual corpus and a large lexical database, we have tested the system for large variety of constructs normally used. There was no readymade corpus and database available, and the testing was carried out by creating our own lexical database and example-base. We are confident that our system, when grown to full scale, will continue to yield fairly accurate translation with the limitations as pointed out earlier.

References

- [A.B87] A.B.Tucker. "Current Strategies in Machine Translation Research and Development", In *Machine Translation, Theoretical and Methodological Issues*, pp. 2-41. (ed.) Sergei Nirenburg, Cambridge University Press, 1987.
- [B.P92] B.Phondke. "Life: From Cell To Cell, CSIR Golden Jubilee Series". Publications and Informations Directorate, 1992.
- [CD86] C.Stanfill and D.Waltz. "Towards Memory-Based Reasoning". *Communications of the ACM*, 29(12), 1986.
- [Com66] Automatic Language Processing Advisory Committee. "A Report on Language and Machines: Computers in Translation and Linguistics", 1966.
- [EH91] E.Sumita and H.Iida. "Experiments and Prospects of Example-Based Machine Translation". In *28th Annual Meeting of the Association of Computational Linguistics*, 1991.
- [EKH+93] E.Sumita, K.Oi, H.Iida, T.Hiyuchi, N.Takahashi, and H.Kitano. "Example-Based Machine Translation on Massively Parallel Processors". In *Proceedings of the IJCAI-93*, 1993.
- [GD93] G.V.Singh and D.K.Lobiyal. "A Computational Grammar for Hindi: Simple Sentences and Beyond.". In *Proceedings of Natural Language Processing Pacific Rim Symposium (NLPRS:93)*, Japan, 1993.
- [Hil60] B. Hillel. "The Present Status of Automatic Translation of Languages". *Advances in Computers 1*, 1:91-163, 1960.
- [H.K93a] H.Kitano. "A Comprehensive and Practical Model of Memory-Based Machine Translation". In *Proceedings of IJCAI-93*, 1993.
- [H.K93b] H.Kitano. "Challenges of Massive Parallelism". In *Proceedings of IJCAI-93*, 1993.

- [H.M93] H.Maruyama. "Pattern-Based Translation: Context Free Transducer and its Applications to Practical NLP". In *Proceedings of National Language Processing Pacific Rim Symposium(NLPRS'93)*, Fukuoka Japan, 1993.
- [J.A87] J.Allen. "Natural Language Understanding" Benjamin/Cumming publishing Company, 1987.
- [J.H93] J.Hutchins. "Latest Developments in Machine Translation Technology:Beginning a new era in MT Research". In *Proceedings of the MT Summit IV, Japan, 1993*.
- [JJR95] Renu Jain, Ajai Jain, and R.M.K Sinha. "Emerging Trends in Machine Translation between English and Indian Languages". Technical Report TRCS - 95 - 229, I.I.T. Kanpur, Department of Computer Science and Engineering, January 1995.
- [Jos91] A. K. Joshi. "Natural Language Processing". *Science*, 253(1242-1248), 1991.
- [JRA81] J.G.Carbonell, R.E.Cullingford, and A.V.Gershman. "Steps Toward Knowledge-Based Machine Translation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4)(376-392), 1981.
- [JRJ95a] Renu Jain, R.M.K.Sinha, and Ajai Jain. "A Pattern Directed Hybrid Approach To Machine Translation Through Examples". In *Proceedings of Symposium on Natural Language Processing '95 (SNLP'95) to be held in Aug. 95 in Bangkok, Thailand (Accepted)*, 1995.
- [JRJ95b] Renu Jain, R.M.K.Sinha, and Ajai Jain. "Role of Examples in Translation". In *International Conference on Systems, Man and Cybernetics, to be held on Oct 22-25, 1995 in Vancouver, Canada (Accepted)*, 1995.
- [JRJS95] Renu Jain, R.M.K.Sinha, Ajai Jain, and Rakesh Srivastava. "HFSM : A finite state machine for Analyzing Hindi Sentences". In *Proceedings of Symposium on Natural Language Processing '95 (SNLP'95) to be held in Aug. 95 in Bangkok, Thailand (Accepted)*, 1995.
- [Maa84] H.D. Maas. "The MT System SUSY". Paper presented at The ISSCO Tutorial on Machine Translation, 1984.
- [M.N85] M.Nagao. "A Framework of a Mechanical Translation between Japanese and English by Analogy Principle", 1985.
- [Nar95] Narayana. *Anusarak: A Device to Overcome the Language Barrier*. PhD thesis, I.I.T., Kanpur, 1995.

- [Nir87] S. Nirenburg. "Knowledge and Choices in Machine Translation". In *Machine Translation- Theoretical and Methodological Issues*. Cambridge University Press, 1987.
- [OH92] O.Furuse and H.Iida. "Cooperation between Transfer and Analysis in Example-Based Framework". In *Proceedings of COLING-92*, 1992.
- [OSV93] O.Vikas, S.Younes, and V.Aggarwal. "Object Oriented Design of Semantically Driven Tag Based Hindi Sentence Analyzer". In *Proceedings of Natural Language Processing Pacific Rim Symposium (NLPRS:93)*, Japan, 1993.
- [P.T77] P.Toma. "SYSTRAN as a Multi-Lingual Machine Translation System in Commission of European Communities:Overcoming the Language Barrier". In Munich: Dokumentation Verlag, 1977.
- [RC93] R.M.K.Sinha and C.Sanyal. "Correcting Ill Formed Hindi Sentences in Machine Translated Output". In *Proceedings of Natural Language Processing Pacific Rim Symposium (NLPRS:93)*, Japan, 1993.
- [RKA+95] R.M.K.Sinha, K.Sivaraman, A.Agrawal, R.Jain, R.Srivastava, and A.Jain. "ANGLABHARTI: A Multilingual Machine Aided Translation Project on Translation from English to Indian Languages". In *International Conference on Systems, Man and Cybernetics, to be held on Oct 22-25, 1995 in Vancouver, Canada (Accepted)*, 1995.
- [RS90] C. Reisbeck and R. Schan. "*Inside Case-Based Reasoning*". Lawrence Erlbaum Associates, 1990.
- [Sin85] S. B. Singh. "*Hindi Ka Vakyatamak Vyakaran (A syntactic grammer of Hindi)*". Sahitya Sahakar, 1985.
- [Sin93] S. B. Singh. "Concepts of Semantic Field and Collocation in Hindi/Urdu Lexicography", 1993.
- [SM90] S.Sato and M.Nagao. "Towards Memory-Based Translation". In *Proceedings of COLING -90*, 1990.
- [S.S87] S.Singh. "A Feature Analysis of Equational Sentences in Hindi". In *Indian Linguistics* vol. 40, No. 1:1-17, 1987.
- [Ter83] W. Terry. "Language as a Cognitive Process, Vol. 1". *Communications of the ACM*, 1983.
- [Wei63] U. Weinreich. "Lexicography". In *Current Trends in Linguistics, VOL. 1, Ed. Thoms A sebek, Mouton*, pages 60-93, 1963.

Appendix A

List of Semantic Tags

“[insect]”

“[animal_group]”

“[animate]”

“[phy_obj]”

“[animal]”

“[bird]”

“[general]”

“[human]”

“[relation]”

“[post_holder]”

“[plants]”

“[thread]”

“[house_hold]”

“[inanimate]”

“[machinery]”

“[chemical]”

“[liquid]”

“[building]”

“[instrument]”

“[vehicle]”

“[place]”

“[edible]”

“[currency]”

“[body_part]”

“[natural]”

“[day]”

“[time]”

“[abs_idea]”

“[month]”

“[season]”

“[fixed]”

“[unfixed]”

“[static]”

“[activity]”

“[dynamic]”

“[illness]”

“[quality]”

“[feeling]”

“[habit]”

“[measurement]”

“[color]”

"[event]"

"[nature]"

"[topic]"

"[message]"

"[picture]"

Appendix B

HFSM Symbols

npk1	noun + <i>ne</i>
npk2	noun + <i>ko</i>
npk5	noun + <i>para</i>
npk8	noun + <i>taka</i>
npk7ank	noun + <i>ke pasa</i>
npk7nk	noun + <i>ke + lie/bāda/piche ..etc</i>
vnp	verb + <i>ne</i> , verb + <i>nā</i> ,
vn_ne_adj	verb + <i>ne</i> + adj
sadv	only adverb
pos_anonkarak	poss_case + <i>pasa</i>
pos_nonkarak	poss_case + <i>lie/bada/piche ..etc</i>
pos_nverb	poss_case + verb + <i>ne/nā</i>
pos_case	poss_case
snpk6	noun + <i>ka</i>
snp	noun, adj + noun, noun + <i>ka</i> + noun, noun + <i>ke</i> + noun
npk4	noun + <i>mē</i>

n _{pk7}	noun + <i>ke</i>
n _{pk6}	noun + <i>ka</i>
s _{adj}	only adjective
s _{advk3}	adv + <i>se</i>
n _{pk4_pos}	noun (animate) + <i>mé</i>
v _{npk6}	vnp form + <i>ka</i>
n _{pk3adj}	noun + <i>se</i> + adj
n _{pk6}	noun + <i>ka/ke</i> + det + noun + \$

Appendix C

Sample Entries from Lexical Database

Some of the sample entries from lexical database are given below.

chota (Hindi root word)

33 adj (address of syntactic information)

small:1;younger:4; (English meanings, address for other degree of adjective)

[];[relation]; (semantic tags for each meaning)¹

**

laṛakā

1 noun

boy:1;son:1; (English meaning, address for plural suffix entry)

[general];[relation];

**

laṛakī

3 noun

¹For some entries semantic tag is blank, this corresponds to default meaning, when matching is not found.)

girl:1;daughter:1;

[general];[relation];

**

bahuta

32 adverb

very:100; (the number 100 indicates that this word remains as it is)

[quantity];

##

33 adj

lots of:100;

[];

**

baraā

33 adj

big:2;elder:100;great:1;great:1;

[];[relation];[profession];[concept];

**

ōra

45 conj

and:100;

[];

##

18 pronoun

other:100;other:100;other:100;

[subject];[object];[n];

##

33 adj

more:100;

[];

##

32 adv

more:100;

[];

**

ulta

33 adj

opposite:100;left:100;

[concept];[body-part];

##

32 adverb

reversely:100;

[manner];

**

kala purja

7 noun

part:1;component:1;

[instrument];[machine];

**

limpha pravaha

7 noun

lymph flow:8;

[body_part];

**

usa

25 pronoun

he/she:100;him/her:100;that:100;

[subject];[object];[n];

**

hamāra

13 poss_case

we:100;ours:100;our:100;us:100;

[ank];[\$];[n];[nk];

**

usakā

15 poss_case

he/she:100;his/hers:100;his/her:100;him/her:100;

[ank];[\$];[n];[nk];

**

Appendix D

Examples of Translation for different Verb-forms

Following four sentences constitute the example-base. Our system is able to translate 50 input sentences with the help of these four sentences. Those 50 input sentences with their translated output are shown in the next section.

1. *jōna elisā ko paṛhātā hē.*
2. *merā dostā itavāra ko ayegā.*
3. *rāma ne mujhe dhokhā diyā he.*
4. *mujhe rāma ke dvārā dhokhā diyā jatā hē.*

D.1 Examples of Translation of Input Sentences in the Present Tense

- | | |
|---|----------------------------|
| 1. <i>rāma mujhe dhokhā deta hē.</i> | Ram deceives me. |
| 2. <i>rāma mujhe dhokhā de rahā hē.</i> | Ram is deceiving me. |
| 3. <i>rāma ne mujhe dhokha diyā he.</i> | Ram has deceived me. |
| 4. <i>rāma mujhe dhokhā deta raha hē.</i> | Ram has been deceiving me. |

- | | | |
|----|--|---|
| 5. | <i>mujhe rama ke dvāra dhokhā diya jata he.</i> | I is deceived by Ram. |
| 6. | <i>mujhe rama ke dvāra dhokha diya ja raha he.</i> | I is_being deceived by Ram. |
| 7. | <i>mujhe rama ke dvara dhokha diya gaya he.</i> | I has_been deceived by Ram ¹ . |

D.2 Examples of Translation of Input Sentences in the Past Tense

- | | | |
|-----|---|------------------------------|
| 8. | <i>rama ne mujhe dhokhā diya.</i> | Ram deceived me. |
| 9. | <i>rama mujhe dhokhā de raha thā.</i> | Ram was deceiving me. |
| 10. | <i>rāma ne mujhe dhokha diya thā.</i> | Ram had deceived me. |
| 11. | <i>rama mujhe dhokha deta rahā thā.</i> | Ram had_been deceiving me. |
| 12. | <i>mujhe rāma ke dvara dhokha diya gayā.</i> | I was deceived by Ram. |
| 13. | <i>mujhe rāma ke dvārā dhokhā diyā ja rahā thā.</i> | I was_being deceived by Ram. |
| 14. | <i>mujhe rāma ke dvārā dhokhā diyā gayā thā.</i> | I had_been deceived by Ram. |

D.3 Examples of Translation of Input Sentences in the Future Tense

- | | | |
|-----|--|--|
| 15. | <i>rāma mujhe dhokha dega.</i> | Ram will/shall deceive me. |
| 16. | <i>rama mujhe dhokha de rahā hogā.</i> | Ram will_be/shall_be deceiving me |
| 17. | <i>rama ne mujhe
dhokha diyā hogā.</i> | Ram will_have/shall
_have deceived me. |
| 18. | <i>rāma mujhe dhokhā,
detā rahā hoga</i> | Ram will_have_been/
shall_have_been deceiving me. |

19. *mujhe rāma ke dvāra dhokha diyā jātā hogā.* I will_be/shall_be deceived by Ram.
20. *mujhe rāma ke dvarā dhokhā diyā gaya hogā.* I will_have_been/shall_have_been deceived by Ram.

D.4 Examples of Translation of Input Sentences in Modal forms

21. *rama mujhe dhokha de sakatā hē.* Ram can/may deceive me.
22. *rama mujhe dhokha de sakatā tha.* Ram could/might deceive me.
23. *mujhe rāma ke dvāra dhokha dena cahīye.* I should/ought_to deceive by Ram.
24. *mujhe rama ke dvarā dhokha diya jā sakata hē.* I can_be/may_be deceived by Ram
25. *mujhe rāma ke dvara dhokhā diyā jana. cahīye* I should_be/ought_to_be deceived by Ram.
26. *mujhe rāma ke dvara dhokhā diyā jā sakatā hē.* I can_be/may_be deceived by Ram
27. *mujhe rama ke dvarā dhokha de diya jana. cāhiye* I should_have_been deceived by Ram.

D.5 Examples of Translation of Input Sentences with Different Forms of Noun Phrases

28. *sudhira sita ko pita raha hē.* Sudhir is beating Sita.
29. *sudhira kā barā bhāī mujhe parhātā hē.* Sudhir's elder brother teaches me.
30. *elisa mere dostā ko āvāja de rahī thī.* Elisa was calling my friend.
31. *vaha hamako pareshāna karatā hē.* He troubles us.

32. *usane saba logō ko pareshana kara diyā he.* He/she has troubled all people.
33. *mere pitāji ne darāivara ko niyukta kiya.* My father appointed driver.
34. *sitā ne mohana ko dhakka diyā.* Sita pushed Mohan.
35. *mohana ke dostā ke chote larake ne
gita ke bare bhāi ko parhaya thā.* Mohan's friend's younger son had
taught Gita's elder brother.
36. *mohana ko sitā ke dvarā pareshāna kiyā gayā.* Mohan was troubled by Sita.
37. *vaha larakā elisā ko pyara karatā hē.* That boy loves Elisa.
38. *mohana usa pera ko kātā rahā thā.* Mohan was cutting that tree.
39. *sudhira ne sāre gilāsō ko tora diyā.* Sudhir broke all the/whole glasses.
40. *daktara ne mere datō ko tora diyā.* Doctor broke my teeth.
41. *pitāji baccō ko le jā rahe hē.* Father are taking away children.
42. *ūta pera ko khā cukā thā.* Camel had eaten tree.
43. *vaha meri bahana ko pitatā hē.* He beats my sister.
44. *hamane sigareṭa ko chora diyā thā.* We had given up cigarette.
45. *sitā ko rāma ke dvārā parhāyā gayā he.* Sita has been taught by Ram.
46. *usa larake ne usa sundara laraki
ko badala diyā hē.* That boy has changed that
beautiful girl.
47. *elisā ne naye kapare ko pahana liya hē.* Elisa has worn new cloth.
48. *kāra marijō ko le jā rahī thī.* Car was taking away patients.
49. *elisā ki bahana kā laraka mere bhāi ki
laraki ko pīta rahā hogā.* Elisa's sister's son will be/shall be
beating my brother's daughter.
50. *mohana ke acche dostā ne
mohana ko dhokha diya thā.* Mohan's good friend had
deceived Mohan.

Appendix E

Mapping file for Different Tenses

This file maps the tense tag to the corresponding translation in English.

For example: prpa (Present Perfect Active) is mapped as 'has/have', verb_4 form

Some of the notations used are explained below:

bpr_suffix - (is,are as main verb)

bpa_suffix - (was,were as main verb)

Different verb forms :

1 : verb_1 : verbs as such

2 : verb_2 : verbs in present form (singular)

3 : verb_3 : verbs in past form

4 : verb_4 : verbs in past participle form

5 : verb_5 : verbs with ing form

0 : auxiliary verb as a main verb ***/ Each entry has the following structure.

'number','tense', 'constant_expression', 'verb form number'

"S","prcp","is_being/am_being",4,

"P","prcp","are_being",4,

"S", "prca", "is/am", 5,
 "P", "prca", "are", 5,
 "S", "prpca", "has_been", 5,
 "P", "prpca", "have_been", 5,
 "S", "prpa", "has", 4,
 "P", "prpa", "have", 4,
 "S", "prpp", "has_been", 4,
 "P", "prpp", "have_been", 4,
 "S", "prip", "is/am", 4,
 "P", "prip", "are", 4,
 "S", "m5", "can_be/may_be", 4,
 "P", "m5", "can_be/may_be", 4,
 "D", "m5", "can_be/may_be", 4,
 "S", "m1", "can/may", 1,
 "P", "m1", "can/may", 1,
 "D", "m1", "can/may", 1,
 "S", "pria", "", 2,
 "P", "pria", "", 1,
 "D", "pria", "", 1,
 "S", "bpr_suffix", "is/am", 0,
 "P", "bpr_suffix", "are", 0,
 "S", "bpr_suffix+howA", "is/am", 0,
 "P", "bpr_suffix+howA", "are", 0,
 "S", "pacp", "was_being", 4,
 "P", "pacp", "were_being", 4.

"S", "paca", "was", 5,
 "P", "paca", "were", 5,
 "S", "papca", "had_been", 5,
 "P", "papca", "had_been", 5,
 "S", "papa", "had", 4,
 "P", "papa", "had", 4,
 "S", "papp", "had_been", 4,
 "P", "papp", "had_been", 4,
 "S", "paip", "was", 4,
 "P", "paip", "were", 4,
 "S", "m4", "used_to", 1,
 "P", "m4", "used_to", 1,
 "D", "m4", "used_to", 1,
 "S", "m7", "could_be/might_be", 4,
 "P", "m7", "could_be/might_be", 4,
 "D", "m7", "could_be/might_be", 4,
 "S", "m2b", "could/might", 1,
 "P", "m2b", "could/might", 1,
 "D", "m2b", "could/might", 1,
 "S", "bpa_suffix", "was", 0,
 "P", "bpa_suffix", "were", 0,
 "S", "bpa_suffix+howA", "was", 0,
 "P", "bpa_suffix+howA", "were", 0,
 "S", "fucp", "will_be/shall_be", 5,
 "P", "fucp", "will_be/shall_be", 5,

- "S", "fuca", "will_be/shall_be", 5,
 "P", "fuca", "will_be/shall_be", 5,
 "S", "fupca", "will_have_been/shall_have_been", 5,
 "P", "fupca", "will_have_been/shall_have_been", 5,
 "S", "fupa", "will_have/shall_have", 4,
 "P", "fupa", "will_have/shall_have", 4,
 "S", "m12", "might_have_been", 4,
 "P", "m12", "might_have_been", 4,
 "D", "m12", "might_have_been", 4,
 "S", "fupp", "will_have_been/shall_have_been", 4,
 "P", "fupp", "will_have_been/shall_have_been", 4,
 "S", "fuiP", "will_be/shall_be", 4,
 "P", "fuiP", "will_be/shall_be", 4,
 "S", "m2a", "could/might", 1,
 "P", "m2a", "could/might", 1,
 "D", "m2a", "could/might", 1,
 "S", "paia", "", 3,
 "P", "paia", "", 3,
 "S", "m11", "should_have_been", 4,
 "P", "m11", "should_have_been", 4,
 "D", "m11", "should_have_been", 4,
 "S", "m9", "should_be", 5,
 "P", "m9", "should_be", 5,
 "D", "m9", "should_be", 5,
 "S", "fupa", "will_have/shall_have", 4,

"P", "fupa", "will_have/shall_have", 4,
 "S", "fupp", "will_have_been/shall_have_been", 4,
 "P", "fupp", "will_have_been/shall_have_been", 4,
 "S", "fuiP", "will/shall", 1,
 "P", "fuiP", "will/shall", 1,
 "S", "fupa", "will_have/shall_have", 4,
 "P", "fupa", "will_have/shall_have", 4,
 "S", "fupp", "will_have_been/shall_have_been", 4,
 "P", "fupp", "will_have_been/shall_have_been", 4,
 "S", "fuiP", "will/shall", 1,
 "P", "fuiP", "will/shall", 1,
 "S", "m13", "should_have_been", 4,
 "P", "m13", "should_have_been", 4,
 "D", "m13", "should_have_been", 4,
 "S", "m6", "should_be/ought_to_be", 4,
 "P", "m6", "should_be/ought_to_be", 4,
 "D", "m6", "should_be/ought_to_be", 4,
 "S", "m3", "should/ought_to", 1,
 "P", "m3", "should/ought_to", 1,
 "D", "m3", "should/ought_to", 1,
 "S", "m8", "could_be/might_be", 5,
 "P", "m8", "could_be/might_be", 5,
 "D", "m8", "could_be/might_be", 5,
 "S", "m10", "could_have/might_have", 4,
 "P", "m10", "could_have/might_have", 4,

"D", "m10", "could_have/might_have", 4,

"S", "fuia", "will/shall", 1,

"P", "fuia", "will/shall", 1,

"D", "imp", "", 1,

"S", "imp", "", 1,

"P", "imp", "", 1,

"S", "about_form", "about_to", 1,

"P", "about_form", "about_to", 1,

"D", "about_form", "about_to", 1,

"D", "Let_form", "", 1,

Appendix F

Sample Entries from the Raw and Abstracted Example-Base

In this appendix, we have shown few raw examples and abstracted examples corresponding to them. Raw examples are not stored in our actual example-base: Only abstracted examples are stored in the example-base. Raw examples are stored only for the convenience of the system developer. All the partitions of the abstracted example-base have been combined here. A record of each partition has been explained for reader's convenience. A different coding scheme for Hindi was used for the system which is given in the next page.

Coding Scheme for Hindi Alphabets Used in the System

a	A	i	l	u	U	q
अ	आ	इ	ई	उ	ऊ	क
e	E	o	O	M	H	z
ए	ऐ	ओ	औ			
k	K	g	G	f		
क	ख	ग	घ	ङ		
c	C	j	J	F		
च	छ	ज	झ	ञ		
t	T	d	D	N		
ट	ठ	ड	ढ	ण		
w	W	x	X	n		
त	थ	द	ध	न		
p	P	b	B	m		
प	फ	ब	भ	म		
y	r	l	v	S		
य	र	ल	व	श		
R	s	h	kR			
प	स	ह	क्ष			

Raw examples

1. jOna elisA ko paDZAwa hE
2. merA xoswa iwavAra ko AyegA
3. rAma ne muJe XoKA xiyA hE
4. muJe rAma ke xvArA XoKA xiyA jAWA hE
5. usa ladZake ne usa sunxara ladZakI ko baxala xiyA hE
6. sAXanA merI bahana lagawI hE
7. suXIra Gara meM hE
8. yaha kiwAba vixeSiyom ke lie hE
9. suXIra ko buKARA hE
10. saBI kiwAbeM bika gaIM
11. suXIra rAXA se ladZawA hE
12. Gara jAo
13. suXIra majaxUroM se makAna banavA rahA hE
14. mamMI ko una logom ke jAne ke bAXa hI rAhawa mila karawI WI
15. usa xina xonom BAI bahuwa KuSa We
16. eka xina mamMI Ora pApA ne unake xoswoM ke sAmane ladZanA SurU
kara xiyA
17. vARpa iMjana isa UrJA se pUrI relagAdZI calAWA hE
18. livara ise BaviRya meM upayoga ke lie saMcIwa kara lewA hE
19. kisI BI kAraKane meM kArya karane ke lie bijali kI AvaSyakawA howI hE
20. wuma merA inwajAra karo

Abstracted examples

First Level of Partitioning:

3 (number of entries in this level)

mv nindex.1h (main verb, pointer to the next level)

av nindex.2h (auxiliary verb, pointer to the next level)

imp nindex.3h (imperative verb, pointer to the next level)

/ End of First Level */*

Second Level of Partitioning:

Contents of file nindex.1h:

4 (number of entries at this level)

2 exmp2.1h (number of syntactic units, address of next level)

1 exmp1.1h

3 exmp3.1h

4 exmp4.1h

/ end of file nindex.1h */*

Contents of file nindex.2h:

2

2 exmp2.2h

3 exmp3.2h

/ end of file nindex.2h */*

Contents of file nindex.3h:

2

1 exmp1.3h

2 exmp2.3h

/* end of file nindex.3h*/

Third Level Partitioning:

Contents of file exmp1.1h

snp (sentence structure in the form of syntactic units)

9 bika jA 500 be sold (verb-category number, hindi-root, verb-form-suffix, english-meaning)

8 (address for the syntactic information for the above syntactic unit)

[thing] (semantic-tag of above syntactic unit)

1 V (target language pattern)

10 (number of corresponding raw example)

/* End of exmp1.1h*/

Contents of file exmp2.1h

snp npk2

4 paDZA 74 teach

1 3

[human] [human]

1 V 2

1

snp npk2

2 A 10 come

5 7

[general] [day]

1 V on 2

2

nPK1 nPK2

4 XoKA xe 9 deceive

11 29

[] [object]

1 V 2

3

nPK2 nPK7nk

4 XoKA xe 9 deceive

29 11

[subject] []

1 V 2

4

nPK1 nPK2

3 baxala 9 change

1 3

[general] [general]

1 V 2

5

snp snp

2 laga 500 be

11 3

[] [human]

1 V 2

6

snp npk3

4 ladZa 8 fight

1 11

[general] []

1 V with 2

11

/* End of exmp2.1h*/

Contents of exmp4.1h:

npk2 npk7 npk7nk sadv

16 rAhawa mila 34 get relieved

3 6 49 32

[relation] [general] [v-participle] [emphatic]

1 V 4 3 of 2

14

.....

snp npk1 npk7nk vnp

4 SurU kara xe 8 start

7 3 6 50

[day] [relation] [general] [v-participle]

1 2 V 4 3

16

snp npk2 npk4 npk7nk

4 saMciwa kara le 9 accumulate

7 27 9 9

[body-part] [object] [unfixed] [concept]

1 V 2 4 in 3

18

.....

/ End of exmp4.1h*/*

Contents of exmp2.2h

snp npk4 (sentence structure in the form of syntactic units)

1 7 (pointers to the syntactic information)

[general] [place] (semantic-tags)

1 V in the 2 (Target pattern)

7 (corresponding raw example number)

snp npk7nk

7 6

{thing} [general]

1 V 2

8

npk2 snp

1 9

[general] [illness]

1 has/have 2

9

/* End of exmp2.2h*/

Contents of exmp3.2h

snp snp sadj

7 1 33

[day] [relation] []

1 2 V 3

15

npk4 npk7nk snp

7 49 7

[building] [v_participle] [concept]

there V 3 2 in 1

19

.....

/* End of exmp3.2h*/

Contents of exmp1.3h (for imperative sentences):

snp

2 jA 20 go

7

[place]

V 1

12

/* End of exmp1.3h*/

Contents of exmp2.3h:

snp pos_case

2 inwajAra kara 8 wait

20 12

[subject] [nk]

1 V for 2

20

/* End of exmp2.3h*/

Appendix G

Sample Output of Translation from Hindi to English

In the following three pages, sample outputs of translation (output obtained from our system) from Hindi to English are shown. Sentences have been taken from three different sources. First page contains sentences from the book [Sin85] given in the end of the book. Second page presents the translation for a short story taken from a story book. Third page contains the translation of some sentences from a science book [B.P92].

Few sentences from Dr. Suraj Bhan Singh's book

सुधरंग मेरा ड्राईवर है
उंट एक जानवर होता है
मैं इस समिति का अध्यक्ष रहा
यह पुलिस अफसर दिखाई देता है
यह दवा कारगर सिद्ध हुई
सुधरंग घर में है
शादी मामला को है
आज गर्मी है
यह किताब विदेशियों के लिये है
सुधरंग अपने परिवार के साथ है
यह सब झगड़ा आपकी वजह से है
हम आगम से हैं
महिला लाल साड़ी पहने हुए है
सुधरंग को बुझा है
गधा का नौकर चोर लगता है
सुधरंग के पास तास रूपये हैं
गधा के तीन बहन हैं
गधा में प्रतिभा है
सर्मा किताबें विक गइं
पृथ्वी घूम रही है
यह कपड़ा काफी दिन चलेगा
चोर पुलिस से डरता है
सुधरंग पानी पीता है
सुधरंग कार चलाना जानता है
सुधरंग जाने की सोच रहा है
बच्चों ने सोने का बहाना किया
मिस्त्री कार की मरम्मत कर रहा है
ये छात्र अध्यापक से गणित पढ़ते हैं
सुधरंग ने भारी से एक बात कही
डॉक्टर ने मरीज से जन आने को कहा
नौकर से गिलास टूट गया
सुधरंग ने लड़की को यहाँ से गुजरते हुए देखा
नौकरानों लड़की को सुला रही है
सुधरंग मजदूरी से मकान बनवा रहा है

Few sentences from Dr. Suraj Bhan Singh's book

सुध्रार मेरा डाइवर है
उट एक जानवर होता है
मैं इस समिति का अध्यक्ष रहा
वह पुलिस अफसर दिखाई देता है
यह दवा कारगर सिद्ध हुई
सुध्रार घर में है
शादी सोमवार को है
आज गर्मी है
यह किताब विदेशियों के लिये है
सुध्रार अपने परिवार के साथ है
यह सब झगड़ा आपकी वजह से है
हम आगम से है
महिला लाल साड़ी पहने हुए है
सुध्रार को बुखार है
गधा को नोकर चोर लगता है
सुध्रार के पास तीस रुपये हैं
गधा के तीन बच्चे हैं
गधा में प्रतिभा है
मर्मा किताबें विक गई
रुक्मी घूम रही है
पत्र कपड़ा काफी दिन चलेगा
चोर पुलिस से डरता है
सुध्रार पानी पीता है
सुध्रार कार चलाना जानता है
सुध्रार जानें का सोच रहा है
बच्चों ने सोने का बहाना किया
मिस्त्रों का मरम्मत कर रहा है
ये छात्र अध्यापक में गणित पढते हैं
सुध्रार ने भाभी से एक बात कही
डाक्टर ने मर्गज से कल आने को कहा
नोकर से गिलास टूट गया
सुध्रार ने लड़कों को यहाँ से गुजरते हुए देखा
नौकरानी लड़कों को मुला रही है
सुध्रार मजदूरी से मकान बनवा रहा है

Story : A good lesson

सुधीर और राम सगे भाई थे । सुधीर नवी कक्षा में पढ़ता था । राम दसवीं कक्षा में पढ़ता था । दोनों भाई पास के स्कूल में पढ़ते थे । दोनों भाई पढ़ाई में अच्छे थे । सब लोग उनके मां बाप को बहुत भाग्यवान समझा करते थे । लेकिन उनके मां बाप बहुत परेशान थे । क्योंकि दोनों भाई हमेशा लड़ाई किया करते थे । दोनों भाई सुबह से ही लड़ना शुरू कर दिया करते थे । दोनों हर वक्त एक दूसरे की शिकायत किया करते थे । उनकी लड़ाई से पापा भी झल्ला जाया करते थे । पापा आठ बजे दफ्तर जाया करते थे । बच्चे नौ बजे स्कूल जाया करते थे । मम्मा को उन लोगों के जाने के बाद ही राहत मिला करता था । दोपहर को मम्मा उनकी पसंद का खाना बनाकर रखा करता था । लेकिन दोनों भाई घर में घुसने के बाद ही लड़ना शुरू कर दिया करते थे । मम्मा का सारा उत्साह खत्म हो जाया करता था । उस दिन दोनों भाई बहुत खुश थे । उनके नानाजी आए हुए थे । नानाजी उनके लिए कई उपहार लाये थे । दोनों भाई उछल कूद कर रहे थे । फिर दोनों भाई छत पर खेलने चले गये । तभी सुधीर की चीख आई । मम्मा और नानाजी दौड़कर छत पर गये । सुधीर छत पर रो रहा था । सुधीर ने मम्मा को बताया । राम ने मुझे धक्का दिया । मैं छत से नीचे गिर सकता था । मम्मा बहुत दुखा हुई । मम्मा ने नानाजी से दुखी स्वर में कहा । इसलिए मैं घर पर ही रहती हूँ । ये भाई एक दूसरे को कुछ भी कर सकते हैं । रात को मम्मा ने पापा को सब बातें बताईं । नानाजी ने मां बाप को एक तरकीब बताईं । जब दूसरे दिन दोनों भाई खेलकर वापस आये । मां बाप ने एक दूसरे से बहस करना शुरू कर दिया । अब मम्मा और पापा अक्सर लड़ा करते थे । दोनों भाई बहुत दुखा थे । एक दिन मम्मा और पापा ने उनके दोस्तों के सामने लड़ना शुरू कर दिया । दोनों भाइयों को शर्म आई । राम और सुधीर ने नानाजी से बात की । नानाजी ने उनको समझाया । तुम लोग भी हमेशा लड़ते हो । घर में दो लोगों के लड़ने से पूरे घर की शांति खत्म हो जाती है । दोनों भाइयों ने सब कुछ समझ लिया था । दोनों भाइयों ने अपने मां बाप से क्षमा मांगी । इस के बाद दोनों भाइयों ने प्यार से रहना शुरू कर दिया । अब उनके मां बाप खुश थे ।

sudhir and ram were real brother. sudhir was studying in ninth class. ram was studying in tenth class. both brother were studying in nearby school. both brother were good in education. all people used_to understand their parents very lucky. but their parents were very troubled. because both brother used_to fight always. both brother used_to start only to fight since morning. both always used_to do one another's complaints. due to their battle/fight father also used_to get irritated. father used_to go to office at eight o'clock. children used_to go to school at nine o'clock. mother used_to get relieved only after going of those people. at noon mother used_to keep food of their liking after making. but only after entering into house both brother used_to start to fight. mother's whole enthusiasm used_to get over. that day both brother were very happy. their grandfather had come. grandfather had brought many gifts for them. both brother were doing the hopping and skipping. then both brother were gone on the roof for playing. just then sudhir's scream came. mother and grandfather went after running to the roof. sudhir was weeping on the roof. sudhir told mother. ram pushed me. I could/might fall down from roof. mother was very sad. mother told grandfather in sad voice. therefore I stays at house only. these brother can/may do anything to one another. at night mother told to father all matter/thing. grandfather told a/an device to parents. when second day both brother came back after playing. parents started to argue with one another. now mother and father used_to fight often. both brother were very sad. a/an day mother and father started to fight in front of their friends. both brothers felt shy. ram and sudhir talked with grandfather. grandfather explained them. you people also fight always. in a house peace of whole house gets over due to fighting of two people. both brothers had understood everything. both brothers apologized with own parents. after this matter/thing both brothers started to live with love. now their parents were very happy.

Sentences from CSIR Book

वाष्प इंजन जोड़ने और पाना का प्रयोग करता है। वाष्प इंजन इस ऊर्जा से पूरी ग्लेगाडी चलाता है। मानव शरीर भी एक इंजन के समान होता है। वाष्प इंजन अनेक छोटे छोटे कल पुर्जों से बनता है। शरीर भी अनेक अंगों से बनता है। शरीर ऊर्जा के लिए श्वाय पदार्थों का प्रयोग करता है। इस तरह शरीर एक जटिल और कोमल मर्शन है। मुख्य घटक प्रोटॉन और कार्बोहाइड्रेट और लिपिड हैं। भोजन दांतों द्वारा छोटे छोटे टुकड़ों में तोड़ा जाता है। दांतों की संरचना बहुत जटिल होती है। जब हमको बोलने में सहायता देती है। चबाया हुआ भोजन लार में मिलाया जाता है। लार में एक एन्जाइम, एमाइलेस होता है। एन्जाइम जैविक उत्प्रेरक होते हैं। वे जैव रासायनिक क्रियाओं को तेज करने में मदद करते हैं। एमाइलेस स्टार्च को शर्करा में बदल देता है। जब चबाये हुए भोजन को आहारनली में भेज देती है। अब पाचन तंत्र सक्रिय हो जाता है। पाचक रसों में मुख्य रूप से हाइड्रोक्लोरिक अम्ल होता है। यह किसी भी पदार्थ को तोड़ सकता है। एन्जाइम पैप्सिन प्रोटॉन को अमीनो अम्लों में तोड़ देता है। इसे क्षारीय रसों द्वारा उदासीन कर दिया जाता है। म्यूकस की परत द्वारा सुरक्षा प्रदान की जाती है। आंतों की नली वास्तविक वायु उत्पादक होती है। यह एक श्वाय समाधान फैक्टरी है। वसा को वसा अम्ल और ग्लिसरील में परिवर्तित किया जाता है। सेल्युलोज के तंतुओं को अलग कर दिया जाता है। अन्य सब कुछ रक्त अथवा लिम्फ प्रवाह में भेज दिया जाता है। अग्नाशय क्षारीय रस उत्पन्न करता है। यकृत पित्त का योगदान करता है। यह पित्ताशय द्वारा भेजा जाता है। यह वसा के बड़े पिंडों को छोटे जल विन्येय खंडों में तोड़ देता है। यकृत एक नियमनिष्ठ स्राह करता है। यह अतिरिक्त ग्लूकोस को ग्लाइकोजन में परिवर्तित कर देता है। लिवर इसे पवित्र में उपयोग के लिये संचित कर लेता है। ये निस्वार्थ दोस्त हैं। ये अपने पड़ोसियों की सहायता के लिये अपने जीवन की कुरबानी देते हैं। व्यर्थ उत्पाद में भोजन के अपच्य अंश और मृत बैक्टीरिया होते हैं। वे नौ वर्ग मीटर की अवशोषण सतह बनाती हैं। ये सम्पाधित भोजन में अमीनो अम्ल और शर्करा ले लेती हैं। ये उन्हें रक्त प्रवाह में भेज देती हैं। वसा के उत्पाद लिम्फ प्रवाह में भेजे जाते हैं। बड़ी आंत का व्यास अधिक होता है। यह धीरे धीरे पूरा जल अवशोषित कर लेती है। अर्भटोस व्यर्थ उत्पाद मलाशय द्वारा उत्सर्जित कर दिये जाते हैं। धुआं और व्यर्थ गरम पानी, इंजन टवाग फोक दिये जाते हैं।

steam engine makes use of coal and water. steam engine runs whole train with this energy. human body is also like a/an engine. steam engine is made up of many small parts. body is made up of also many organs. body makes use of eatables for energy. in this way body is one complicated and delicate machine. main component are proteins and carbohydrate and lipids. food is broken into small pieces by teeth. structure of teeth is very complicated. tongue gives us help in speaking. chewed food is mixed with saliva. there is a/an enzyme , amylase in saliva. enzyme are biological catalysts. they help biochemical reactions in speeding. amylase changes starch in sugar. tongue sends chewed food in oesophagus. now digestive system becomes active. there is mainly hydrochloric acid in digestive juices. this can/may break any material. enzyme breaks pepsin protein in amino acids. this is neutralized by alkaline juices. protection is given by layer of mucous. tube of intestine is real steam producer. this is a/an food processing factory. fat is changed into fatty acid and glycerol. fibres of selyulosa is separated. everything else is sent in blood or lymph flow. pancreas produces alkaline juice. liver contributes for bile. this is sent by gall bladder. this breaks big globules of fat in small, water soluble pieces. liver is a/an regular store master. this changes extra glucose in glycogen. liver accumulates this for use in future. these are unselfish friend. these sacrifice own life for own host's help. useless product has/have undigested part of food and dead bacteria. they build absorbent surface of nine square meter. these take amino acid and sugar from processed food. these send them in blood flow. product of fat are sent in lymph flow. diameter of big intestine is much. this absorbs whole water slowly. semisolid, useless product are excreted by rectum. smoke and useless, hot water are thrown by engine.