CHAPTER 1 : INTRODUCTION **1**
1.1 PROJECT DESCRIPTION: 1
1.2 MOTIVATION: 2
1.3 APPLICATION OF PROJECT: 4
1.4 PROJECT GOAL: 6
1.5 PROJECT OBJECTIVES: 6
1.6 SCOPE OF PROJECT: 7
1.7 BRIEF REVIEW OF RELATED WORK: 7
1.7.1 Major MT Projects In India: 8
1.8 MACHINE TRANSLATION: 9
1.8.1 Problems of Machine Translation: 9
1.9 OVERVIEW OF MT-METHODS: 13
1.9.1 Approaches to MT: 13
1.9.2 Direct Method: 14
1.9.3 Interlingua Method: 14
1.9.4 Transfer Method: 15
1.9.5 LWG Mapping Method: 16
1.9.6 Statistical Method: 16
1.9.7 Conclusion: 18

CHAPTER 2 : PROJECT PLAN AND SCHEDULING **19**
2.1 PROJECT PLAN: 19

CHAPTER 3 : DESIGN **21**
3.1 PRESENT APPROACH: 21
3.1.1 Advantages of this Approach: 21
3.1.2 Broad Steps: 22
3.2 CONTEXT DIAGRAM: 23
3.3 DATAFLOW DIAGRAM: 24
3.4 DESIGN ISSUES: 25
3.4.1 Analyzer: 25
3.4.2 Lexical Database Design: 25
3.4.3 Issues related to making Lexical Database: 27
3.4.5 Intermediate form Generation: 27
3.4.6 Target Language Sentence Generation: 29
3.5 MAJOR TASKS DESIGN: 30
3.5.1 Prepare Electronic corpora For Hind-Gujarati: 30
3.5.2 Preparation of Bilingual dictionary Hindi-Gujarati: 30
3.5.2.1 Generate Word list from Text Corpora: 31
3.5.2.2 Develop Automatic Dictionary Generation Tool: 31

3.5.3 English-Hindi Dictionary Generation Tool: 34
3.5.4 English-Hindi Direct Translation From Dictionary Lookup: 36
3.5.5 English Corpus Collection: 37
3.5.6 Stop word Remover And Module Separator: 37
3.5.7 Word Attribute Generation and Sentence Analysis Tool: 39
3.5.7.1 Assumptions : 39
3.5.8 Neural Network Design: 45

CHAPTER 4 : IMPLEMENTATION 46

4.1 STRUCTURE CHART: 46
4.2 COLLECTION OF CORPORA: 47
4.2.1 Gujarati Corpus: 47
4.2.2 Hindi Corpus: 47
4.2.3 English Corpus: 48
4.2.4 English-Hindi-Gujarati Parallel Corpus: 48
4.3 WORDLIST GENERATION PROGRAM: 49
4.3.1 Guideline: 49
4.3.2 Algorithm: 51
4.4 DICTIONARY GENERATION TOOL: 54
4.4.1 Class Description: 54
4.4.2 Methods Description: 54
4.4.3 User Screen: 57
4.4.4 Guideline: 57
4.5 STOP WORDS REMOVAL PROGRAM: 58
4.5.1 Class Description: 58
4.5.2 Methods Description: 60
4.5.2.1 Methods of Class CNewdictView: 60
4.5.2.2 Methods of Class CcheckrmsDlg: 61
4.5.3 Guideline: 62
4.5.4 User Screen: 63
4.6 WORD ATTRIBUTES GENERATION AND SENTENCE ANALYZER: 64
Steps for Program: 64
4.6.1 Class Description: 65
4.6.2 Methods Description: 66
4.6.3 User Screen: 67
4.7 PHONEME EXTRACTION: 68
4.7.1 Program Steps: 69

CHAPTER 5 : TESTING 70

5.1 WORDLIST GENERATION PROGRAM FOR HINDI-GUJARATI: 70
5.1.1 Hindi Wordlist: 70

5.1.1.2 Input: 70
5.1.1.2 Output: 71
5.1.2 Gujarati Wordlist: 71
5.1.2.1 Input: 71
5.1.2.2 Output: 71
5.2 DICTIONARY GENERATION TOOL: 71
5.2.1 Input: 72
5.2.2 Output: 72
5.2.3 Result: 72
5.3 STOP WORDS REMOVAL PROGRAM: 72
5.3.1 Input: 73
5.3.2 Output: 73
5.4 WORD ATTRIBUTE GENERATION AND SENTENCE ANALYZER: 73
5.4.1 Input: 73
5.4.2 Output: 73
5.4.3 Result: 74
5.5 PHONEME EXTRACTION: 74
5.5.1 Input: 74
5.5.2 Output: 74
Sample output: 74
5.5.3 Result: 75
CHAPTER 6 : CONCLUSION 76
6.1 CONCLUSION: 76
6.2 FUTURE ENHANCEMENT: 76
APPENDIX
A HINDI-GUJARATI CHARACTER TABLE
B OUTPUT OF WORDLIST GENERATION TOOL
C OUTPUT OF DICTIONARY GENERATION TOOL
D NEURAL NETWORK
REFERENCES

ABSTRACT

KEY WORDS: Machine Translation, Natural Language Processing, Interlingua method, Morphology, Phonology, Neural Network.

Machine Translation is an important technology for localization, and is particularly relevant in a linguistically diverse country like India. Human translation in India is a rich and ancient tradition. The market is largest for translation from English into Indian languages, primarily Hindi. Hence, it is no surprise that a majority of the Indian Machine Translation (MT) systems are for English-Hindi translation. But there is also demand for Indian to Indian language translation. There is lots of research going for South Indian languages. There is great demand for Hindi to Gujarati machine translation also for English to Gujarati translation. As is well known, natural language processing presents many challenges, of which the biggest is the inherent ambiguity of natural language. Machine Translation systems have to deal with ambiguity, and various other Natural Language phenomena. In addition, the linguistic diversity between the source and target language makes MT a bigger challenge.

Here in this project main method for Machine Translation for Indian to Indian language used is combination Interlingua and Statistical method. In this method intermediate code is generated between source and target language. Same method may be applied for English to Indian language translation.

In first phase of project major work done is for collection electronic corpus and dictionary generation using Interlingua method at character level for Hindi to Gujarati. In second phase collection of English corpus and English to Hindi Dictionary database. Stop word removal , module separator and Sentence analyzer using attribute generation from word statistic techniques. Design for neural network for find out meaning full sentences.

ABBREVIATIONS

AI	Artificial Intelligence
FGH-MT	Fully-automatic General-purpose High-quality Machine Translation
IL	Indian Language
LWG	Local Word Grouper
MAT	Machine Aided Translation
MT	Machine Translation
NLP	Natural Language Processing
NLT	Natural Language Translation
UCSG	Universal Clause Structure Grammar
UNL	Universal Networking Language

LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Suggesting the Relation Between Transfer and Interlingua Models..	15
1.2	The transfer architecture for Machine Translation.....	16
3.1	Context Diagram of System.....	23
3.2	General DFD of Machine Translation System.....	24
3.3	DFD of Hindi-Gujarati Dictionary Generation Tool.....	33
3.4	DFD of English-Hindi Generation Tool.....	35
3.5	DFD of English-Hindi Direct Translation Tool.....	37
3.6	DFD of Stop word Removal and Module Separator Tool.....	39
3.7	DFD of Word Attribute Generation and Sentence Analysis Tool.....	44
4.1	Structure Chart of System.....	46
4.2	Flow Chart of Wordlist Generation.....	50

CHAPTER 1 : INTRODUCTION

1.1 PROJECT DESCRIPTION:

Natural Language Translation is technique to use of computers to automate some or all of the process of translating from one language to another. Translation, in its full generality, is a difficult, fascinating, and intensely human endeavor, as rich as any other area of human creativity.

Natural Language Translationrequires a deep and rich understanding of the source language and the input text, and sophisticated, poetic and creative command of the target language. The problem of automatically producing a high-quality translation of an arbitrary text from one language to another is thus far too hard to automate completely. But certain simpler translation tasks can be addressed with current computational model. So here we are taking "Limited Domain", this domain has a limited vocabulary and only a few basic phrase types. Ambiguity is rare, and the senses of ambiguous words are distinct and easily disambiguated based on local context, using word classes and semantic features.

The technology behind developing a Natural Language Translation system is not so simple. A good Natural Language Translationsystem cannot be produced by merely replacing source language words with target language words. A word for word translation does not exactly produce a very satisfying target language text. A good Natural Language Translationsystem must incorporate not only a good knowledge of the vocabulary of both the source and target language, but also of their grammar.

The most common technique to use Natural Language Translation is by coding the grammatical rules of source and target languages in the software and get the translation done using these rules and dictionaries specifically created for this purpose. The other technique is to store the source and target language pairs and try and match the new sentences for similarities from the existing example base and obtain a translation based on the best match. There can be an amalgamation of the above techniques, wherein patterns are stored in place of raw examples. In addition, statistical methods can be deployed to increase efficiency of the translation.

Fully-automatic general-purpose high-quality machine translation systems (FGH-MT) [CP 2.] are extremely difficult to build. In fact, there is no system in the world for any pair of languages, which qualifies to be called FGH-MT. The reasons are not far to seek. Translation is a creative process, which involves interpretation of the given text by the translator. Translation would also vary depending on the audience and the purpose for which it is meant. This would explain the difficulty of building a machine translation system. Since, the machine is not capable of interpreting a general text with sufficient accuracy automatically at present - let alone re-expressing it for a given audience, it fails to perform as FGH-MT. The major difficulty that the machine faces in interpreting a given text is the lack of general world knowledge or common sense knowledge.

Now what is possible bye machine to translate is General purpose MT system with a degree of errors (rough translation), Domain specific and Human-aided (man-machine effort) Machine Translation System.

1.2 MOTIVATION:

No one knows fully understands the meaning of 'Unity in diversity' better than an Indian. Eighteen different languages, plus the countless dialects, and what you have is one messy pottage, which very few would like to put their hands in. While this diversity has been India's distinguishing mark on the global scene, it has created quite a few hiccups in the day to day administration of the country.

In an effort to root out this problem, the founding fathers of our country nominated 'Hindi' as the official language of the country and ordained that all government communication should be made through this medium. However, the situation on ground zero seems to be far from ideal. Quite a chunk of this communication is done in English. Couple this with the fact that most state governments function in their own regional languages and the situation becomes even more complex. This predicament has given rise to an urgent need to translate these documents into a language best understood by the target audience, but with translators increasingly hard to find, what could be the solution to this problem?

In a large multi-lingual society like ours, there is a great demand for translation of documents from one language to another. Most of the state governments work in the respective regional languages where as the union Government's official documents and reports are in bilingual (Hindi/English). In order to have a proper communication there is a need to translate these reports and documents in the respective regional languages. With the limitations of human translators, most of this information (report/document) is missing and not percolating down. A machine assisted translation system or a translators' workstation would increase the efficiency of the human translators. In order to realize a

NLT system, development of domain specific translation systems could be identified as: Government Administrative procedures and formats, Parliamentary Questions and Answers, Pharmaceutical information, Legal terminology and important judgments etc.

Machine translation or Natural Language Translation, say experts, could offer a viable option to those wishing to move on to an environment where thousands of verses in English could be converted into regional languages on the trot. In fact, Machine Translation (MT), thanks to its ability to change the way communication is done, has emerged as one of the most exciting technologies in recent times.

Today, the Ministry of Information Technology has realized the importance of Machine Translation and has identified the following domains for development of domain specific translation systems: government administrative procedures and formats, parliamentary questions and answers, pharmaceutical information and legal terminology and judgments. The ministry also initiated the `Technology for Development of Indian languages' project in the year 1990-91 to support and fund R&D efforts in the area of Information processing in Indian languages covering machine translation among others.[CP 8.]

1.3 APPLICATION OF PROJECT:

- **Education:**

  Machine Translation can be useful for student, who wants to refer book written in other language into another language. Also expert lecture in one language taken for particular field can be translated by Machine Translation.

- **Literature and Entertainment:**

  Storybooks, Novels and other book of literature can be translated from one language to another. Also by using speech-to-speech translation movies and songs of one language can be translated to another language.

- **Communication and Internet:**

  By use of Machine Translation in communication field people can communicate. In Internet Server that can be accessed by anyone on the Internet using a browser. All a user has to do is send the English text and the server sends back the translated text in the language requested.

  Also domain specific translation for chat application can be possible. Here, one can select the language and all the communication will be done in the selected language. This means that even if you select Hindi and the other person selects English, you will receive all messages in Hindi although the other person types in English.

  In Cell phone devices people can communicate through messages like SMS from one langue to another language.

- **News:**

  International and National new agencies published their news in English or other foreign language, which can be translated in Indian language by local newspaper.

So different language newspaper can published same news in their respective language.

- **Government:**

Government organization documents and bill from central government to state or local government office can be translated in local language.

1.4 PROJECT GOAL:

"Natural Language Translation Tools" :

To design and development of tools which is used for conversion of simple sentence of a given domain (1) From one Indian language (Hindi) to another Indian language (Gujarati). (2) From one foreign language (English) to Indian language (Hindi).

1.5 PROJECT OBJECTIVES:

Here major objective of our project is to convert one source language into Natural Language independent form which is intermediate code, after that this intermediate code is converted into target language.

- Breaking down the sentence into the Subgroup according to relationship between words.
- Coding the inherent meaning in Natural Language abstract form as sequence of integer.

- Converting each sequence to subsequence through a small word-cluster in a target language.

1.6 SCOPE OF PROJECT:

The scope of project is to design and develop domain specific corpus based Natural Language Translation Tool. This project uses the statistical language model to modeling user sentences. The main limitation of statistical approach is it behaves on types of data in training corpus. So, user may not get desired output by word prediction and they want to mine the database appropriately.

1.7 BRIEF REVIEW OF RELATED WORK:

The history of MT can be traced starting from the early 50s when it was realized that computers could be used for translation. In the US, a large number of research groups sprang up to work on the task (Russian to English), with funding from defence and intelligence establishments. In the USSR, there was a similar effort to translate from English and French to Russian.[B 2.]

The field revived in the late 70s after the successful completion of the TAUM-METEO system in Canada in 1977. It translated the Canadian weather forecasts from English to French. Around the same time other systems like Titus (English to French for textile technology), CULT(Chinese to English for Mathematics and Physics journals), etc. were also developed.[B 2.]

In 80s, the Japanese successfully completed a national project (Mu)n on MT between English and Japanese. The European Community has also undertaken an ambitious project called Eurotra covering all the languaes of the Community. Work ahs also been undertaken by groups in France, Germany, Switzerland, the US and India.[B 2.]

1.7.1 Major MT Projects In India:

Table 1.1 Summary of major MT projects in India

Project	**Languages**	**Domain/ Main Application**	**Approach**
Anglabharati (IIT-K, ER&DCI-N)	Eng-IL (Hindi)	General (Health)	Transfer
Anusaaraka (IIT-K, UoH)	IL-IL (5IL->Hindi)	General (Children)	LWG mapping
MaTra (NCST)	Eng-IL (Hindi)	General (News)	Transfer
Mantra (CDAC)	Eng-IL (Hindi)	Govt. notifications	Transfer
UCSG MAT (UoH)	Eng-IL (Kannada)	Govt. circulars	Transfer
UNL MT (IIT-B)	Eng/IL (Hindi, Marathi)	General	Interlingua UNL
Tamil Anusaaraka (AU-KBC)	IL-IL (Tamil-Hindi)	General (Children)	LWG mapping
MAT (JadavpurU)	Eng-IL (Hindi)	News Sentences	Transfer

Anuvadak (Super Infosoft)	En g-IL (Hindi)	General	N/A
StatMT (IBM)	Eng-IL	General	Statistical

1.8 MACHINE TRANSLATION:

1.8.1 Problems of Machine Translation:

The main problem faced in the area of MT [B 1.] is syntactical. With various grammatical issues involved in the languages-since each language has disparate structures-it is difficult to capture these differences. However, MT has over the years made notable progress and has been quite successful in scientific and domain specific fields because of its objectivity.

What makes the MT task so difficult?

1. ***Ambiguities :*** Language is highly ambiguous. Ambiguities are present at two different levels. (i) word level and (ii) sentence level.

Word level ambiguity can be

Word level ambiguities can be

(a) Content words can be ambiguous:

For example Head word e.g. "RUN" in English have different meaning as Verb and as Noun in Hindi in dictionary. [English-Hindi Dict].

(b) Function words can also be ambiguous:

For example - Preposition 'in'

(a) I met him in the garden

mEM usase baglce meM milA
(b) I met him in the morning
mEM usase subaha 0 milA

2. ***Syntactic difference in concerned language:***

For example: English and Hindi have different syntactic structures

A few examples

(a) word order :

- English has a more or less fixed word order.
- Hindi and most other Indian languages have a relatively free word order.
- Therefore, the way the two languages encode information is different

  - Hindi uses lexicalized vibhaktis (ko) for encoding object relation

  - English uses 'position' in the sentence for doing so

(b) Yes-no question sentences :

'Yes-No' question formation in English and Hindi

* English has a 'positioned' auxiliary

(no separate 'question word')

Example :
(a) Is he coming ?

* Hindi has a 'question word' kyA

Example :
(b) kyA vaha A rahA hE ?
Therefore,
(b1) vaha A rahA hE kyA ?
is also possible

(c) post nominal modification

(i) ing clauses

(ii) pepositional clauses

(d) 'un-' negative clauses

For Example :
English : Unless you reach there the job will not be done
Hindi : jaba taka tuma vahAM pahuMcate nahIM ho, kAma nahIM hogA

It was believed at one time that all that was required for MT was a bilingual dictionary and rules for reordering words in a sentence. For example, to translate the following sentence from Hindi to English:

Raama	*ne*	*kheta*	*jotaa.*
(noun)		*(noun)*	*(verb)*
Ram	*ploughed*	*the*	*field*
(n)	*(v)*		*(n)*

The Hindi words are replaced by English equivalents and reordered to follow the sequence non-verb-no instead of noun-noun-verb. Unfortunately, this naïve method fails to work generally.

A number of difficult problems come in the way, like word sense selection, choice of sentence structure, pronoun reference, noun-noun modification, conjunctions like 'and' and 'or', identification of tense and modality, etc. *We will see some of these below:*

Each word has many different meanings or senses. Selection of the appropriate sense is necessary for translation. For example, consider another sentence in Hindi in which *'jotaa'* has a different sense than *'plough'*:

Raama ne gaDii jotti
Ram prepared the cart.
OR
Raama ne ghoDaa jotaa
Ram harnessed the horse.

Thus *'jotaa'* can be replaced by *'plough'*, *'prepare'* or *'harness'* depending on the sense implied in the sentence. The correct sense must first be identified for each of the words before selecting the appropriate replacement.

The sentence structure must also be interpreted correctly for translation. Fro example, in

I saw Ramesh on the hill with the telescope.

The telescope could have been the instument of seeing, or Ramesh could have been carrying the telescope, or it is the hill with the telescope. The three translation are different in Hindi:

Meine durbiina dwaaraa Ramesh ko pahaaDii par dekhaa.
Meine Ramesh ko duurbiina ke saath pahhaDii par dekhaa.
Meine Ramesh ko duurbiina walli pahaaDii par dekhaa.

Hence, it would be important to identify the relationship of the telescope correctly. Such identifications might require following a paragraph and maintaining a context.

Frequently, it is important to find the referent of the pronoun. Consider the following sentence for example:

A dog saw a cow on the road.
It started barking on seeing it.

The first 'it' can refer to a dog, cow or road. If one were translating into Hindi, the gender of the verb would depend on the gender of the referent:

Vaha use dekhakara bhounkane lagaa.

If '*vaha*' was referring to the cow or the road, feminine would be used.

It may sound ridiculous to even consider that the road can bark, or it may seem obvious that it is the dog which barks. However, reaching such conclusions requires more than just word replacement ability.

Human beings interpret and infer the right sense using their

(1) Linguistic knowledge
(2) World knowledge
(3) Domain knowledge.

Providing linguistic knowledge to the machine is possible, though requires lot of effort, but Providing world knowledge is not possible in near future.

The problem described above point to a common theme: A sentence must be "understood" before it can be translated. Natural language understanding is a hard task because it requires formulating not only a grammar for the language but also using background knowledge including common sense knowledge. All this must be used in complex and as yet unknown ways to process a given text.

1.9 OVERVIEW OF MT-METHODS:

1.9.1 Approaches to MT:

Rule-Based Approach: In this approach linguistic rules are used to map the two languages such as:
- Phrase structure rules
- Disambiguation rules

Knowledge-Based: This approach uses extensive knowledge of the domain, concepts in the language and ability to reason.

Example-Based: In this approach mapping is based on stored example translations.

Statistical Approach: This approach tries to do away with explicit formulation of linguistic knowledge and tries to develop rules based on probabilities

Hybrid Approach: This approach is mixes one or more above Approach.

1.9.2 Direct Method:

MT system are designed for a particular pair of language. If translation capability is needed between another pair of language, a new system must be constructed. Because of its dependence on the language, the system has the advantage that special features and similarities between the concerned language can be made use of.

In this method, one is to use the robust parsing techniques, which sometimes amount to translating by fragments. Another is to give up an producing elaborate structural analyses at all, and just do simple operation that can be done reliably. In this method MT system should do as little work as possible. Typically such systems are built with only one language pair in mind, and the only processing done is that needed to get from one specific source language to one specific target language.[B 2.]

1.9.3 Interlingua Method:

A sentence in a source language is first analyzed and is represented in an intermediate language. The intermediate language need not be a human "natural" language and is typically a formal "mathematical" language. Next a generator takes the intermediate representation and generates a sentence in the target language.

Advantage:

The analyzer of parser for the source language is independent of the generator for the target language. So, as a result if one wants to build a system with translation capability among say 15 language, only 15 parser and generator need to be constructed. Instead of 210(15*14) system that need to be constructed in first Direct Approach.

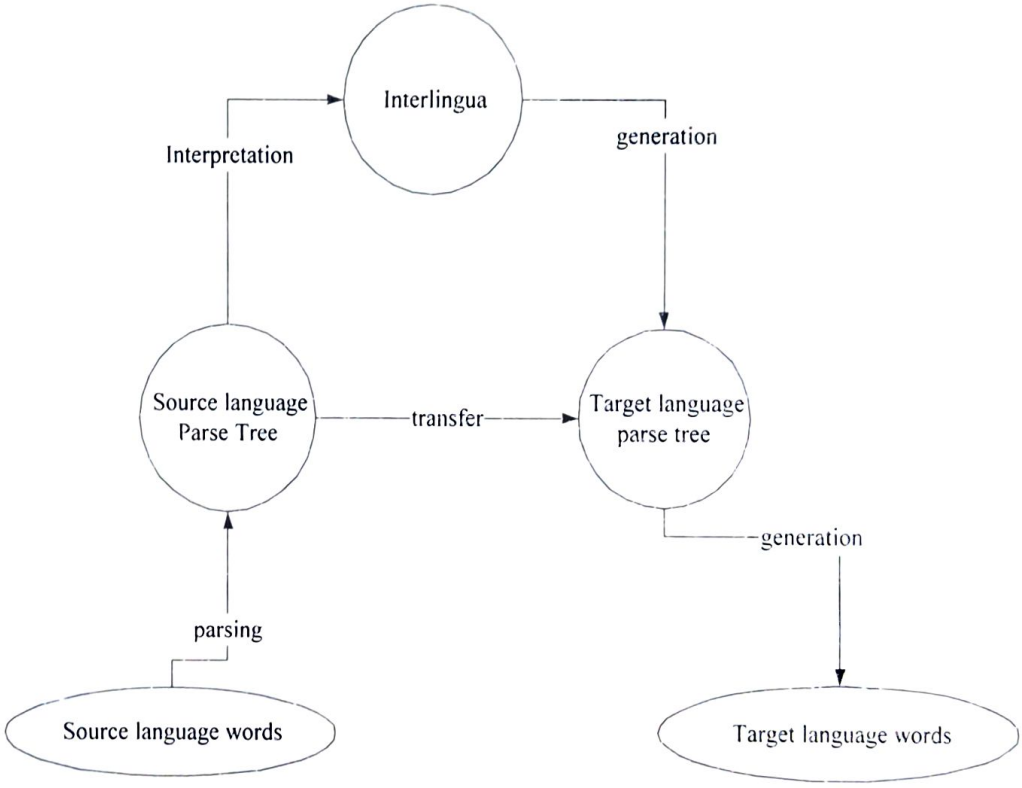


Fig 1.1 Diagram Suggesting the Relation Between the Transfer and Interlingua Models.

1.9.4 Transfer Method:

The parser produces the representation for the target language. The generator takes over from here. This approach is intermediate between first two.

For 15 language in this approach only 15 parser and generator needed however, the no of component needed are 210.

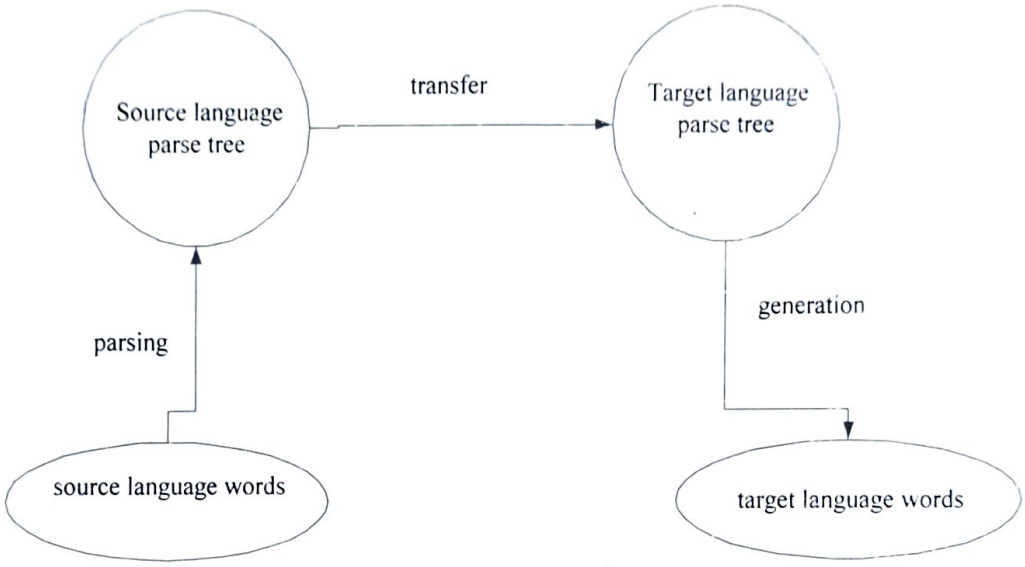


Fig 1.2 The transfer architacture for Machine Translation

1.9.5 LWG Mapping Method:

LWG means local word grouper, the function of this block is to form the word group on the basis of the 'local information' (i.e. information based on adjacent words).only those groupings are done by LWG, however, which will need no revision later on. In this approach LWG of source language is mapped to the target language local word splitter (LWS). LWS splits the local word groups into elements consisting of root and features.

1.9.6 Statistical Method:

In this approach the problem of translation: to focus on the result, not the process. There is a problem arise not only for culture-specific concepts, but whenever one language uses a metaphor, a construction, a word, or a tense without an exact parallel in the other language.[B 2.]

So, true translation, which is both faithful to the source language and natural as an utterance in the target language, is sometimes impossible. If you are going to go ahead and produce a translation anyway, you have to compromise. This is exactly what translators do in practice, they produce translations that do tolerably well on both criteria.

This provides us with a hint for how to do MT. We can model the goal of translation as the production of an output that maximizes some value function that represents the importance of both faithfulness and fluency. If we chose the product of fluency and faithfulness as our quality metric, we can formalize the translation problem as:

Best-translation T= argmax fluency(T) faithfulness(T,S)
Where T- Target sentence, and S- source sentence,
Best-translation T= argmax P(T) P(S/T)

To implement this, we need to do three things:

Quantify fluency, P(T), Quantify faithfulness, P(S/T) and create an algorithm to find the sentence that maximizes the product of these two things.

In the transfer, interlingua, and direct models, each step of the process made some adjustment to the input sentence to make it closer to a fluent TL sentence, while obeying the constraint of not changing the meaning too much. In those models the process is

fixed, in that there is not flexibility to trade-off a modicum of faithfulness for a smidgeon of naturalness, or conversely, based on the specific input sentence at hand. This new model called the Statistical model of translation.

1.9.7 Conclusion:

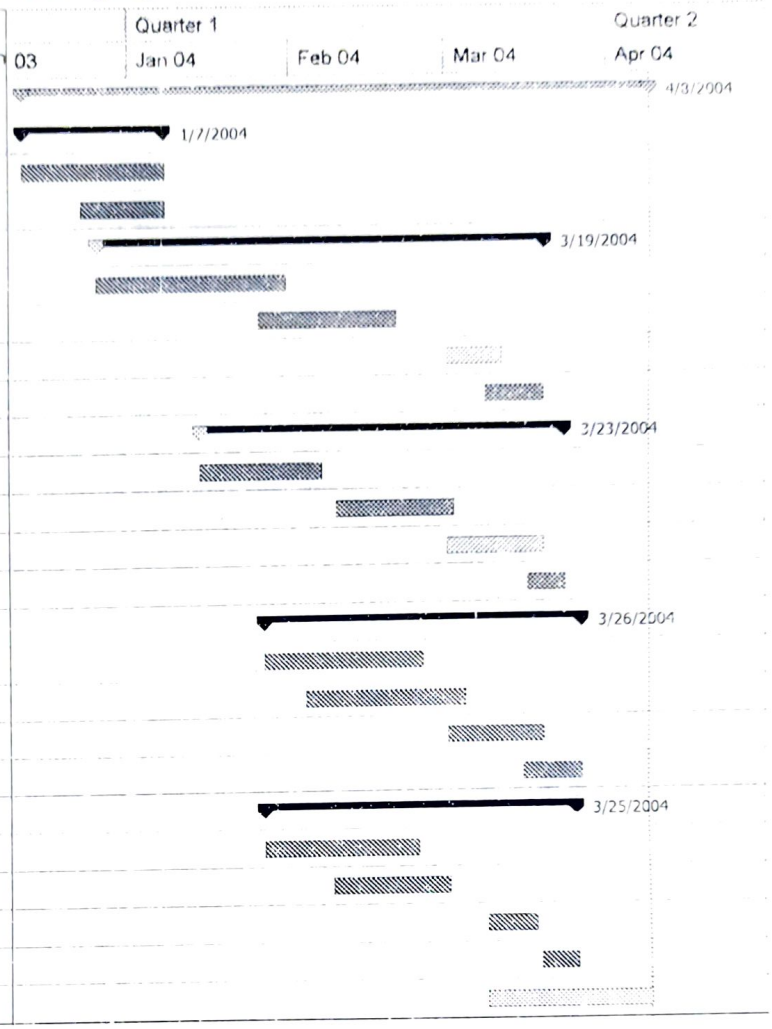
In practice, of course, working MT systems tend to be combinations of the direct, transfer, and interlingua methods. But of course syntactic processing is not an all-or-nothing thing. Even if the system does not do a full parse, it can adorn its input with various useful syntactic information, such as part-of-speech tags, segmentation into clauses or phrases, dependency links, and bracketing. Man systems that are often characterized as direct translation systems also adopt various techniques generally associated with the transfer and interlingua approaches.

Here for Indian languages translation Interlingua approach is more suitable. But we can use combination of Interlingua, Direct and Statistical for more distinct languages like English and Hindi, rather than only single method. So for advantage of more than one we have to use combination of above methods at different stages of MT.

Start: 12/12/2003

NLT : Natural Language Translation Tools

	Activity Name	Orig Duration
1	**Natural Language Translati**	**85**
2	**project requirement Study & (**	**19**
3	Data Collection	19
4	Dictionary collection	12
5	**Design**	**61**
6	Wordlist Generation	26
7	Dictionary Generation	20
8	Stop Word Removal and Moc	5
9	Word Attribute and Sentence	9
10	**Implimentation**	**49**
11	Wordlist Generation	17
12	Dictionary Generation	16
13	Stop Word Removal and Moc	5
14	Word Attribute and Sentence	5
15	**Testing**	**44**
16	Wordlist Generation	22
17	Dictionary Generation	22
18	Stop Word Removal and Moc	14
19	Word Attribute and Sentence	9
20	**debugging**	**43**
21	Wordlist Generation	21
22	Dictionary Generation	16
23	Stop Word Removal and Moc	7
24	Word Attribute and Sentence	5
25	Documentation	5



Activity — Resource Names & %Alloc
Subproject — Name — Name — Early Finish
Event — Early Start — Name
Interface Event — Early Start

Cum. Original Profile — Cum. Remaining Profile — Non-Cum. Actual Profile
Cum. Act. +Rem. Profile — Non-Cum. Original Profile — Non-Cum. Remaining Profile

Filtered by <All Objects> and Sorted by <None>

CHAPTER 3 : DESIGN

3.1 PRESENT APPROACH:

Indian languages that are very close especially Gujarati and Hindi, so here Interlingua approach is profitable. Because most of Indian languages are derived from Sanskrit.

In this project for One Indian language to another Indian language translation, we choose the "Interlingua Approach"[CP 1., CP 2.] for MT.

Here the task of building an MT System is subdivided into two parts:

1. The first module does language-based analysis: It takes all the information in the source text and presents it in its output, in an intermediate language, which is natural language independent form.

2. The second module may do domain specific knowledge based processing, statistical processing, etc. in which it may utilize world knowledge, frequency information, concordances, etc. to produce output in the target language.

3.1.1 Advantages of this Approach:

1. *Early availability:* The first module is much easier to build than the second, requiring an order of magnitude less effort. Therefore, the system becomes available for use at an early date. However, it must be pointed out that a certain amount of training would be needed for the user, to read the output in the intermediate language.

2. *Early Feedback:* Since the system can be used at an early date, not only does it serve a useful purpose, it also becomes easier to build the second module. The early feedback guides the refinement and building of the system .

3. *Robustness:* The second module by its very nature will be fragile. The first module will be much more rebust. Therefore, the system provides a robust layer which can be used even if the second module fails in any specific text.

4. *Clear task demarcation:* By separating the two modules based on clear cut criteria (of language knowledge and background knowledge), it becomes a lot easier to coordinate the activities of system building. The builders of each module

understand the limits within which they have to operate, therefore, it becomes possible to engage in a large coordinated team effort.

5. *Evolution of interlingua:* The output language would have some constructions of the source language, those which cannot be transferred to any existing construction in the target language (and hence requires training as mentioned earlier). By using this approach with many source and target languages, an interlingua would evolve naturally. Thus, the intermediate language no longer needs to be designed beforehand, rather it evolves naturally.

6. The second module becomes easier to build once the first module is ready because a person working on the second module does not need to know two languages (source and target). He needs to know only one, namely, the intermediate language, which is close to the target language.

3.1.2 Broad Steps:

Here one source language is converted to natural language independent form which is intermediate code, after that this intermediate code is converted into target language.

(1) Breaking down the sentence into the Subgroup according to relationship between words.

(2) Coding the inherent meaning in Natural Language abstract form as sequence of integer.

(3) Converting each sequence to subsequence through a small word-cluster in a target language.

Here, problem for Hindi to Gujarati Translation can be solved stepwise i.e. first word to word translation and second sentence to sentence translation, third limited domain paragraph is translated.

3.2 CONTEXT DIAGRAM:

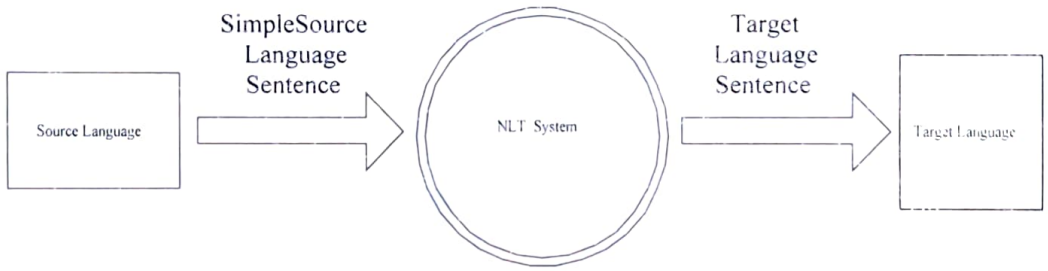


Fig 3.1 Context Diagram of System

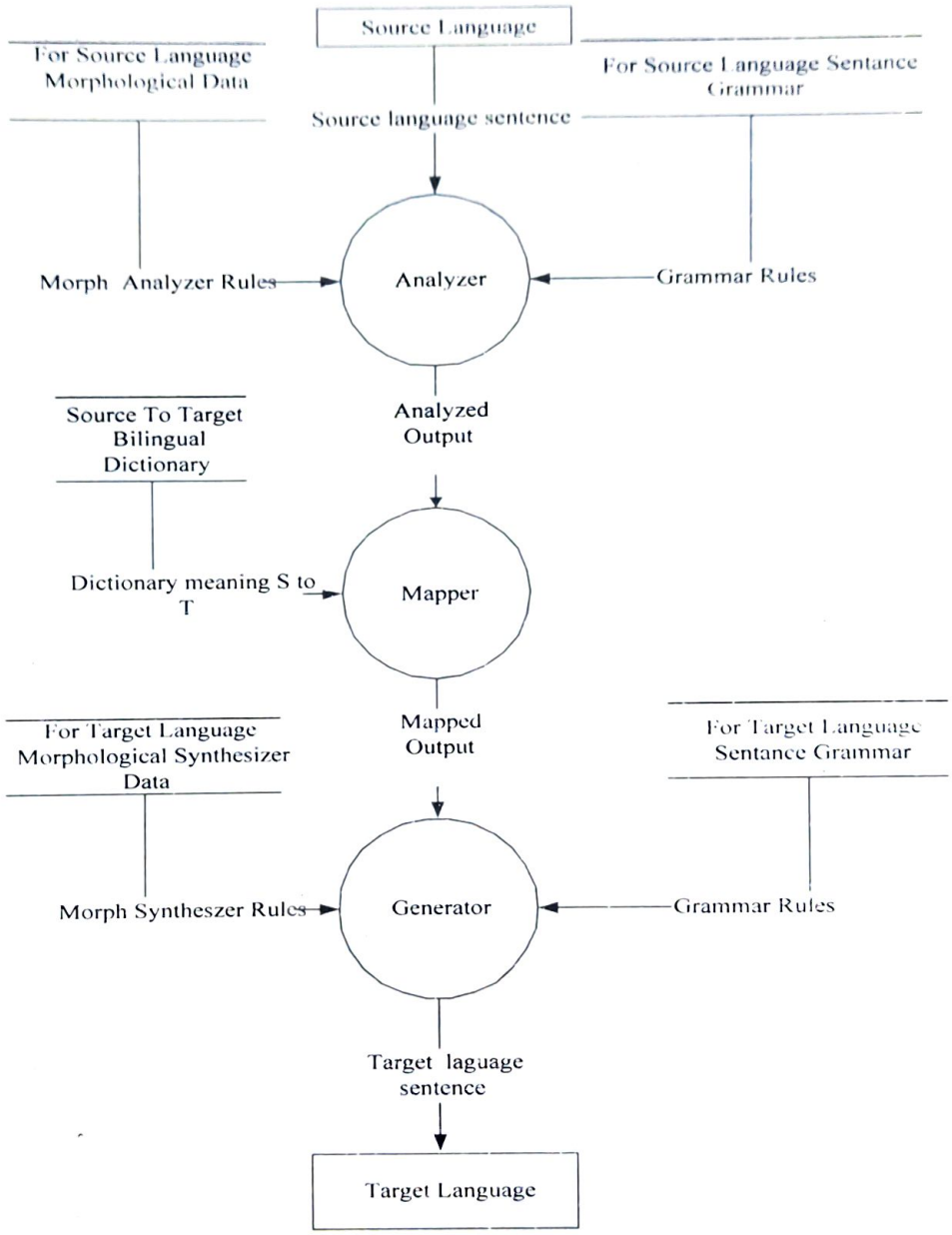


Fig 3.2 Flow Diagram of Machine Translation System

3.3 DATAFLOW DIAGRAM:

3.4 DESIGN ISSUES:

Figure Shows Architecture of machine translation system. This Design is general for IL to IL and English to IL. Some design issues are described below to understand above design. The system has three main processing phases: Analyzer, Mapper and Generator.

3.4.1 Analyzer:

Here analyzer is morphological analyzer, which takes as input the Source Language sentence, reads each word, identifies the root word, and retrieves the necessary information from the lexical database about that word. Identification of different kinds of phrasal(group of words when used together exhibit entirely different meaning from their original meaning) is also done during morphological analysis. If a particular word is not found in lexical database, system transliterates the word and also tries to guess the expected syntactic category of the word. An on-line lexical database is created, as we proceed with the process of analysis of the input text. Here, store information about all the words encountered in the text, and the Analyzer examines the entries of this lexical database before searching in the large multi-lingual database. As a large number of words get repeated in a typical text, the on-line lexical database helps to reduce the search time to a very large extent.

3.4.2 Lexical Database Design:

A lexical database is like an electronic dictionary (a collection of words along with the information about its different syntactic categories, its derivatives and uses under different possible contexts) with some additional information needed for the translation

system or for some other NLP application. This multi-lingual lexical database keeps most of the syntactic, semantic and other information needed for disambiguation, only for the source language. It has multiple fields for each of the target language when each field has TL meaning and some TL specific information needed by the target language text generator. To resolve multiple target, meanings, certain patterns, semantic tags and some constraints for disambiguation are also stored.

Table 3.1 Dictionary Entry

Lexicon	Syntactic info	Source language meaning	Verb pattern	Semantic tags	Target language meaning

The above table shows the general format of entry in Dictionary.

In the above figure, Lexicon represents the root word of Source language. Syntactic information means category information of the word along with other information like number, person, gender etc. Each verb in any language exhibits certain expectations and according to that possible syntactic patterns are assigned to each verb. These patterns help us in selecting the correct meaning of the verb while usage.

Semantic tags try to represent the behaviour and perception of the object represented by the lexicon. The list of semantic tags should be designed keeping in mind the application of machine translation. The semantic tree is for nouns only and these nouns tags are assigned to the expected subject and object of the verb.

like structure for the source language and corresponding target language structures applicable to a group of Indian Language. A rule is constructed by substituting constant parts of the sentence to the variable parts as far as possible if the right hand side depicting the pseudo target code remains the same.

Information about the words of the input sentence is used to form patterns and these patterns are matched to the left-hand side of the rules stored in the rule-base. Top down approach is used for matching the rules. System recursively tries to unify with all the possible patterns and on finding the match, the corresponding rules is invoked, and the right-hand side of the rule yields the pseudo target. System tries all the possible combinations by back tracking and retains only the successful ones and due to that unification of multiple rules is also possible. In such a case, more than one pseudo target is not truly in the language used in interlingua in the sense that no attempt has been made here to develop a meaning representation.

A sample typical rule in the rule-base is as follows:

[noun - phrase]
[verb- phrase] ⟶ *[target-noun-phrase]*
[prep- phrase] *[target-prep- phrase]*
[target-verb-phrase].

These are non-terminal symbols and these are further defined in terms of terminal or non-terminal symbols. For example:

Basic-noun-phrase = noun/determiner noun/adjective noun.....

And similarly prep-phrase and verb-phrase are defined.

After the identification of appropriate pattern/patterns for an input sentence, pseudo code generator generates the intermediate form according to the structure of the target

language for all the parsing. Sense disambiguator makes use of semantic knowledge and verb patterns supplied by the morphological analyzer to discard some of the parsing and some of the meanings reducing number of translations. But, sense disambiguator is able to disambiguate only partially due to incomplete and inconsistent knowledge.

From the above examples, we can see that the same word is translated differently in target language depending upon the sentence type. In some cases meaning resolution is possible while in other cases it shows both meanings of word. Multiple translations are obtained mainly due to several valid rules being fired.

3.4.6 Target Language Sentence Generation:

Intermediate version generated by the rule-base is synthesized and target language translation is generated by the target language text generator. For each target language, we will have to have a separate text generator which will make use of language specific rules to supply correct post or pre positions in the sentence and to generate the correct form of words according to the information provided by the intermediate form. Each text generator takes care of the peculiarities of its language. A number of grammatical rules of the target language in the form of expectations and constraints are used to select the appropriate case markers, affixes etc. A Paninian framework [B 1.]is used for this purpose. The ambiguities, which still remain unresolved, are taken care by human post-editing.

3.5 MAJOR TASKS DESIGN:

3.5.1 Prepare Electronic corpora For Hind-Gujarati:

The development of corpora of texts in machine readable form was envisaged as a basic research facility for linguists and computer scientists. Accordingly, the primary objective of the project was to put together collection of machine readable texts in all the constitutionally recognized Indian languages. Simultaneously development of software tools for word level tagging of grammatical categories, word count, frequency count ,spell checkers etc. was also envisaged.

Preparation of electronic corpora is very challenging task for Natural Language Processing specially for Indian language because there is not enough electronic or computerized text available for IL. Electronic corpora is row material or test material for machine translation.

3.5.2 Preparation of Bilingual dictionary Hindi-Gujarati:

Second major task for Indian language is preparing Bilingual dictionary for Source to Target language. Here source language is Hindi and target language is Gujarati. From online resource I can not find ready available dictionary for Hindi to Gujarati. So I have to create dictionary manually.

For that two major task to be done:

1. Generate Word list from Electronic Text Corpora.
2. Develop Automatic Dictionary Generation Tools and Data Entry for it.

3.5.2.1 Generate Word list from Text Corpora:

In this section generated Sorted Wordlist from collected Text corpora.

- In first step we have to collect corpora. Here corpora mean plain text in Hindi and Gujarati. Parallel corpora mean same text having in both languages. If not possible to collect parallel corpora then collect different corpora. Corpora collection is important and time-consuming task in Machine Translation.

- In second step develop an algorithm for separation of sentence from given text paragraph it is done by EOS (end of sentence) symbol for particular language and other delimiter for particular language. These sentences are stored in separate database or file.

- In third step from above sentences words are separated, removing delimiter from sentences for particular language does this. Here delimiters are word separator like coma, semicolon, and other punctuation symbols.

- After that in forth step remove ambiguity in between same words and remove duplicate words from wordlist. In this step same words repeated in wordlist then we have to remove duplicate by comparison, also some ambiguity like words are internally different but we see as similar word i.e. due to font problem and writing mechanism in Indian script which is not unique. So, we have to remove those ambiguities from wordlist.

- In fifth step sort wordlist according to Indian script, taking natural sequence of Indian script i.e. kakko and other symbol of it i.e. barakhshri does this. Also removal of ambiguity present in symbol of barakshari due to font or Indian script problem.

3.5.2.2 Develop Automatic Dictionary Generation Tool:

This tool is for generating bilingual Dictionary automatically; here algorithm is based on phonetic of words. All Indian languages are derived from Sanskrit, so phonetically they are nearer words. Especially Hindi and Gujarati are more nearer phonetically.

So we can take benefit of this to reduce search time for given Hindi word to Gujarati words. For this we are using Hamming Distance between Hindi and Gujarati word.

Hamming Distance is the distance between two words in same language, but here Hindi and Gujarati are different language. To overcome this problem I used the Indian Script as Intermediate code. Conversion of Hindi into Intermediate code; also convert Gujarati word into Intermediate code. Here intermediate code is Indian Script (Kakko-Barkhadi) because it is same for both Hindi and Gujarati. Thus we are using Hamming Distance by natural sequence of Indian Script.

In this section Developing Data Entry tool for Hindi to Gujarati Dictionary. Here Steps for developing Dictionary Generation Tool:

- First of all take Sorted Hindi and Gujarati wordlist file as input file.
- Select One Hindi word from list to enter in wordlist.
- Search nearest hamming distance word from Gujarati Wordlist.
- Select appropriate Gujarati word from nearest hamming distance wordlist.
- Save both Hindi-Gujarati words in Bilingual Dictionary database.

***Design For Dictionary Generation*:**

- In First step find the hamming distance of selected Hindi words with other Gujarati words. Here hamming distance is integer number, which represent number of character in natural sequence according to Indian script match with given Hindi to Gujarati words. After finding hamming distance list nearer words of Gujarati for selected Hindi word.

- In Second step after finding hamming distance between words, words having zero distance should automatically stored for particular Hindi word. The database is updated for both Hindi and Gujarati word.

- In Third step word having nearer distance are listed and Gujarati word for given Hindi word is selected and stored in database.

- In Forth step if according to above hamming distance technique word is not found, means that Gujarati word is not phonetically similar to Hindi so manual

searching according to Indian Script is done by selecting first Indian script character of Gujarati word. After that Hindi-Gujarati words are stored in database.

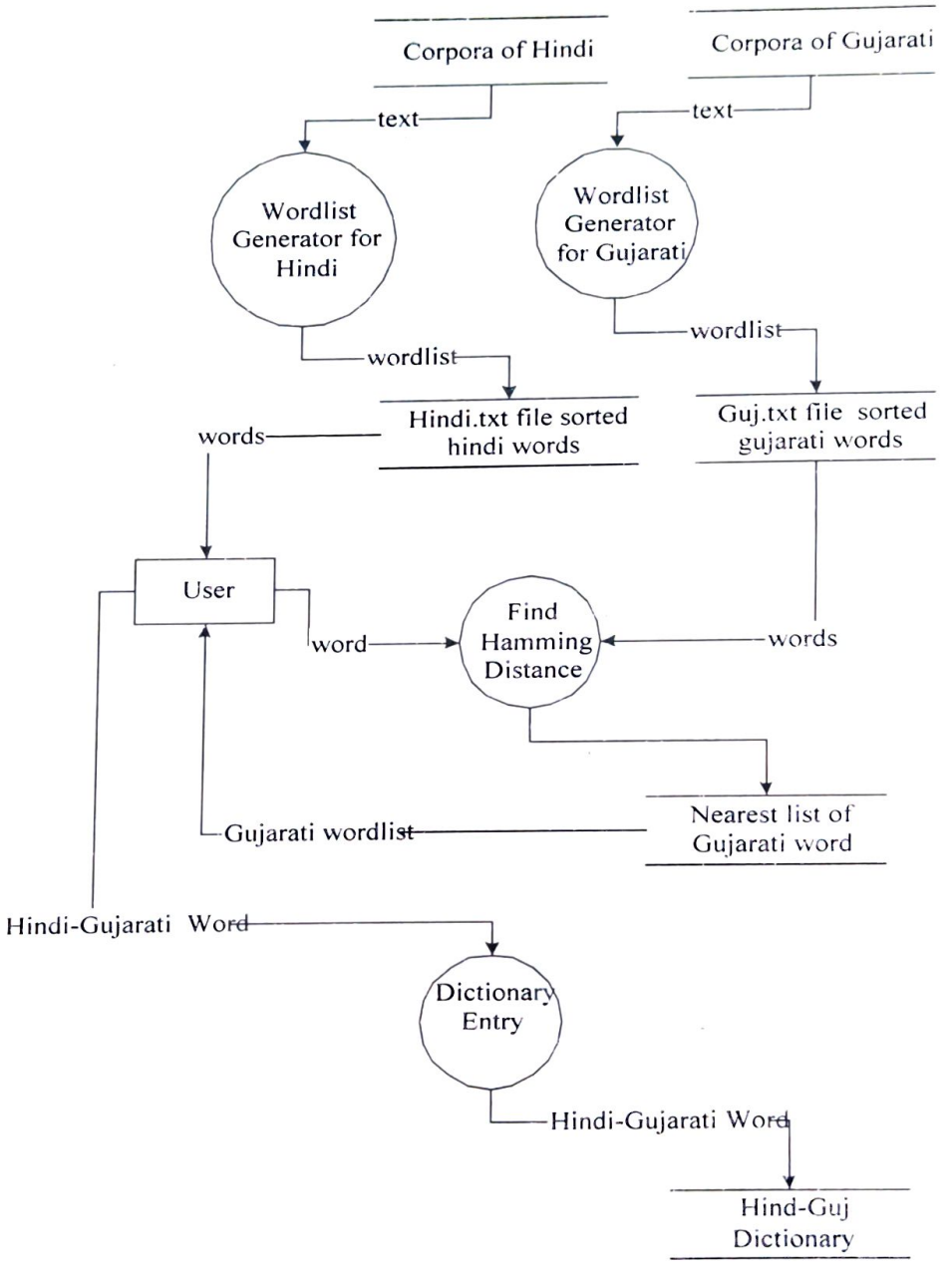


Fig 3.3 Flow Diagram of Dictionary Generation Tool

3.5.3 English-Hindi Dictionary Generation Tool:

Bilingual dictionary is basic requirement of any MT system. For English-Hindi there are more than one dictionary available on internet. But dictionary provided by IIT, Bombay is more useful and bigger in size which contains more than 25000 words meaning. This dictionary also provide example sentence in English according meaning of Hindi for that English word.

This tool is for generating bilingual Dictionary for English to Hindi language. Here the source dictionary is downloaded from IIT Bombay's website, link is www.iitb.ac.in. The dictionary name is *'Shabdanjali'* which have specific format of data. This dictionary is in the format of *.isci format that is ISCII character encoding system. File size is 2.38MB.

Now, this dictionary format is not supported by normal Windows environment for that, we have to convert this file in to *.html. For conversion ISCII plug-in software from IITB is used. After that HTML file is converted to simple Text file. This text file can now be readable by machine for programming purpose in the system. Text file size is very large, so searching the word in is very difficult and time consuming. There is also difficulty in searching retrieving of word meaning due to format of file. So we have to convert this text file into MS-Access database in table format, which made it more convenient for use.

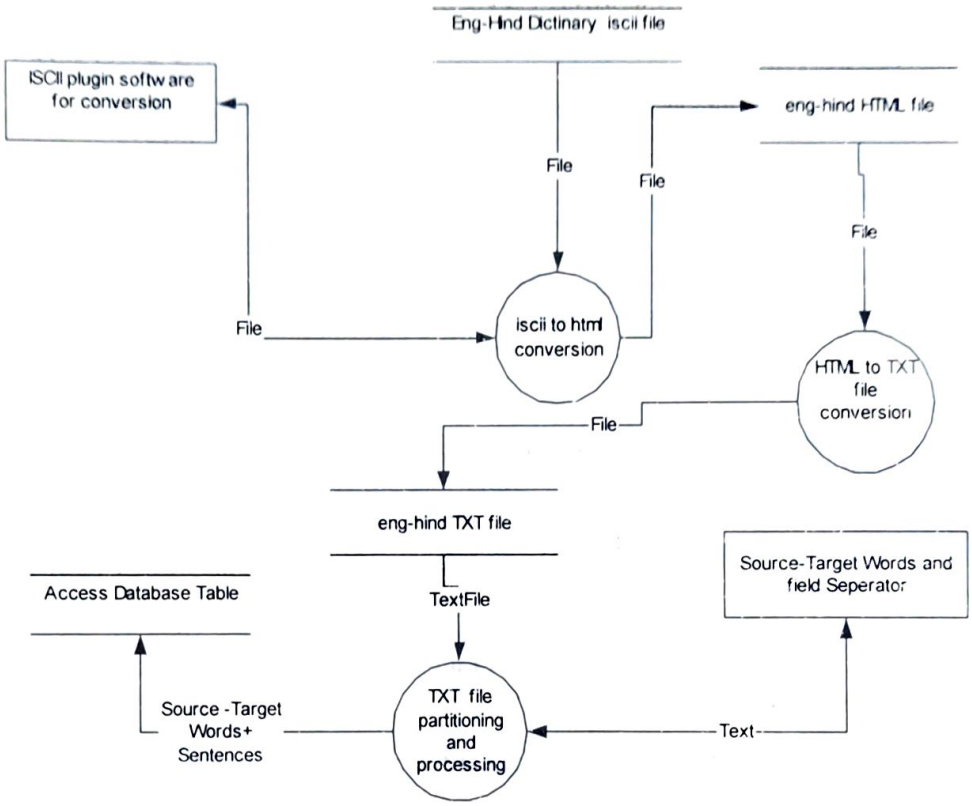


Fig 3.4 Data Flow Diagram of Eng-Hindi Dictionary Generation

Sample format of English-Hindi dictionary in text format file.

The dictionary should have a precise format

(a) every piece of information should be stated in a separate field.

Type of information for an entry in the dictionary

* headword
* grammatical category
* gender (if required)
* number of senses
* an example sentence for every given sense

The above information should be stated in a standard format

"run","N","1.δ>Pα"
he goes for a run every morning.
--"2.sEra"
They have taken out their van for a run around the city for sight seeing.

"run","V","1.xODZanA"
She cannot run fast because her leg pains.
He was the first person to run a mile in four minutes.
--"2.xODZAnA"
She ran her eyes over the stage.

The entries should not only be in a standard format but the way the information is given should also be consistent and format should be strictly followed. Here try to keep the first sense as the default sense

3.5.4 English-Hindi Direct Translation From Dictionary Lookup:

This tool is for direct word to word translation of English to Hindi from dictionary lookup from database. English-Hindi dictionary which is transform in to database having following fields : English word, Hindi meaning word1, Corresponding English example sentence.

Here first the English sentence is taken from dictionary. After that number words are separated out from this sentence and by dictionary searching for that particular English word, corresponding Hindi meaning is taken out. By using this Hindi words and user intervention target Hindi sentence is generated. So by this method we can find percentage accuracy of direct translation.

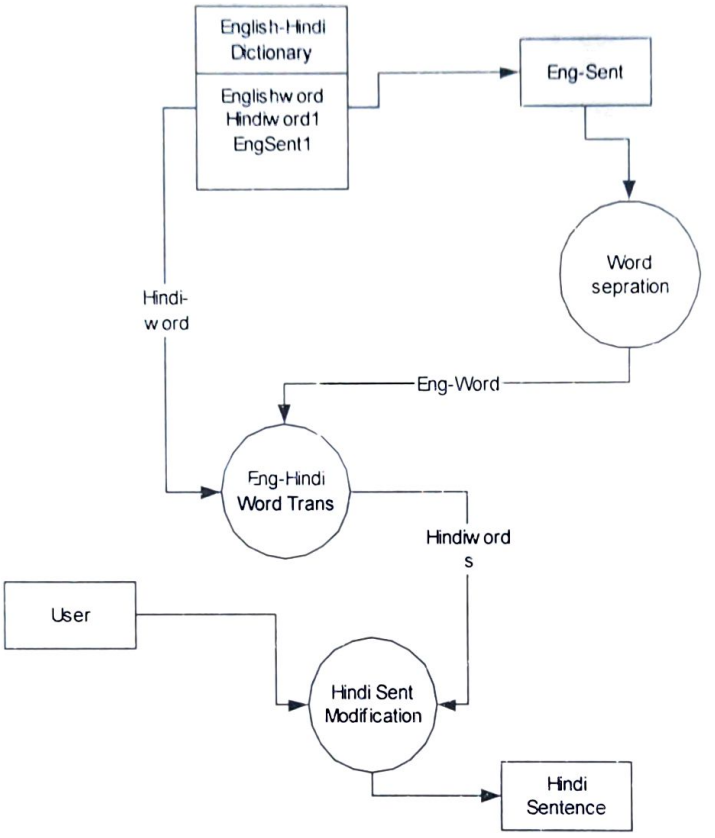


Fig 3.5 DFD for Eng-Hindi Direct Translation

3.5.5 English Corpus Collection:

English corpus is very necessary for machine translation for English to Hindi. For this huge size of English text is needed. This text is machine readable in windows environment and also contained from different types of domain.

3.5.6 Stop word Remover And Module Separator:

English corpus is very necessary for machine translation for English to Hindi. Stop words are words in language which has less meaning or less controlling words. That means if we remove that word from sentence then also original sentence have no more affect on its meaning. So in NLP and Knowledge based system such words are normally remove for convenience for knowledge extraction.

Here in this module stop words removal technique is used by algorithm, in which there is separate list of stop words. By comparing it with current sentences of corpora and if match found then remove that word from sentence. And those output sentences are stored in separate file.

After removal of stop words from original sentence, we have to separate sentences which are simple. Here for MT system is for simple sentences only for that need of simple sentences separation is necessary. For this development of module which separates three word sentences, four word sentences and five word sentences is necessary. For each type of sentence there is separate file in which either whole sentence or seek address of original sentence file is stored.

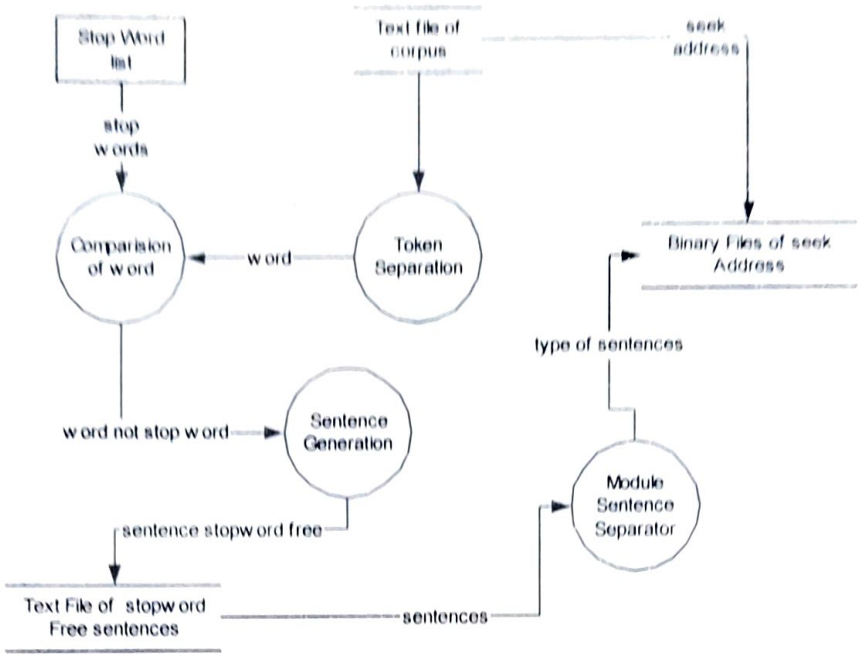


Fig 3.6 DFD for Stop word Removal

3.5.7 Word Attribute Generation and Sentence Analysis Tool:

3.5.7.1 Assumptions :

Our proposed algorithm is based on the assumption of attributes of word. Each word have a lots of attributes related with it.Suppose "cat" is a word then attributes related with it are: noun, types of word, quantity, emotion level (i.e.: sad, joy, etc.), singular or plural, etc.

Now problem is that can we use this information of word to generate the sentence? For this we try to understand "Law of meaning full sentence":

"Sequence of words has a unique meaning, which generates the unique meaningfully sentence".

For example:

Meaning full Sentence : "Here is your Pizza".
Now, if we can change these sequences of the words then generated sentence may be ambiguous or non-ambiguous.
Possible sentences are : (with change sequences)
"is Here your Pizza".
"your Pizza is Here".
"Pizza is Here your".
.
.
.(continue)

We can see that some of the sentence convey the meaning to the user and some are not.

So, we can say that if unique sequence of words ,then only generates the meaning full sentence.

If we assigned some integer number to each attribute of word and see the continuity of the integer number then we can see that meaning full sentence try to formulate continuity of these numbers.

"Here	*is*	*your*	*Pizza".*
100	*119*	*125*	*137*

We can say that sequence of word generates some kind of continuity in integer number and

try to maintain each time when new sentence will be generated.

From above example we can see that if user wants generates new sentence then only their considering parameter is continuity of these numbers. Also, to generate the new sentence move from one word to other word in the knowledge database it required minimum energy to get the next word for generation of meaning full sentence.

Some other possible attributes of words:

- *Gradiation.*

The possible value of word meaning may be, is it indicating positive or negative or may be time dependent.i.e. sentence : "now , it is time to evaluate."
In following sentence if we remove "time" as a redundant term, then instead of it possible term may be "student" or "future". So, we must find the immediate word to generate the meaning full sentence. So ,we must find other possible word instead of the following word with highest probability, to generate the sentence.

- *Relative position of word in sentence.*
  Most of time some particular words may come at particular position in the sentences. e.g. is ,it, own, am, was, etc.So, may use it as a attribute of word in sentence .

- *Class of words.*
  e.g. "I am going."
  "He was going".
  Both "am" and "was" fall in the same class.
  So, we can take this class parameter as possible attribute.

- *Emotion.*
  Some of word has a special attribute taken as emotion, which uses as a highest probability of a particular word at that position.
  e.g. In some cases if all possible word of sentence try to convey the message that hole sentence is with full of joy or sorrow then next probable word in the sentence with relative matching emotion vector.

Here in this module attribute of words in given sentence is generated. Here attributes do not have fix number but fix property. Attribute generation is iterative process which randomly initialized and after that it changes its value. Attribute calculation is a statistical process and will meet very large number of natural stop-word free sentences. Here natural sentences are human sentences not artificial sentence but meaning full sentences only.

There are two property of attributes :

1. ***Positional Properties:*** Here by this property the relative position of the word with respect to sentence. If it is fixed than translation property is set to zero. And if it is varying then property differs with degree up to 100.

2. ***Associative Properties:*** Associative property of word describes co-occurrence with other words. This property is high when co-occurrence with words with other properties is high. E.g. Noun →Verb here noun followed by verb rule has very high probability so here in this case Associative property is high.

Here initially the range of attribute sets are distributes from 0 to 99, all at average distance 50 to 70. If particular set is co-occurring with another set you increase or decrease by small amount, so that attribute sets come nearer to each other.

Suppose initially there are 15000 words and range of attribute is (0-99) 100 and there are 5 attribute assigned to each words.

Average distance should be approximately = (Range)^(no of attributes)/total words.

So, initially each word getting each attribute with clear distance between attributes.

Words having identical behaviors then the attribute distances between them should reduced. This process of learning will either increase or decrease value of attributes.

Repeatedly this process reduces total number of attributes by merging process, because attribute sets are club together with other and make one set. After some time it reaches equilibrium or steady state with optimal number of attribute set. This attribute sets will be close to Natural language attributes of languages.

Thus attribute generation is iterative learning process, which can be done by Neural network mapping. Here there are 5 attributes and range is 100, so for three word sentences 3*(100)^5 possibilities which is very large but there are sentences which are meaningful sentence is less than total. So now we have two sets one meaningful and meaningless. After that those words from dictionary (wordlist) which are use in meaningful sentences are taken out and make one set. From this set possible artificial sentences are made by using algorithm. After that 1 million of meaningful sentences and whole set of artificial sentences are given to Neural Network for training. So by using Multilayer Perceptron algorithm of classification, machine can identify new unknown sentence as meaningful or not.

A Multilayer Perceptron model of Neural Network is used to map sentence type. A Network has fixed word sentence attribute as input and N different types of outputs. Which representing :

1. *Meaningless (Nonsense) sentence*
2. *Interrogative Sentence*
3. *Exclamatory Sentence*
4. *Affirmative Sentence*
5. *Question Sentence*

A sentences are initially evaluated by human reader in appropriate class and stored in a database. This database is used for training the Neural Network.

A use of above classification identifies a language structure which could be appropriately adopted in the target language. Also this method is very crude semantic analysis. Same technique could be used for further grouping using Neural Network.(Future Work).

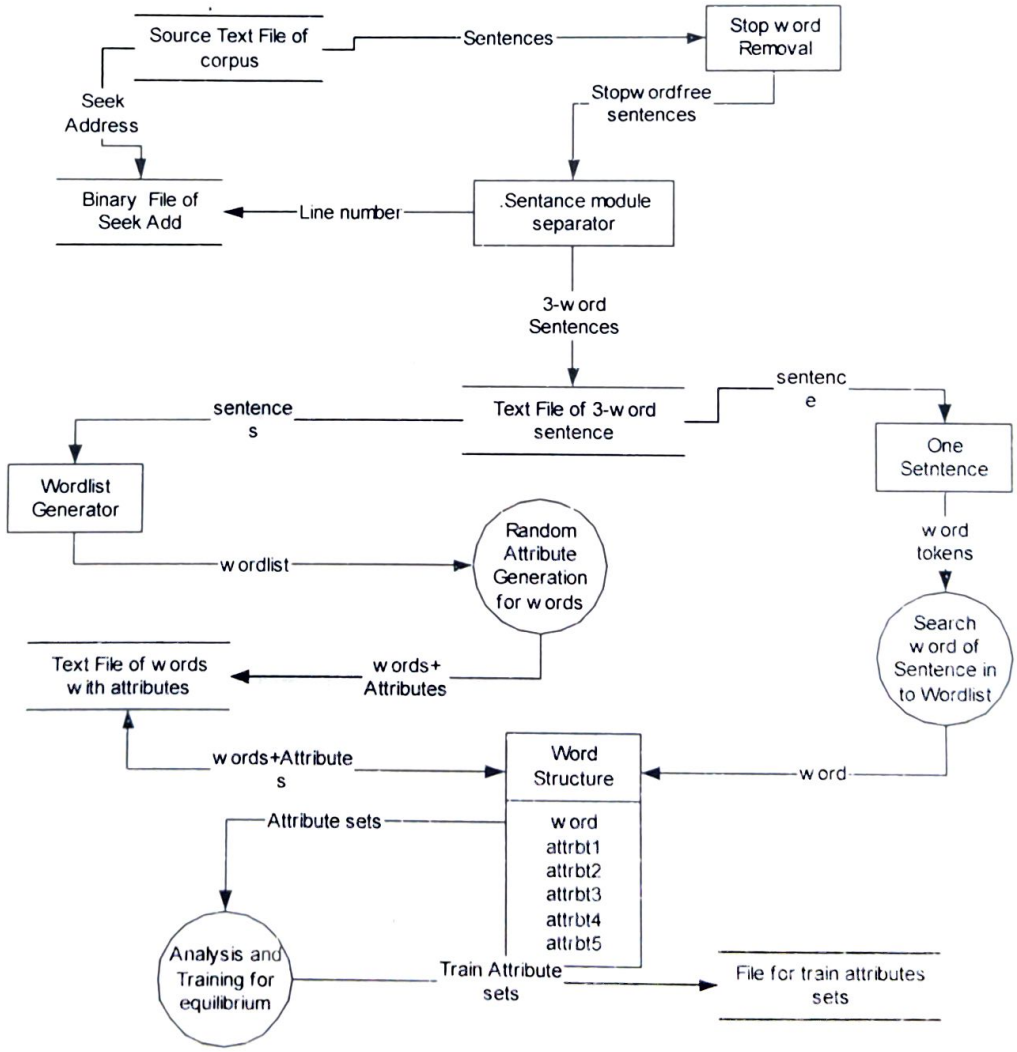


Fig 3.7 DFD for Word Attribute Generation

3.5.8 Neural Network Design:

Here to solve the above problem of using Neural Network. There are two possible approach of Neural Network Application.

1. Taking help from human subject to classify the exemplar data and train the Network. If examplar data as underlined feature separating the class, the Network will learn this (hopefully) and will work with the same rule to analyze new data. Here Multilayer perceptron is used to classification (Appendix).

2. Second method is to naturally divide the sentences in a hypothetical space in N number of group. Membership of the group is based on hypothetical parameter (attribute), which are derived from the usage of the word in natural sentence. Here Hamming distance is taken for attributes, Hamming distance is minimum in one group. The type of Neural Network are called Self learning feature map, one of the most popular model (SOM) is Kohenen's Network (Appendix).

CHAPTER 4 : IMPLEMENTATION

4.1 STRUCTURE CHART:

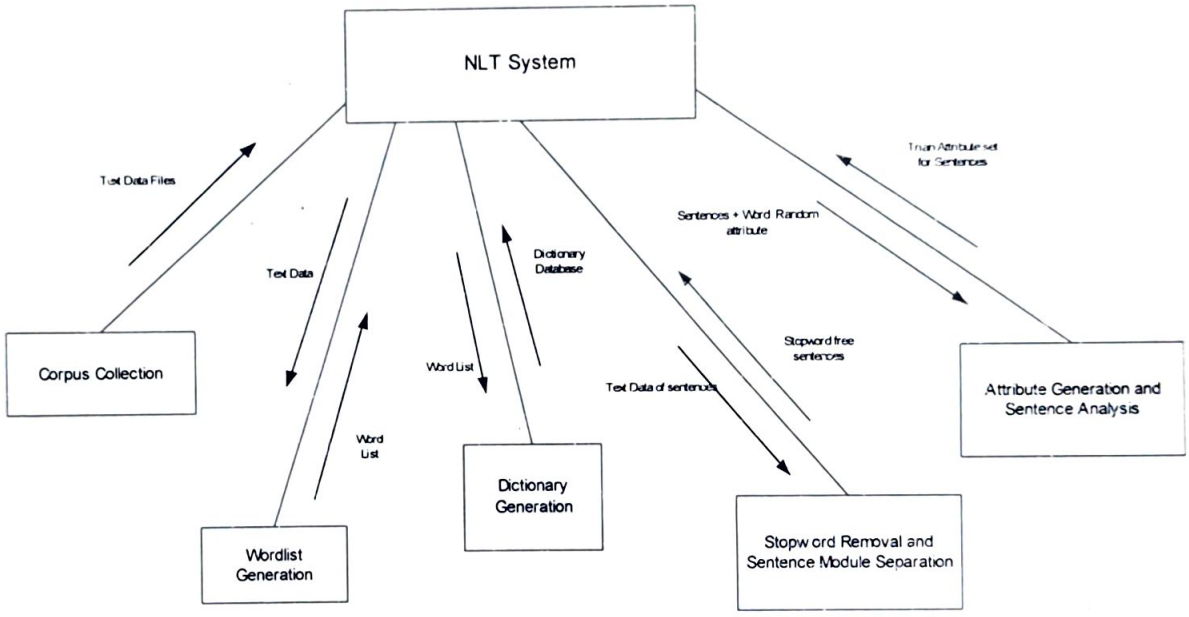


Fig 4.1 Structure Chart for NLT System

4.2 COLLECTION OF CORPORA:

4.2.1 Gujarati Corpus:

Here I collect Gujarati machine readable corpora from following sources :

From Gujarati Newspaper website, from website of Gujarati literature online.

Problem to process literature corpora was that, there is not enough material available. Our Requirement is to collect 10MB pure text of Gujarati but not same font same type of text available on literature website So I choose second option i.e. Gujarati News paper site. From *.HTML files of gujarati news from Gujarat Samachar and converted these files into *.TEXT files.This Text files are merge in Single Text file up to 10MB size.

Final Source for Gujarati Corpus: 'www.gujaratsamachar.com'
Font use: 'Gopika' TTF font

4.2.2 Hindi Corpus:

For Hindi machine readable corpora from Government website of India, Hindi literature website and from Hindi News paper website. I was collected 12 MB of text of Hindi from "Amar Ujala" News paper, previous article available from Archive.

First I had downloaded *.HTML text pages from website, then convert this HTML pages into pure *.TXT file i.e. pure text. Merge all files into single TXT file of 10MB.

Final Source of Hindi Corpus: 'www.amarujala.com'.
Font : 'AU' TTF font.

4.2.3 English Corpus:

For English language large corpus is need for analysis of sentences and attribute generation and training. For English from different domain of text is taken here major sources are as follows:

1. *"Project Gutenburg Corpus"*, which provides most of domain text. Project Gutenberg provide all type of E-Book. Here taken simple stories of English literatures. URL :www.gutenberg.net and total size of text corpus is 52 MB.

2. English-French parallel corpus from which only English taken, Domain is Debates of the Senate (Hansard) which Session of Parliament at UK. Format is *.e files simple text, total size of text is 20MB training corpus and 10MB testing corpus.

3. European Parliament corpus in *.al format total size is 59MB text.

4. www.mastertext.com provides different text includes literature, here domain taken is children adventure story E-book in *.html format, which converted to simple txt format, total size of text is 3MB.

4.2.4 English-Hindi-Gujarati Parallel Corpus:

English-Hindi-Gujarati language parallel corpus is need for analysis of sentences and training and dictionary generation. Parallel corpus is same text in different language Main source is in English which is translated into Hindi and Gujarati. Text collected from the EMILLE project, May 2000, *Dept of Linguistics, Lancaster University, Lancaster, UK.*

Domain taken is health, sociology, finance, housing and legal. Average Size for each language is 2.8 MB and encoding of character set is Universal Multiple-Octet Coded Character Set (UCS), not supported by Win-98 but by conversion program converted in to WX-notation provided by *IIIT, Hyderabad.* WX-notation is Romanization or transliteration of Indian languages in English.

4.3 WORDLIST GENERATION PROGRAM:

In this module developed in VC++ input taken as Hindi ,Gujarati or English Corpora of text, which is processed by this algorithm and outputs wordlist from this text. This program gives sorted wordlist according to Particular Language Script.

4.3.1 Guideline:

- Split the 10MB files of Hindi text file into 500Kb small files by software File Splitter of Partridge Software.
- Take one by one 500Kb files as input to Wordlist Generation program and store output file word.txt. Rename output file word.txt to serial number sequence as postfix to 'word' i.e. word1.txt, word2.txt ...and so on.
- Merge all files word1.txt, word2.txt.... in a single output file dic.txt by using Prof. H.S. Mazumdar package of 'Dictionary maker' in which give 'Merge' option by checkbox.
- After merging all files in to dic.txt .Run again program of Wordlist Generation by input as dic.txt because in 'Dictionary maker' sorting not according to Indian Script and also duplication and ambiguous word removal option is not included.
- Word.txt file of this program is final output file of this entire wordlist generation process.

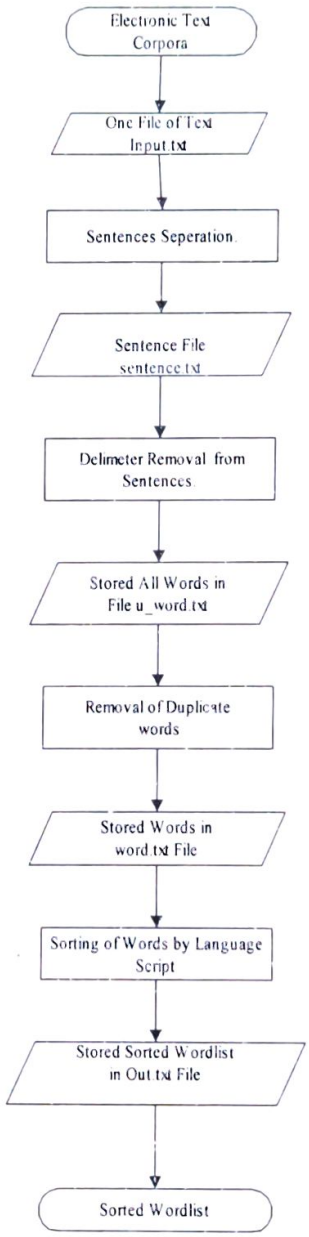


Fig 4.2 Flow Chart of Wordlist Generation

4.3.2 Algorithm:

```
Algo:  WordlistGen()
{
Readsent();
Extractword();
Sortword();
}

Readsent()
{
        char ch,prvch
        int count=0
fp1=fopen("text.txt" in read mode)
fp2=fopen("sentence.txt" in write mode)

        ch=read one character from fp1
        Write character ch into fp2
        do
        {
                prvch=ch;
                ch=read one character from fp1
                if(ch==End Of File)
                        break;
                //Removes the Dilimiters
if(ch==',' || ch==':' || ch==';' || ch=='\"' || ch=='.'|| ch=='\\' ||ch== '/'|| ch=='?' || ch=='!'|| ch==
\"
||ch== '-' || ch== '#'||ch == '@'||ch == '$'||ch == '%'||ch == '^'||ch == '&'||ch == '*'||ch ==
'('||ch = ')'||ch == '{'||ch == '}'||ch == '['||ch == ']' ||ch=='Đ'||ch == '|'     ||ch == '<'||ch ==
||ch=='0'||ch=='1'||ch=='2'||ch=='3'||ch=='4'||ch=='5'||ch=='6'||ch=='7'||ch=='8'||ch=='9')

        ch= read one character from fp1
        if(prvch==' ' && ch==' ')
                ch= read one character from fp1

                if(ch==new line){
                        Write ch='.' Into fp2
                        Count=count+1
                }
                if(ch!=End Of File)
                        Write ch='.' Into fp2
        }while(ch!=End Of File);

        Write ch='.' Into fp2
```

```
        Close File fp1
        Close File fp2
        Print " Number of sentence Generated = count"
        return 0;
}
ExtractWords()
{
FILE *fp,*fp1;
char st[2000];
int i=0,j,flag=0,n
char *strings[50000]
fp=fopen("sent.txt" in read mode)
fp1=fopen("u_word.txt" in write mode)

        while(!feof(fp))
        {
                st = Read string  from fp
                n=strlen(st);
                if(st[n-1]=='.')
                        st[n-1]=NULL
                for(j=0 to i){
                        //Removes the duplicates words
                        if(strcmpi(strings[j],st)==0)   {
                                flag=1;
                                free(st);
                                break;  }
                        }
                if(flag==0) {
                        strcpy(strings[j],st);
                        Write strings[j] in fp1
                        free(st);
                        i=i+1
                         }
           flag=0;
        }
        close File fp1
        close File fp
        return 0;}

SortWords()
{       int len,nlines;
        char *p,word[SIZE];
        char *wordptr[MAX];
        nlines=0;
fp1=fopen("u_word.txt" in read mode)
fp2=fopen("word.txt" in write mode)
```

```
while(feof(fp1) == 0) {
                word= Read string from fp1
                len = Lengthof(word)+1;
                strcpy(p,word);
                wordptr[nlines] = p;
                nlines++;
        }
        File close fp1
        //call the sort Function
        Print " Number of words read =" nlines
        qsort(wordptr,0,nlines-1);
        Print "Number words read  After Qsort= " nlines
        for(i=0 to nlines)
        Write   wordptr[i]  into fp2
        File close fp2
        return 0;
}
qsort(char *v[],int l,int r)
{
        int i,la
        if(l >= r) return;
        swap(v ,l ,(l+r)/2);
        la =l;
        for(i=l+1;i<= r;i++)
                if(string comp(v[i],v[l]) <0)
                        swap(v ,++la,i);
        swap(v,l,la);
        qsort(v,l,la-1);
        qsort(v,la+1,r);
}

swap(char *v[],int i,int j)
{
        char *temp;
        temp = v[i];
        v[i] = v[j];
        v[j] = temp;
}
```

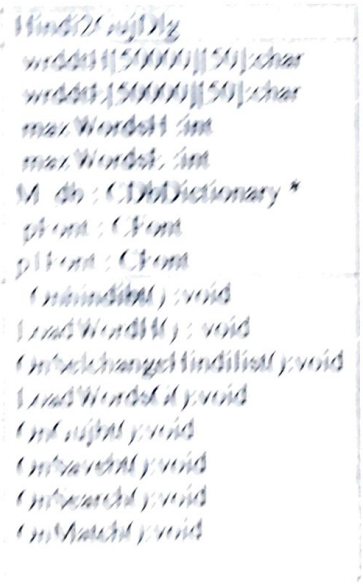
4.4 DICTIONARY GENERATION TOOL:

This program is developed in VC++ and database in MS-Access. This program is for partially automatic dictionary generation from Hindi to Gujarati and manually Data entry tool for Hindi to Gujarati wordlist generated by above wordlist program.

Input:
Hindi wordlist file hindi.txt, Gujarati wordlist file gujarati.txt, Gujarati Script file gu.txt
Output:
Hindi to Gujarati dictionary file dic.html and dictionary.mdb MS-Access file.

4.4.1 Class Description:

CLASS Hindi2Gujdlg



4.4.2 Methods Description:

OnHindibt():

This method objective is to open selected file by user and after that loads words from that file into list box after setting the list box font as "AU". It calls LoadWordH() method.

LoadWordH():

This method objective is to loads all words in to list box from file open by above OnHindibt().

OnGujbt():

This method objective is to open selected file by user and after that loads words from that file into list box after setting the list box font as "Gopika". It calls LoadWordG() method.

LoadWordG():

This method objective is to loads all words in to list box from file open by above OnGujbt().

Onmatch():

This method is for matching the nearest word from selected Hindi word to corresponding Gujarati word in list box. Here in this method there are two array of Hindi and Gujarati script character set is taken in ASCII character for "AU" and "Gopika" font in h[] and g[] character array respectively. After that phoneme extraction for Hindi word is taken place, which is stored in temp number sequence. This number sequence is matched with corresponding Gujarati words phoneme number sequence. When exact hamming distance

match then that word is displayed in List box. If exact distance is not zero then word with minimum distance is displayed in Gujarati list box. Here the threshold value from level list box is taken for taking limit of minimum distance for matching word.

OnSearch():

By this method the searching of Gujarati words which start with selected Gujarati later list box is taken out and displayed in Gujarati Word list box. Here the searching is taken place in word array of Gujarati word list.

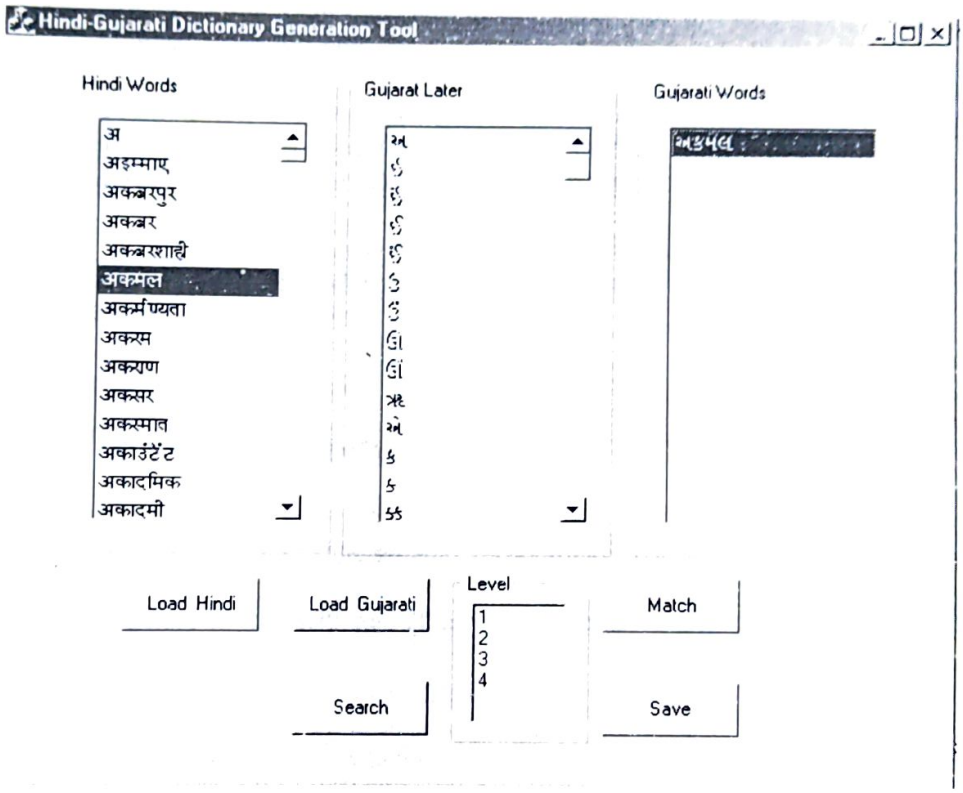
OnSelchangeHindilist():

This method is used to select the level for threshold for matching Hindi to Gujarati word. When in level list box the number is change then this method passed the number to the corresponding Onmatch() method.

OnSavebt():

By this method the selected Hindi and Gujarati words from respective list box is stored in HTML file as well as MS-Access database table. The html file name is "dict.html" and database name is dict.mdb in which table stored Index, Hindi word and Gujarati word.

4.4.3 User Screen:



4.4.4 Guideline:

- First of all in this package click 'Load Hindi' button to load hindi.txt file in 'Hindi Word' list box.
- After that click 'Load Guj' button to load Gujarati script in 'Gujarati Word' list box. Here in background Gujarati.txt file for Gujarati wordlist is load in memory.
- Select one of Hindi word from 'Hindi Word' which you want to store in dictionary.
- Click 'Match' button for matched nearest hamming distance word from Gujarati word list. If the exact similar word is there in Gujarati list then it displayed first in Gujarati list box.
- If Gujarati word is not found by simple 'Match' click, then increment threshold from list box named 'Level'. And then click 'Match ' button.

- After try for all threshold from 'Level' list the wanted Gujarati word not found then select First later of Gujarati word from Gujarati character list box. Click 'Search' button, so that all Gujarati words start with that later is displayed in 'Gujarati Word' list box.
- If the selected Hindi word's Gujarati meaning is not phonetically nearer, then directly select first character of Gujarati word from Gujarati later list box. Click on 'Search' button and select Gujarati word from 'Gujarati Word' list box.
- Finally for storing these Hindi-Gujarati words in dictionary click on 'Save' button. This stores both words in dic.html file and table of dictionary.mdb database of MS-Access.

4.5 STOP WORDS REMOVAL PROGRAM:

Stop words removal and module separator is combine tool for language which takes input as large corpus of English with more than one file and give output as single separated files for each type of sentence. Here type of sentences is number of word per sentence, i.e. 3-word sentences, 4-word sentences etc.

Input:English language corpus text files of different domain and source.

Output:Stop word free sentences files for each type of sentences.

4.5.1 Class Description:

Class CNewdictview:

Objective of this class is for Separate out the Stop words free sentences from given source text File. This class have following structure of Attributes and Methods.

CNewdictview
as:int DataSt:CNewdictSet mnextf:int St: CString
Onsentgen(): void Onrmsword():void Onmodulesep():void OnFileopen():void OnCheckrms():void Onbin():void

CLASS CheckrmsDlg:

Objective of this class is for dynamically removing stop words and displayed it on screen. By next and previous button user can see the sentence one by one with and without stop words for particular type of sentence. Here type of sentence is number of word per sentences after removal or stop word. So by this class the meaning full and meaning less sentence can be identify and stored in binary file.

CcheckrmsDlg
fp:FILE * sw:CStringArray word:CStringArray log: unsigned int wrd[15][50]: char words: int ln: int resul: int s[2000]: char pos3: long token: char * line: char *
Onnext_crms():void Onmeaning():void Onprevbtn():void

4.5.2 Methods Description:

4.5.2.1 Methods of Class CNewdictView:

Onsentgen():

This method objective is for opening all files given by OnFileopen() method one by one and combined these selected file into one single merge file named "sent.txt". In this file all sentences are separated by removing delimiters. After that each sentences are written at different new lines. So there is a one sentences at each line in this file.

Onrmsword():

This method objective is Removing all stop words from given input file "sent.txt" and write stop word free sentences into "o.txt" file. This method takes one sentences from input file separates token from it. Compare each token with stop word list in array and if stop word present in token remove that token from sentence.

Here not only text file for type of sentence is created but Binary file is also created for each type of sentence. Where dynamically the seek address of source file "sent.txt" is stored for respective sentences in particular type of sentences file. So by this the searching is taken place very faster than text file. The binary file named as "index3.bin" for 3-word sentences, "index4.bin" for 4-word sentences like wise.

Onmoduleseρ():

In this method the stop word free sentences are taken as input from "o.txt" and separate out the sentences in different file according to number of word present in sentence. This is done by separating token from each sentence and counting the token. If the count match with particular number then that sentence is stored in respective file of type of sentence. Here text file is created "mod3.txt","mod4.txt" for 3-word and 4-word sentences respectively.

4.5.2.2 Methods of Class CcheckrmsDlg:

Onnext_crms():

This method's objective is for fetching next sentence from the "sent.txt" file for display on screen along with removed stop word sentence. Here the sentence line number and seek address is taken from "index3.bin" file which is created during Onrms() method. By using this binary file pointer is move next for sentence and dynamically remove stop word and display on screen.

Onprebtn():

This method's objective is for fetching previous sentence from the "sent.txt" file for display on screen along with removed stop word sentence. Here the sentence line number and seek address is taken from "index3.bin" file which is created during Onrms() method. By using this binary file pointer is move next for sentence and dynamically remove stop word and display on screen.

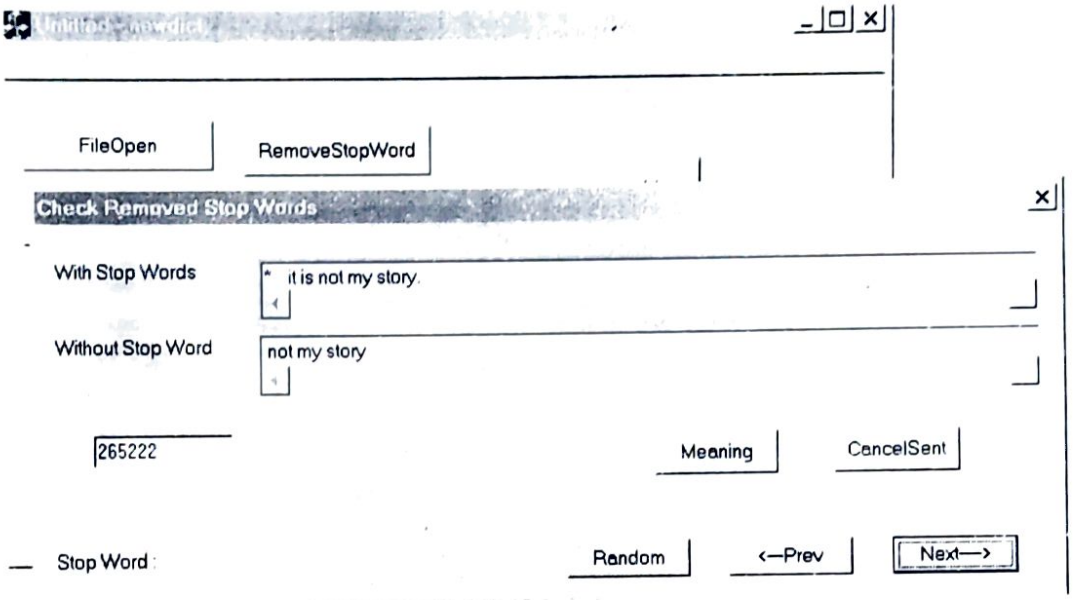
Onmeaning():

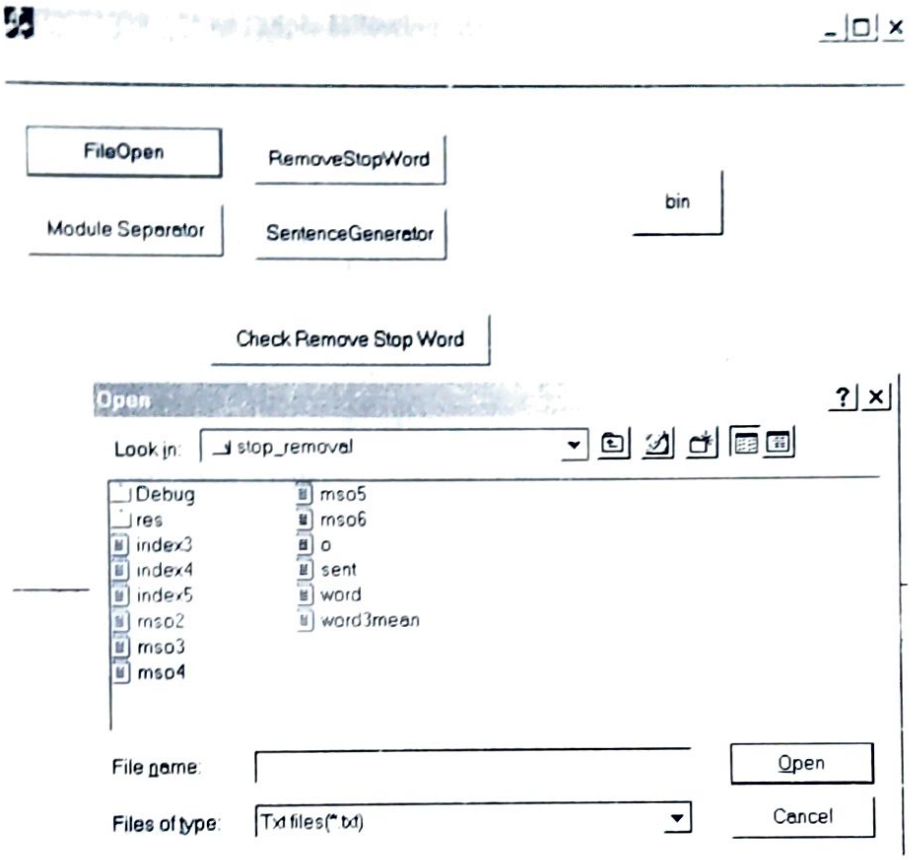
This method is used for storing the list of meaning full and meaning less sentences. When by above two method displayed on screen with and without stop word sentence, by human verification the meaning full sentence is identified. So when meaning full sentence occurred then by this method flag of meaning is set to '1' other wise it remains '0' for meaning less sentences. So by this in "meaningtxt.bin" file there is seek address of all sentence and flag for meaning full and meaning less for each sentence i.e. 0 or 1.

4.5.3 Guideline:

- First of all in this package click 'File Open' button to open multiple files by selection for sentences.
- After that press 'Sentence Generation' button for merging all selected files and generating one sentence per line in 'sent.txt'
- Press 'Remove Stop word' for removing the stop words from 'sent.txt' file sentences and create output file 'o.txt' and 'index*.bin' binary files.
- Press 'Module Separator' for separate out sentences in different files as per number of words per sentences. Files created are mso*.txt.
- Now press "Check Remove Stop word" for display of new dialog box for with and without stop word sentence simultaneously on screen text box.
- By pressing 'Next" and "Previous" button sentence are moved and displayed in text box.
- Press "Meaning" for meaningful sentences and "Cancel sent" for meaning less. Which is stored in 'wordmean.txt' binary file.

4.5.4 User Screen:





4.6 WORD ATTRIBUTES GENERATION AND SENTENCE ANALYZER:

Steps for Program:

Training the word database using Sentence Analysis algorithm to stabilize the statistics of words for word prediction in Sentence generation. Training will be done using following steps:

- Collect the Domain word Dictionary.
- Modify the word dictionary by adding attributes of a word as another fields.
- Initialize each attributes of word with random no.

- Now , apply following steps to all the sentences of the vocabulary text file.
- Tack one sentence form sentence text file and for all of it's word attribute modify according to Sentence Generation algorithm given in chapter 3.
- After successful completion of the program, it gives stable statistics of all the dictionary words will be store in file as a output. This file is used in Language modeling.

4.6.1 Class Description:

CLASS CWordattrbtDlg:

This class objective is for generation of random attributes for word of all sentence of given type of sentences. This class assigned attribute and iteratively increase or decrease value of attribute to make a set of attribute having common behaviour.

CWordattrbtDlg
Fptext: FILE * Maxsent: int wor[100][100]:char words:int filename[MAX]:char
Onfileopen():void Onloassent():void Onloadword():void Onsaveattrbt():void OnGetsentence():void Setattrbt():void Binsearch():void

Wordstruct:struct
word[100]:char a1:float a2:float a3:float a4:float a5:float

4.6.2 Methods Description:

Onloadword():

Objective of this method is load the words form wordlist file "word.txt" of given sentences file "mod3.txt", where wordlist generated by wordlist generation tool. After that the all words are taken into one big array of structure. At this time the random attributes are stored in this array of structure name "wordstrct", where a1,a2,a3,a4 and a5 are word's attributes.

Onsaveattrbt():

Objective of this method is to save the array structure "wordstrct" into the "temp.txt" file. In this file all words and its attributes are written sequentially.

Onloadsent():

Objective of this method is to read sentence from "mso3.txt" file and then take sentence one bye one in memory. After that

OnSetattribute():

Objective of this methods is read whole file till end of file and for each line of file apply OnGetsentence() and setattrbt() method.

OnGetsentence():

This method is use to read one line from file and return the array of character. In this method the sentence in form of character array is load and separation of token is taken place. The tokens of sentence are stored in one global character rarray named wrd[][].

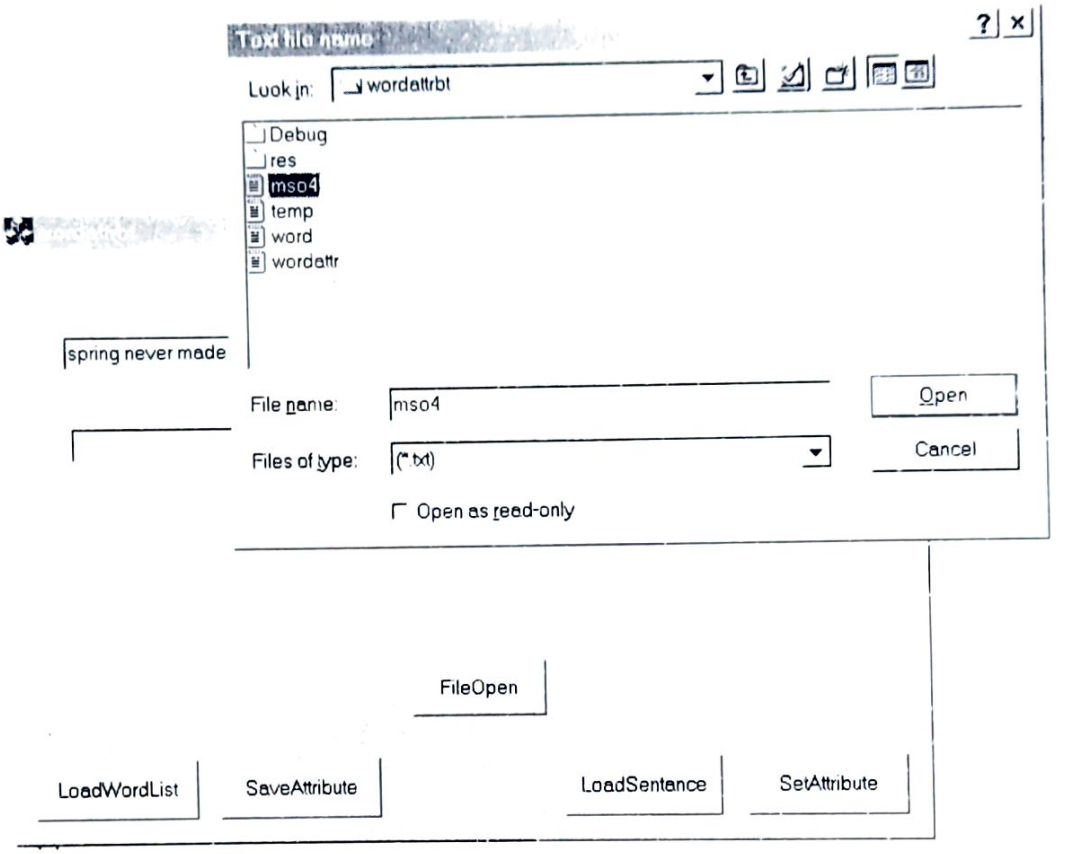
Setattrbi():

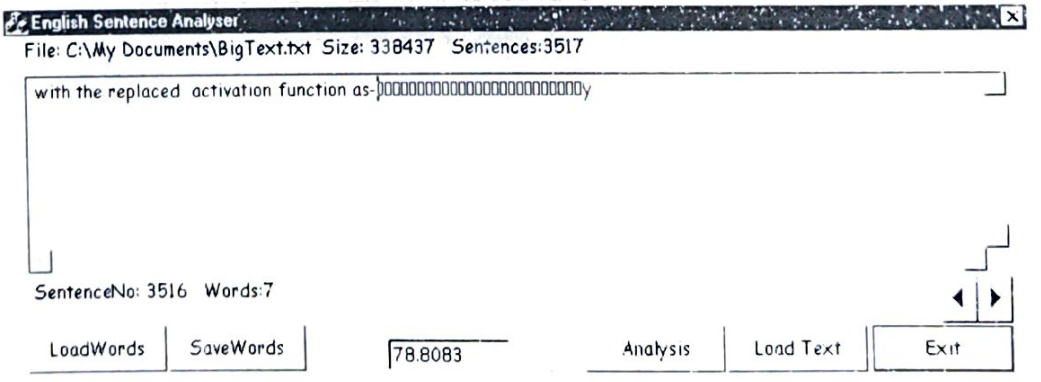
In this method the wrd[][] array is taken as input and after that each word from this array is searched by calling binsearch() method and if found then position in array of structure "wordstrct" return by binary search. After finding position in array of structure of given word corresponding word and attributes variables are stored in one temporary array of structure named "tmp". After that "process() " method is call for analysis and training of attribute to become in equilibrium using increasing or decreasing value of attribute.

Binsearch():

This method is used for searching by using binary search method of searching. Here the wrd[] is taken as input for search in array of structure "wordstrct". This method returns position of wrd[] in array of structure. If found then return positive value else it return negative value.

4.6.3 User Screen:





4.7 PHONEME EXTRACTION:

A phoneme is the smallest unit of sound. The whole word in any language is made up of one or more phoneme. For example word 'maximum' number of phonemes are three i.e.'ma'+ 'xi'+ 'mu' +'m'.

Here I develop program for phoneme extraction from English word by utterance of Gujarati language script.

Phoneme is combination of consonant and vowels. There are five vowels in English but here I take vowels of Gujarati, which is as follows: A, E, I, O, U, AU, EE, AI, AN,AO. Other consonants are taken from Gujarati Scripts. These Gujarati script consonant followed by Gujarati vowel is make different phonemes by combinations of one or more vowels.

4.7.1 Program Steps:

- First take English word as input.
- Compare each character of word with given array of vowel. If it is vowel add that character in array of phoneme-list of given word.
- Likewise take look-ahead characters from words and compare with array of consonants, vowel and combination of both. If found phoneme add that phoneme into phoneme-list of given word.
- Here check all combination of following three array. One array is for vowels and other two are for consonants.
- Here list of phoneme extracted is the phoneme sequence of given word.
- Convert phoneme sequence of word into integer sequence according to index taken in phoneme compare array.

CHAPTER 5 : TESTING

In this project testing is done through Unit Testing Techniques both Black box and White box Testing is possible. Here each module is separately tested.

5.1 WORDLIST GENERATION PROGRAM FOR HINDI-GUJARATI:

Here in this program developed in VC++ input taken as Hindi or Gujarati Corpora of text, which is processed by this algorithm and outputs wordlist from this text. This program gives sorted wordlist according to Indian Script i.e. 'Kakko-Brakhadi'.

5.1.1 Hindi Wordlist:

The number of words generated from single Hindi text file is not enough. So we have to take more than one input file and reputedly run this program for all input files. After that we have to Merge the all output files into single output file.

5.1.1.2 Input:

Hindi text file of corpus hindi.txt of size 500Kb.

Limits:

Specific Font for input file = 'AU' TTF Font
Maximum length of word = 50.
Maximum number of words=60000.
Maximum input file size=500Kb

Sample Input files is in Appendix A.3.2.1

5.1.1.2 Output:

Sorted Hindi words file hword.txt with 10000 words.

Sample Output file is in Appendix A.3.2.2

5.1.2 Gujarati Wordlist:

The number of words generated from single Hindi text file is not enough. So we have to take more than one input file and reputedly run this program for all input files. After that we have to Merge the all output files into single output file.

5.1.2.1 Input:

Gujarati text file of corpus guj.txt of size 500Kb.

Sample Input file is in Appendix.

Limits:

Specific Font for input file = 'Gopika' TTF Font
Maximum length of word = 50.
Maximum number of words=60000.
Maximum input file size=500Kb

5.1.2.2 Output:

Sorted Gujarati words file gword.txt with 10000 words.

Sample Output file is in Appendix.

5.2 DICTIONARY GENERATION TOOL:

This program is developed in VC++ and database in MS-Access. This program is for partially automatic dictionary generation from Hindi to Gujarati and manually Data entry tool for Hindi to Gujarati wordlist generated by above wordlist program.

5.2.1 Input:

Hindi wordlist file hindi.txt, Gujarati wordlist file gujarati.txt, Gujarati Script file gu.txt

5.2.2 Output:

Hindi to Gujarati dictionary file dic.html and dictionary.mdb MS-Access file. Dic.html file snap sort is present in Appendix.

5.2.3 Result:

Here by this dictionary generation tool the most of Hindi to Gujarati words are directly translated because of phonology similarity. For other word which are nearer to each other phonetically is also translated. But for phonetically different words here difficult to translate. For this problem morphological analysis is need for both languages.

5.3 STOP WORDS REMOVAL PROGRAM:

Stop words removal and module separator is combine tool for language which takes input as large corpus of English with more than one file and give output as single separated files for each type of sentence. Here type of sentences is number of word per sentence, i.e. 3-word sentences, 4-word sentences etc.

5.3.1 Input:

English language corpus text files of different domain and source. Here taken the list of files of 'Gutenburg' corpus having total size 51MB.

5.3.2 Output:

Combined sentence file 'sent.txt'. Binary files of seek address of Stop word free sentences for each type of sentences named 'index*.bin'. Also text file for all type of sentences 'o.txt', 'mso*.txt'. List of meaningful and meaningless sentences in stop word free sentences file.

Here number of stop word free 3-words sentences are extracted are 15000 along with meaningful and meaningless flag.

5.4 WORD ATTRIBUTE GENERATION AND SENTENCE ANALYZER:

5.4.1 Input:

Here the input is stop word free sentences file 'mso3.txt' fro 3-word sentences and wordlist of this file 'word.txt'. Which contains 15000 3-word stop word free sentences. Also random attributes a1 to a5 given as input in program.

5.4.2 Output:

Output is word attributes for each word in 'wordattrbt.txt' file and 'temp.txt' file in which after training of attribute new attributes are stored.

5.4.3 Result:

By above output result analysis show that the random attribute is generated initially for each word of sentences in word attribute structure. After that the by training the attribute value changes and come nearer for set of attributes and attribute are club in set of attributes. After some 20 minute of execution the system of attribute error is in equilibrium.

5.5 PHONEME EXTRACTION:

5.5.1 Input:

Here this program is develop in Turbo C in DOS mode. Program take input from word.txt list of 10,000 words.

5.5.2 Output:

Read the file one word at a time. After applying algorithm of phoneme extraction of sequence. Generation of second file phonseq.txt and write sequence of phoneme of that word.

Sample output:

Input word: 'ashwin'

Phoneme sequence: 'a' + 'sh' + 'wi' + 'n'

Input word: 'Gujarati'
Phoneme sequence: 'Gu' + 'ja' + 'ra' + 'ti'

Part of Sample Output file:

Input Word	Phoneme Count	Phonemes	Phoneme seq. ID
Aback	4	A+ba+c+k	0,33,9,9
Abalone	4	A+ba+lo+ne	0,33,182,115
abandon	5	A+ba+n+do+n	0,33,12,176,12
Abase	3	A+ba+se	0,33,118

5.5.3 Result:

This program is used for only English likewise same experiment can be done for Gujarati and Hindi language font so which can be useful for phonetically similar word translation for Hindi and Gujarati.

CHAPTER 6 : CONCLUSION

6.1 CONCLUSION:

Here in this dissertation implementation is not fully completed. But necessary resource collection and tool for Machine Translation is done. Main resource here is English, Hindi and Gujarati corpus both annotated and parallel. The Hindi to Gujarati Dictionary Generation tool is work for phonetically similar words. English-Hindi Dictionary taken and modified it for 25000 words and stored in database format along with grammatical attribute of word. Wordlist Generation, Stop word removal with module separator for given language is done. Word attribute generation for sentence analysis is designed and partially implemented.

6.2 FUTURE ENHANCEMENT:

In this project the future enhancement is for Dictionary generation tool for Hindi-Gujarati by morphological analysis. Word attribute generation and sentence analysis tool using neural network for making rule based for each language. By using more neural network the type of sentences can also identified. So, taking both language rule based vector bilingual translation can be possible.

REFERENCES

Books:

[B 1.] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal (1995) *Natural Language Processing: A Paninian Perspective,* Prentice Hall of India Private Limited, New Delhi.

[B 2.] Daniel Jurafsky, James H. Martin, (2002) *Speech and Language Processing An Introduction to Natural Language Processing, Computation Linguistics, and Speech Recognition,* Pearson Education, Delhi.

[B 3.] Rich and Knight, *Artificial Intelligence,*(2001) Second Edition, Tata McGraw-Hill Edition.

Conference Paper:

[CP 1.] Akshar Bharati Amba P. Kulkarni Vineet Chaitanya Satyam [July 1996.] *,Challanges in developing word analyzers for indian languages;* Presented at Workshop on Morphology, CIEFL, Hyderabad.

[CP 2.] Akshar Bharati Amba P. Kulkarni Vineet Chaitanya Satyam [1996]. *New challanges in automatic translation from an Indian perspective;* Presented at Int. Seminar on Automatic Translation, Annual DLA Meeting, Kuppam, June 1996. Organized by Dravidian Linguistics Association.

[CP 3.] Akshar Bharati Vineet Chaitanya Amba P. Kulkarni Rajeev Sangal Satyam[July 1997],; *Anukaaraka: Machine Translation in Stages;* Appeared in Vivek, A Quarterly in Artificial Intelligence, Vol.10, No.3 NCST, Mumbai, pp.22-25.

[CP 4.] Durgesh Rao *Machine Translation in India: A Brief Survey,* NCST,Juhu,Mumbai. www.elda.fr/proj/scalla/SCALLA2001/SCALLA2001Rao.pdf

[CP 5.] Saruoj Kaushik (1) and Prema Zutsh (2); *Analysis and Generation of Hindi Sentences, (1) IIT, New Delhi.(2)Banasthali Vidyapith, Banasthal.i*

[CP 6.] Rayner D'Souza, G.V. Shivakumar,D. Swathi and Pushpak Bhattacharya, *Natural Language Generation from Semantic Net like Structures with Application to Hindi*, Department of Computer Science and Engineering, IIT Bombay.

[CP 7.] Shachi Dave, Jignashu Parikh and Pushpak Bhattacharyya: *Interlingua-based English–Hindi Machine Translation and Language Divergence* Department of Computer Science and Engineering, Engineering, IIT Bombay.

[CP 8.] Srikanth RP & Dheeraj Kapoor ;*Machine Translation set for quantum leap in Indiahttp,* Report from Express Computer Magazine from www.expresscomputeronline.com/20011203/archives.htm

APPENDIX

A Hindi-Gujarati Character table:

Below table contains Two sub table Table-1 and Table-2.

Table-1 is for mapping ASCII values and ANSI- English characters with corresponding character of Gujarati and Hindi, for specific fonts. For Gujarati 'Gopika' font of *Gujarat Samachar* news paper is used and for Hindi 'AU' font of *Amar Ujala* newspaper is used.

Table-2 is for Gujarati-Hindi character mapping according to Indian Script. '*' placed in place where corresponding Hindi/Gujarati character is not found or there is Combination for two character for given single character of Hindi/Gujarati.

Table-1 for ASCII-Hindi-Gujarati Mapping				Table-2 for Gujarati-Hindi Script Mapping		
ASCII-Value	ANSI-English	ગુજરાતી	हिंद	Sr. No.	ગુજરાતી	हिंद
22			ा		અ	अ
24			औ		ઇ	इ
25			ाँ		ઇં	इ़
32					*	॰
33	!	!	!		ઈ	ई
34	"	ધ	ऋ		*	॰
35	#	ઈં	स		ઈં	ई़
36	$	ઈં	.		*	॰
37	%	%	ल		*	॰
38	&	:	×		ઉ	उ
39	'	’	%		ઉં	उ
40	(	(	(		*	॰
41	)	)	)		ઊ	ऊ
42	*	*	*		ઊં	ऊ
43	+	+	+		*	॰
44	,	,	,		ઋ	ऋ
45	-	-	-		ઋ	ऋ़
46	.	.	.		અ	ए
47	/	/	/			*

48	0	૦	0		ક	क
49	1	૧	1		ક	क
50	2	૨	2		ક્ક	क्क
51	3	૩	3		ક	क
52	4	૪	4		"	'
53	5	૫	5		ક	क्र
54	6	૬	6		ક્ક	क्क
55	7	૭	7		ક	क
56	8	૮	8		*	'
57	9	૯	9		ખ	ख
58	:	થ	=		ખ	ख़
59	;	ત	;		ખ	ख़
60	<	;	<		ક્ષ	ख
61	=	=	=		*	त
62	>	।	>		ખ	ख
63	?	?	?		ખ	ख
64	@	જ્ઞ	ज्ञ		ગ	ग
65	A	છ	'		ગ	ग
66	B	ઋ	क्र		ઘ	घ
67	C	ભ	ष्ट		ઙ	ङ
68	D	ઘ	ष्ठ		ચ	च
69	E	ઈ	श्व		ચ	च
70	F	ખ	स्र		ચ	च्च
71	G	ટ	त		છ	छ
72	H	ક્ષ	॥		છ	छ
73	I	ઁ	ड्ड		જ	ज
74	J	વ	छ्व		જ	ज
75	K	ણ	च्च		જ	ज
76	L	ર	रु		'	ो
77	M	સ	रू		જ	ज
78	N	શ	ह्न		જ	ज
79	O	ં	ह्व		જ્ઞ	ज्ञ
80	P	ઃ	क्क		જ	ज़
81	Q	ઊ	क्त		જ	ज
82	R	ઈ	क्र		જ્ઞ	ज

83	S	જી	ॄ		"	।
84	T	े	झ		ͻ	ड़
85	U	ै			ઝ	ड़
86	V	ફ	ङ्क		"	।
87	W	ઉ	ङ्ख		ઝ	झ
88	X	ઠ	ङ्ग		ટ	ट
89	Y	૩	ङ्घ		ટ્ઠ	ट्ठ
90	Z	ઢ	॰		ટ્ટ	ट्ट
91	[	^	[		ઠ	ठ
92	\	ૃ	झ		ઠ્ઠ	ठ्ठ
93	]	૮	]		ડ	ड
94	^	દ્દ	ट्ट		ડ્ડ	ड्ड
95	_	–	ट्ठ		ડ્ઢ	ड्ढ
96	`	ૐ	क्र		ઢ	ढ
97	a	ચ	ड्ड		ઢ્ઢ	ढ्ढ
98	b	મ	ड्ढ		ડ	ड्ट
99	c	બ	ठ		ણ	ण
100	d	ગ	स्र		*	।
101	e	ી	द्र		ણ	ण
102	f	ક	द्ध		ત	त
103	g	પ	द्द		(	त
104	h	ર	द्ध		ट	त्त
105	i	ે	द्ब		ત્ત	त्त
106	j	લ	द्ध्र		૭	३
107	k	ં	द्म		મ	म
108	l	ન	द्य		ત્ન	त्न
109	m	સ	द्व		ट	क्त
110	n	હ	द्ठ		૭	क्र
111	o	ૈ	थ		થ	थ
112	p	ુ	श्व		શ	श
113	q	ૂ	ह्ल		દ	द
114	r	િ	हृ		દ	द्द
115	s	જ	ह्य		દ	द
116	t	ા	ह्ल		મ	ह्म
117	u	ે	ह्व		ધ્	द्ध

118	v	५	व		ઙ	ઙ
119	w	◌	ख		ઙ	ઙ
120	x	ट	फ		ઙ	ઙ
121	y	ઔ	ब		ઙ	ઙ
122	z	ऽ	ध		ઙ	ઙ
123	{	◌	ऱ		ઙ	ઙ
124	|	.	स्र		ધ	ધ
125	}	ऋ	ॅ		ઘ	ધ
126	~	२	◌		ઽ	ઽ
127	□	□			ન	ન
128	€	€	€		ન	ન
129	□	ङ्ग	ख		ન	ન
130	‚	,	ग		પ	પ
131	ƒ	ƒ	ह		૨	૨
132	„	„	ह्ल		પ્ર	પ
133	…	…	ज		*	◌
134	†	†	ङ		પ્ર	પ
135	‡	‡	ण		*	◌
136	ˆ	ˆ	त		ફ	ફ
137	‰	‰	थ		ફ	ફ
138	Š	◌	ध		ફ	ફ
139	‹	◌	ऩ		ફ	ફ
140	Œ	Œ	ट		*	◌
141	□	□	फ		બ	બ
142	Ž	Ž			૭	૭
143	□	□	ब		ભ	ભ
144	□	□	भ		મ	મ
145	‘	‘	म		મ	મ
146	’	ह	ज		મ	મ
147	“	“	क्ष		ર	ર
148	”	”	ञ		ર	ર
149	•	ट	ज़		ય	ય
150	–	–	ङ्ख		ચ	ચ
151	—	—	—		૨	૨
152	˜	˜	ऊ		ર	ર

153	™	™	ञ		३	ह
154	š	š	ः		२	ह
155	›	›	‴		ल	ल
156	œ	œ	७		६	ल
157	□	□			ह्म	ह्म
158	ž	ž			ल	ल
159	Ÿ	Ÿ	४		ल	*
160					व	व
161	¡	५	ॅ		८	ठ
162	¢	६	·		श	श
163	£	७	‴		॥	ा
164	¤	८			९	ए
165	¥	’	अ		ष	ष
166	¦	¦	ट्ट		ा	*
167	§	श्र	इ		ष	ष
168	¨	९	ि		ष्ट	ष्ट
169	©	श्र	उ		ष	ष्ठ
170	ª	”	ऊ		ठ	*
171	«	५	ऋ		ष्ट	ष्ट
172	¬	ऽऽ	ॠ		श्र	श्र
173	-	-	ऌ		॥	ा
174	®	ॅ	0		श्र	श्र
175	¯	¯	ख्र		॥	ा
176	°	०	ए		श्च	श्च
177	±	·	ह		श्र	श्र
178	²	ॅ	ह		ा	*
179	³	ऊं	श		श्च	श्च
180	´	ॅ	·		स	स
181	µ	श्च	ह्र		स	स
182	¶	¶	ल		स	स्र
183	·	·	क		ऽ	*
184	¸	ॅ	.		ा	*
185	¹	श्र	ख		स	स्र
186	º	ऽ	ख्र		ऽ	*
187	»	ष	ग		ा	*

188	¼	ਧ	द		ੇ	ख़
189	½	ਨ	ह्न		ੋ	॰
190	¾	ਪ	ड		ੌ	॰
191	¿	ਫ	च		੍	ह़
192	À	ਬ	छ		ਹ	ह़
193	Á	ਭ	ज		ਹ੍	ह़
194	Â	ਮ	प		ਹ੍ਹ	ह़
195	Ã	ਯ	ठ		ਹੇ	ह़
196	Ä	ਰ	ट		ਹੈ	ह़
197	Å	ੜ	ट		ਹੋ	ह़ा
198	Æ	ਵ	ठ		ਹੌ	ह़ा
199	Ç	ਸ਼	ड		ਹ੍	ह़
200	È	ਖ਼	फ		ਹ੍ਹ	ह़
201	É	ਗ਼	ढ		ਹ੍ਹ	ह़ा
202	Ê	ਜ਼	ढ़		ਃ	ः
203	Ë	ਫ਼	ण		ਁ	ं
204	Ì	ਲ਼	त		ਕ਼	क़
205	Í	ਜ਼	थ		ਖ਼	ख़
206	Î	ਵ	द		ਗ਼	ग़
207	Ï	ੜ	ध		ਜ਼	ज़
208	Ð	ੲ	।		ਫ਼	फ़
209	Ñ	ਲ਼	:		ੱ	़
210	Ò	੶	‘		ੱ	़
211	Ó	ਹ	’		ੴ	ॐ
212	Ô	ਾ	‘		ਪ	ए
213	Õ	੯	ब		ਪ੍	ए
214	Ö	ਖ਼	भ		ੰ	ा
215	×	×	म		ਾ	ा
216	Ø	ਢ	य		ਿ	ि
217	Ù	ਠ	न		ਿ	ि
218	Ú	ੲ	र		ਿ	ि
219	Û	ਫ਼	झ		ਿ	ि
220	Ü	ਸ਼	ल		ਿ	ि
221	Ý	ੴ	ळ		ੀ	ी
222	Þ	]	'		ੀ	ी

223	ß	જ	व			
224	à	ઍ	श			
225	á	ઍં	ष			
226	â	ચ	स		ુ	ા
227	ã	ઞ	ह			
228	ä	ઽ				
229	å	૨				
230	æ	૬	ા		ૂ	ા
231	ç	૭	ि			
232	è	ૂ	ી			
233	é	ુ				
234	ê	ૃ				
235	ë	ૄ				
236	ì	ૅ	ि		ા	ી
237	í	૆	ि			*
238	î	ે	'		ા	ી
239	ï	ૈ				*
240	ð	ૉ	'			
241	ñ	૊	'			
242	ò	ો	'			
243	ó	ૌ	'			
244	ô	્	ી			
245	õ	૎	ૌ			
246	ö	૏	ૣ			
247	÷	ૐ	'			
248	ø	૑	૦			
249	ù	૒	૧			
250	ú	૓	ૐ			
251	û	૔	ૐ			
252	ü	'	'			
253	ý	૫	'			
254	þ	૬	'		ૂ	ા
255	ÿ	[	૭			
0						

B OUTPUT OF WORDLIST GENERATION TOOL:

For Gujarati:

Input: input.txt file.

ભારતમાં વસતા કરોડ હિન્દુઓના હૃદયમાં વસી ગયેલા શ્રી કૃષ્ણ હિન્દુ આત્માનું સર્વશ્રેષ્ઠ પ્રતીક છે. પૃથ્વી પર અવતાર ધરીને કૃષ્ણએ કરેલી અનેક લીલાઓનું વર્ણન ભાગવતમાં આવે છે જ્યારે શ્રીમદ્ ભગવદ્ ગીતા એ કુરુક્ષેત્રમા શ્રીકૃષ્ણએ અર્જુનને નિમિત્ત બનાવીને આખા વિશ્વને આપેલો ઉપદેશ છે. કૃષ્ણના અનેક રૂપો છે. દેવકીનો પુત્ર, યશોદાનો લાલો, ગોપીઓનો કનૈયો, ગોવાળીયાઓનો ગોવાળ, અર્જુનનો મિત્ર- સારથી, રાક્ષસોના વિધ્વંસક, રુકમણી, સત્યભામા જેવી આઠ આઠ પટરાણીઓના પતિ અને દ્વારિકાના રાજા છતાં એક જબરજસ્ત રાજનીતિજ્ઞ એવા કૃષ્ણ જેવું રહસ્યમય છતાં અદ્ભુત વ્યક્તિત્વ આજસુધી જોવા મળ્યું નથી.

જેમણે વેદો, ઉપનિષદો, મહાભારત અને શ્રીમદ્ ભગવદ્ ગીતાનો હાથ પણ લગાડ્યો નથી એવા લોકો મરજી પડે તેવી અપદગ્ધ મિમાંસાઓ કરતા હોય છે.

પણ કૃષ્ણની ગોપીઓ સાથેની લીલાઓને પણ સમજી શક્યા નથી. ગોપીઓને સમજવા માટે કૃષ્ણ પહેલાના રામાવતારમાં ડોકિયું કરવું પડે. કહેવાય છે કે ભગવાન શ્રી રામ અત્યંત રૂપાળા હતા. તેમના સીતાજી સાથે લગ્ન થયાં ત્યારે બ્રહ્માજીએ સરસ્વતીને કહ્યું : ''તમે પૃથ્વી લોક પર જાવ અને રામ અને સીતાની જોડી કેવી લાગે છે તે જોઈ આવી મને કહો.'' સરસ્વતી એક સપ્તાહ સુધી શ્રીરામના મહેલમાં મહેમાન બનીને રહ્યાં અને પાછા બ્રહ્માજી પાસે ગયા. બ્રહ્માજીએ પૂછ્યું ''શ્રી રામ - સીતા કેવા લાગે છે?'' ત્યારે સાક્ષાત સરસ્વતી બોલ્યાં : ''રામ અને સીતા એટલાં સ્વરૂપવાન લાગે છે કે હું તેમની અનુરાગી બની ગઈ છું. મારી પાસે એમનું વર્ણન કરવા શબ્દો નથી.''

''કંદર્પ કોટિ અગણિત છબી...'' એ શ્લોક દ્વારા કહેવાયું છે કે રામને જોઈને તો કરોડો કરોડો કામ

Output: word.txt file

અગણિત	ગોપીઓ	બોલ્યાં	સત્યભામા
કુર્જુનનો	ગોવાળ	ભગવદ	સપ્તાહ
કુર્જુનને	ગોવાળીયાઓનો	ભગવાન	સમજવા
અત્યંત	ગીતાનો	ભાગવતમાં	સમજી
કુદ્ભુત	ગીતા	ભારતમાં	સરસ્વતીને
કુપદગ્ધ	ઘણા	મને	સરસ્વતી
કુનેક	છતાં	મરજી	સર્વશ્રેષ્ઠ
કુને	છબી	મહાભારત	સાથેની
અનુરાગી	છું	મહેમાન	સાથે
અવતાર	છે	મહેલમાં	સારથી
આખા	જબરજસ્ત	મળ્યું	સાક્ષાત
આજસુધી	જેમણે	માટે	સીતાજી
આઠ	જેવી	મારી	સીતાની
આપેલો	જેવું	મિત્ર	સીતા
આવી	જ્યારે	મિમાંસાઓ	સ્વરૂપવાન
આવે	જાવ	યશોદાનો	સુધી
આત્માનું	જોઈ	રહસ્યમય	હતા
કેક	જોઈને	રહ્યાં	હાય
એટલાં	જોડી	રાજનીતિજ્ઞ	હોય
એમનું	જોવા	રાજા	હિન્દુઓના
એવા	ડોકિયું	રામને	હિન્દુ
એ	તમે	રામ	હું
ઉપદેશ	તો	રામાવતારમાં	હૃદયમાં

ઉપનિષદો	તેમના	રાક્ષસોના
કૃષ્ણએ	તેમની	રુકમણી
કૃષ્ણ	તેવી	રૂપાળા
કૃષ્ણના	તે	રૂપો
કૃષ્ણની	ત્યારે	લઈને
કનૈયો	થયાં	લગાડયો
ખરતા	દેવકીનો	લગ્ન
કરવા	દ્વારા	લાગે
કરવું	દ્વારિકાના	લાલો
કરોડ	નથી	લોક
કરોડો	નિમિત્ત	લોકો
કરેલી	પટરાણીઓના	લીલાઓને
કહો	પડે	લીલાઓનું
કહેવાય	પણ	વર્ણન
કહેવાયું	પતિ	વસતા
કહ્યું	પર	વસી
કામ	પહેલાના	વિધ્વંસક
કોટિ	પાછા	વિશ્વને
કુરુક્ષેત્રમાં	પાસે	વેદો
કેવા	પ્રતીક	વ્યક્તિત્વ
કેવી	પ્વી	શક્યા
કે	પુત્ર	શબ્દો
કંદર્પ	પૂછ્યું	શ્રીકૃષ્ણએ
ગઈ	બનાવીને	શ્રીમદ્
ગયા	બનીને	શ્રીરામના
ગયેલા	બની	શ્રી
ગોપીઓનો	બ્રહ્માજીએ	શ્લોક

For Hindi :

Input : input.txt

डब्ल्यूटीए टूर्नामेंट में अपनी पहली जीत के लिए तरस रही चोटिल रूसी खिलाड़ी अन्ना कुर्निकोवा को अब नए काम की तलाश है। अन्ना ने यूएसए नेटवर्क के एंटरटेनमेंट रिपोर्टर का काम छोड़ दिया है। उसका कहना है कि वह अपने साथी टेनिस खिलाड़ियों के साथ बातचीत करते समय सहज महसूस नहीं कर पाती थीं। अन्ना ने कहा कि वह ग्राउंड पर टहलने के दौरान अत्यधिक खाना भी खाने लगी थी। वह तो सिर्फ इसे प्रयोग की तरह ले रही थी। उन्होंने कहा कि वह अगले सप्ताह से यूएसए नेटवर्क के लिए 'एंटरटेनमेंट रिपोर्टर' का काम नहीं करेंगी। अन्ना ने तीन दिन तक नेशनल टेनिस सेंटर के चारों ओर अपना अधिकतर समय लोगों से हल्की-फुल्की बातें करने में बिताया।यह कहा जा रहा है कि अन्ना की नई भूमिका अधिकतर लोगों के गले नहीं उतरी। अखबारों में भी उनकी आलोचना छपी थी। नेटवर्क के प्रवक्ता टॉम कारासियोली ने इससे इनकार किया कि सही काम न कर पाने के कारण अन्ना को हटाया गया है। टॉम ने कहा कि यह बिल्कुल गलत है।

Output : word.txt

अखबारों	को	तीन	यूएसए
अगले	किया	थी	रहा
अत्यधिक	कि	थीं	रही
अधिकतर	की	थी	रिपोर्टर
अन्ना	कुर्निकोवा	दौरान	रूसी
अपने	खाने	दिन	लगी

अपना	खाना	दिया	ले
अपनी	खिलाड़ी	नई	लोगों
अब	खिलाड़ियों	नए	लिए
आलोचना	ग्राउंड	नेटवर्क	वह
ओर	गया	न	सप्ताह
इनकार	गलत	ने	समय
इसे	गलत	नेशनल	सेंटर
इससे	गले	नहीं	से
उतरी	चारों	नहीं	सहज
उनकी	चोटिल	प्रयोग	सही
उन्होंने	छपी	प्रवक्ता	साथ
उसका	छोड़	पर	साथी
एंटरटेनमेंट	जा	पहली	सिर्फ
के	जीत	पाती	हटाया
करते	टेनिस	पाने	हल्कीफुल्की
करने	टहलने	बातचीत	है
करेंगी	टॉम	बातें	
कर	टूर्नामेंट	बितायायह	
कहना	डब्ल्यूटीए	बिल्कुल	
कहा	तक	भी	
काम	तरस	भूमिका	
कारण	तरह	में	
कारासियोली	तलाश	महसूस	
का	तो	यह	

C OUTPUT OF DICTIONARY GENERATION TOOL :

अकबर = અકબર
अकमल = અકમલ
अकस्मात = અકસ્માત
अकारण = અકારણ
अकाल = અકાળે
अकादमी = અકાદમીનો
अकादमिक = અકાદમીનો
अखबार = અખબાર
अकाली = અકાલીદળ
अकेलापन = એકલું
अकेला = એકલા
अकेली = એકલી
अकेले = એકલા

अकेले अकेले से = એકલેહાથે
अक्टूबर = ઓકટોબર
अक्तूबर = ઓકટોમ્બર
अक्लमंद = અક્કલ
अक्ल = અક્કલ
अखबारो = અખબારી
अख्तर = અક્ષર
अगणित = અગણિત
अगरकर = અગરકર
अगरबत्ती = અગરબત્તીઓની
अगर = અગર
अगला = આગલા
अगले = આગલાં
अगली = આગલી
अगले = આગલું
अग्रवाल = અગ્રવાલ
अग्नि = અગ્નિ
अग्निशमन = અગ્નિસ્નાન
अचरज = અચરજ

D NEURAL NETWORK:

Important Feature of Neural Network:

- Neural networks have the ability to model multi-dimensional nonlinear relationships
- Neural models are simple and the model computation is fast
- Neural networks can learn and generalize from available data thus making model development possible even when component formulae are unavailable
- Neural network approach is generic, i.e., the same modeling technique can be re-used for passive/active devices/circuits
- It is easier to update neural models whenever device or component technology changes

Neural Network Structure:

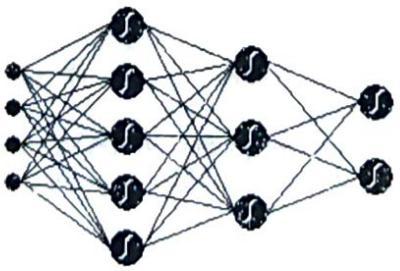
- A neural network contains:
  neurons (processing elements)
  connections (links between neurons)

- A neural network structure defines
  - how information is processed inside a neuron
  - how the neurons are connected
- Examples of neural network structures
  - multi-layer perceptrons (MLP)
- MLP is the basic and most frequently used structure

Multilayer Perceptrons :

This is perhaps the most popular network architecture in use today, due originally to Rumelhart and McClelland (1986) and discussed at length in most neural network textbooks (e.g., Bishop, 1995). This is the type of network discussed briefly in previous sections: the units each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output, and the units are arranged in a layered feedforward topology. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Important issues in Multilayer Perceptrons (MLP) design include specification of the number of hidden layers and the number of units in these layers (see Haykin, 1994; Bishop, 1995).

The number of input and output units is defined by the problem (there may be some uncertainty about precisely which inputs to use, a point to which we will return later. However, for the moment we will assume that the input variables are intuitively selected and are all meaningful). The number of hidden units to use is far from clear. As good a starting point as any is to use one hidden layer, with the number of units equal to half the sum of the number of input and output units. Again, we will discuss how to choose a sensible number later.



Self – organization:

As the complexity of artificial systems around us grow, it gets increasingly hard to keep all the details in the mind at the same time. The human mind cannot grasp infinite complexity (Miller 1956). This might eventually set a limit to what systems can be handled and developed. So far, the usual method of solving the complexity has been by using the analytical approach and trying to split the problem or system into small building blocks (as an example, lines of code), and by combining these blocks into larger blocks (macros).

Implicit in this method is a reduction of the problem in question. To be able to design a system using traditional means, the designer must split the system into parts detailed enough to be possible to describe (as an example in code). However, some systems are not possible to handle this way, as is shown by the work with traditional artificial intelligence (Penrose, 1990). To retain the functionality of the system, the parts will be so many and disperse that the "reduction" resulted in something that was more complex. To continue with the example of intelligence: Obviously construction of intelligence is possible, since it has been done accidentally by natural evolution (Darwin, 1859). Since evolution is a mechanical and determined process (Varela and Maturana, 1987), it is possible to emulate it, something that indeed has been shown (Ray, 2000). Further, the principle of evolution is not something that is limited to biological life, but can be employed in practical situation (Husbands, Harvey, Cliff and Miller, 1997). One alternative to traditional design by the analytical approach is to use a chaotic whole, and let that whole organize according to some internal principle. This principle might, as an example is "natural" selection as in the case of biological evolution. The gain in using self-organization would be that detail design should not have to be tinkered with, since the system is meant to move more or less on it s own accord towards the desired state. When developing (or rather evolving) systems this way, it is implicit in the method that control by analysis is beside the point.

Self – organizing systems has been an area of research to and from since the mid of the previous century. There has been much debate about its usefulness, and even claims that it theoretically impossible (Foerster, 1960), something that might have a base in the difference between the various definitions of the concept.

Forester defined the concept of Self – Organizing system as a system that decreases its level of entropy over time. To obey the second law of thermodynamics ("energy cannot be created nor destroyed"), this entropy would have to be absorbed by the supra system. When using this definition of self-organizing system, the question of Self – Organization largely became academic, since it could be reduced to redistribution of entropy within the supra system.

Later, Ashby defined Self – Organizing system as a system moving from "bad" to "good" organization, claiming that all systems strive for equilibrium (Ashby, 1962). He also

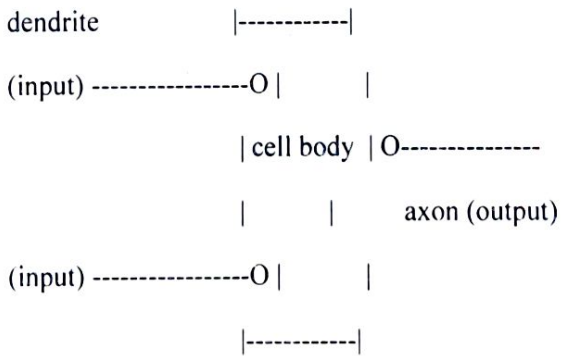
added that Self – Organization was not a special condition, but something that is true for all *dynamic* systems acting under unchanging laws.

After the research discipline of Self – Organization systems theory has been established; a FAQ collecting common definitions and questions has been written (Lucas. 2000). In it, Self – Organization systems are described as when system structure appears and evolves without explicit pressure from the environment: the change agents are internal to the system.

Kohonen algorithm for pattern classification:

In using the outdated Kohonen Algorithm (explained below) we can simulate an intelligent computer opponent that can learn from its mistakes. For the interested few, here in my treatment on the subject.

The least sophisticated of AI algorithms simulates the function of the biological neuron. First proposed four decades ago, the outline of the basic model is as follows



Where

O.....A synapse.

The synapse is not an actual physical linkage; insert it is a temporary chemical one that can very over time. To represent these variable connections in our computer model we assign a multiplicative weight to each input pathway. A high weight term denotes a favorable connection. The cell body contains a predefined value known as the threshold. An output signal will be produced if the input to the neuron is greater than the threshold value. We now require a mechanism that allow us the simulate learning in our perceptrons. In biological systems, the process of learning is thought

to involve modifications in the effective coupling between neurons. In our computer model, we make small adjustments to the weights.

To appreciate what this means, consider this learning paradigm:

- Set the weights w and thresholds t of perceptrons to random values

- Present the input x(0), x(1), x(2), ..., x(n-1)

- Calculate the actual output by comparing the threshold value and the weighted sum of the input

$$\text{Weighted sum} = \sum_{i=0}^{n-1} w(i) * x(i)$$

- Alter the weights to reinforce correct decisions and discourage incorrect decisions.

Adaptation of the learning paradigm lead to the development of the Kohonen Network Algorithm, named after Professor T. Kohonen of the Faculty of Information Sciences at the University of Helsinki, Finland. Instead of comparing input values to thresholds, (as in the case with perceptrons) Kohonen compared the weights of all output nodes and picked the set of nodes having weights that closely matched the magnitude of the input signal. The self-Organizing network consists of a matrix of output nodes j all of which are connected to every input node i.

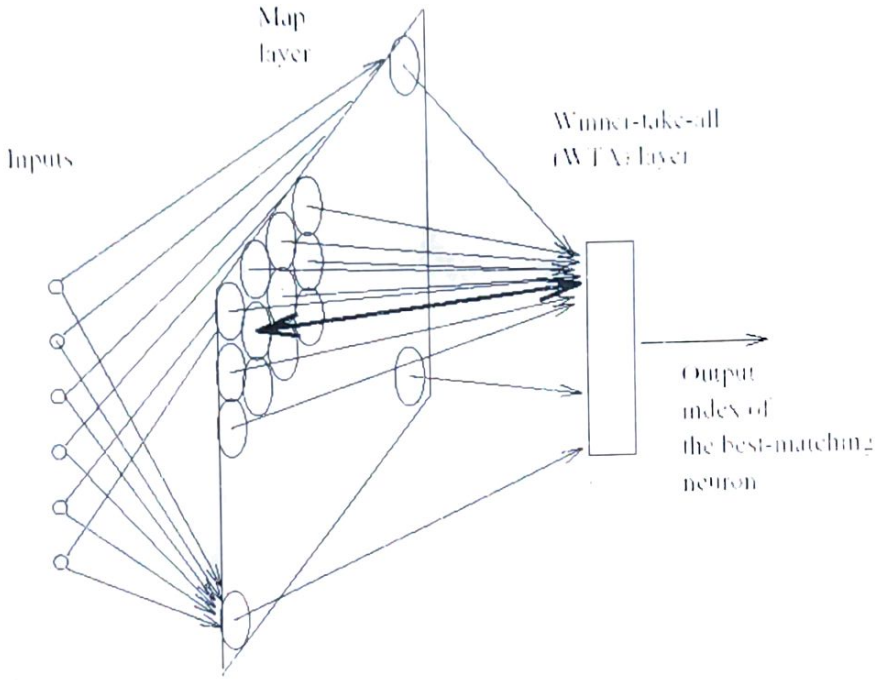


Fig. D Kohonen Model

The algorithm determines the "winning" node j* that closely matches the expected output as determined by the set of input nodes i. Modifying the weights of j* and its neighbors will produce a set of desired outcomes.

Kohonen successfully implemented this technique for a speech recognition system in the mid 1980's.

KOHONEN NETWORK ALGORITHM:

1. Initialise Network

- Define w(ij) (0 <= i <= n-1) to be the weight from input node i to output node j. Set the weights from the n inputs to some small values. Set the initial radius of the neighborhood around node j, N(j), to some large value.

2. Present Input

- Present input x(0), x(1), x(2), ... , x(n-1) where x(i) is the input to node i.

3. Calculate Distance

- Compute distance d(j) between input node i and each output node j as in

$$d(j) = \sum_{i=0}^{n-1} (x(i) - w(ij))^2$$

4. Select minimum distance and denote output node j with minimum d(j) to be j*.

5. Update Weights:

   - Update weight for node j* and its neighbors as defined by the extent of boundary N(j).

$$w(ij) = w(ij) + M * (x(i) - w(ij))$$

The M term is used to control the rate of weight adjustment. This value should be set in the range [0.5, 1] and then gradually decreased in accordance to a linear function as the number of learning cycles increases.

6. If the expected solution set has not been found, repeat the learning cycle (steps 2-5).

7. The solution set S of the network is now a counter-strategy tactic that a computer opponent can use against the player.

If, for example, the network consists of 16 output nodes, four input nodes and minimum neighborhood size of four, the Kohonen Algorithm can develop 216 (9 * 4!) unique strategy tactics. Provided that the network has been trained well, the computer opponent will be rather challenging.