CHAPTER 6

DISAMBIGUATION OF तो ('to')

1. INTRODUCTION

In the last chapter, we have seen how words belonging to different part of speech can be disambiguated. The various techniques applicable for resolving ambiguities at the word level can be achieved though it is a difficult task as it needs an extensive rule-base specifying the context of each instance of word disambiguation.

In this chapter we will see the extent to which one needs to work for disambiguating the simple morpheme ‘to’.

‘to’ is an inflectional suffix attachable to Marathi verbs. The principles of Marathi verb morphology studied for disambiguation of ‘to’ are also applicable while working on disambiguation of any other Marathi inflectional verb suffixes. We refer to inflectional suffixes attachable to verbs as the suffixes which are the realization of tense, aspect, mood (TAM) and gender, number, person (GNP) of the action expressed by a verb. Inflection refers to a change in the form of a word (usually by adding an affix to indicate a change in its grammatical function (e.g., boy-boy’s). Inflection produces words by affixation to stems while derivation produces stems by affixation to stems.

Before we proceed further in the direction of resolving the ambiguity of inflectional suffixes attachable to Marathi verbs for machine translation of Marathi texts into Hindi or English, we need to do the exercise in a mono-lingual situation. For this purpose, it is necessary to understand the complex Marathi verb morphology. We will first take an account of traditional description of Marathi verb morphology. The modified version (as discussed in this chapter) of the original ‘aakhyaata’ theory for Sanskrit¹, explains well the complex Marathi verb morphology. As Marathi language has emerged from Sanskrit language, influenced by Sanskrit language, some of the features of Marathi verb morphology are shared by the Sanskrit language. Even Hindi has emerged from Sanskrit language, influenced by Sanskrit language, and so, this approach based on the ‘aakhyaata’ theory is applicable even to the disambiguation of inflectional verb suffixes of Hindi.

The discussion on the ‘aakhyaata’ theory for Marathi inflectional verb suffixes in the book ‘maraaThiiche wyaaakaRaNa’ written by Mr. Moro Keshav Damale² is modified and presented here. The modifications have been done both in the content and in the way of presentation.
A detailed account of Marathi verb morphology is given in section 2 while discussing ‘aakhyaata’ theory and its theoretical paradigm in section 2.1. Section 3 contains the description of the Marathi inflectional verb suffix ‘to’. The narration of the methodology of disambiguation of the Marathi inflectional verb suffix ‘to’ is presented in section 4.

2. MARATHI VERB MORPHOLOGY

We consider a regular verb ‘basaNe’_to sit. ‘basa’ is considered as ‘dhaatuu’ (root form) of verb ‘basaNe’_to sit. ‘dhaatuu’ ‘basa’ of verb ‘basaNe’_to sit is that root form of verb to which all inflectional and derivational suffixes are attached. ‘basaNe’ is the form which finds a place in any traditional dictionary as a head word. Though the word ‘basaNe’ is used as verb as well as noun, no Marathi dictionary accounts for the entry for ‘basaNe’ as noun. The fact is accounted for through derivational morphology. We restrict only to inflectional suffixes attachable to verbs in Marathi. In other words we will not consider verbal nouns and the inflectional suffixes attachable to them in this chapter.

The list of inflectional verb suffixes in Marathi is shown in table 1.

<table>
<thead>
<tr>
<th>ta</th>
<th>to</th>
<th>ena</th>
<th>laa</th>
<th>laata</th>
<th>leles</th>
<th>lelaasa</th>
<th>aayachesa</th>
<th>lelyaasa</th>
<th>aayachyaata</th>
</tr>
</thead>
<tbody>
<tr>
<td>taa</td>
<td>te</td>
<td>uu</td>
<td>lyaa</td>
<td>lesa</td>
<td>leleta</td>
<td>lelaata</td>
<td>aawes</td>
<td>lelyaata</td>
<td>aayachaa</td>
</tr>
<tr>
<td>tesa</td>
<td>tos</td>
<td>le</td>
<td>laasa</td>
<td>keeta</td>
<td>lelaa</td>
<td>lelo</td>
<td>aaweta</td>
<td>aayacho</td>
<td>aayachii</td>
</tr>
<tr>
<td>taata</td>
<td>tii</td>
<td>lo</td>
<td>liisa</td>
<td>lyaasa</td>
<td>leliisa</td>
<td>aNaara</td>
<td>aawiiita</td>
<td>aayache</td>
<td>aayachyaal</td>
</tr>
<tr>
<td>aa</td>
<td>a</td>
<td>lii</td>
<td>lele</td>
<td>lyata</td>
<td>lelli</td>
<td>ashiila</td>
<td>aawaa</td>
<td>aayachiisa</td>
<td>aayachaasa</td>
</tr>
<tr>
<td>aala</td>
<td>ela</td>
<td>aawe</td>
<td>aawii</td>
<td>lelyaa</td>
<td>atriila</td>
<td>aayache</td>
<td>aawyaata</td>
<td>aayachaata</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Inflectional Suffixes Attachable to Marathi Verbs

2.1 THE ‘aakhyaata’ THEORY

The ‘aakhyaata’ theory forms the basis of analysis of the inflection of Marathi verbs. We need the following definitions to explain ‘aakhyaata’ theory. An example that follows a definition of a term is in the order of Marathi, Hindi and English. (All examples in this chapter are in the same order.) ‘padiima’_morpheme is the smallest meaningful unit of grammatical analysis. A ‘padiima’_morpheme can be ‘mukta padiima’_free morpheme and ‘baddha padiima’_bound morpheme. ‘mukta padiima’_free morpheme can exist alone (e.g., ‘chaalaNe’ / ‘chalanaa’ / walk). ‘dhaatuu’_verb-root is simple, made up of a single morpheme. It is the basis for compounding and affixation. (e.g., ‘chaal’ / ‘chal’ / walk) Stems may be complex, made up of one or more morphemes, which is the basis for affixation (e.g., ‘awichal’ / ‘awichal’ / immovable). An affix is a bound morpheme, which adds lexical or syntactic information to a root or a stem. ‘pratyaya’_suffix is an affix that is attached to the end of a root or a stem (e.g., ‘chaalato’ / ‘chalataaa’ / walking).
‘charama pratyaya’_closing suffix closes the constructions (e.g., ‘chaalato’ / ‘chalataa’ / walking or ‘awichalataapurwaka’ / ‘awichalataapurwaka’ / immovability). ‘pada’ is the (word) form, which is a result of an attachment of the closing suffixes to a stem (e.g., ‘chaalato’ / ‘chalataa’ / walking or ‘awichalataapurwaka’ / ‘awichalataapurwaka’ / immovability) ‘kriyaapada’_verb form is a ‘pada’ in a sentence, which refers to an action (e.g., ‘chaalato’ / ‘chalataa’ / walking). ‘kriyaapada’ is known as ‘aakhyaata’ in Sanskrit.¹ ‘kriyaapada’ carries the information about TAM (tense, aspect, and mood) and GNP (gender, number, and person). ‘aakhyaata’_theory studies the relation between ‘kriyaapada’, inflectional suffixes and TAM-GNP information carried by them. ‘dhaatuu’ or ‘kriyaa’ is the root to which suffixes are attached to form a ‘kriyaapada’ (e.g., ‘chaala+to - chaalato’ / ‘chala+taa - chalataa’ / walk+ing - walking). ‘aakhyaata pratyaya’ is the closing suffix, which is attached to the verb root / stem (e.g., ‘to’ in ‘chaalato’ / ‘taa’ in ‘chalataa’ / ‘ing’ in walking). ‘aakhyaata’ refers to the ‘kaaLa’ (tense) and ‘artha’ that includes aspect, mood and gender, number, person. ‘aakhyaata’ is realized through ‘aakhyaata pratyaya’. By and large, the phonemic shape of the aakhyaata pratyaya is the basis of naming the ‘aakhyaata’. ‘artha’ refers to the power, which enables to get the information about TAM & GNP about ‘kriyaapada’.

To all intents and purposes, an attachment of ‘aakhyaata pratyaya’ to ‘dhaatuu’ or ‘kriyaa’ produces ‘kriyaapada’. ‘aakhyaata pratyaya’ carries information about TAM and GNP. ‘aakhyaataartha’ includes three tenses, six aspects, twenty moods. ‘aakhyaata pratyaya’ may refer to three genders, two numbers and three persons in Marathi.

The list of tenses, aspects, moods, genders, persons and numbers manifested in Marathi grammar is listed below.

- **Tense (T):** Present (‘wartamaana’), Future (‘bhawishya’), Past (‘bhuta’)
- **Aspect (A):** Simple (‘saadhaa’), Habitual (‘ritii’), Perfect (‘puurNa’), Imperfect (‘apuurNa’), Progressive (‘saatatya’), Next (‘sannihita’), Definite (‘nihsanshaya’)
- **Mood (M):** Motive (‘uddesha’), Promise (‘aashwaasana’), Obligation (‘kartawya’), Ability (‘yogyataa’), Surprise (‘aashcharya’), Indicative (‘widhaanaartha’), Imperfect Conditional (‘asiddha sanketa’), Disappointment (‘niraashaa’), Possibility (‘sambhawaniyataa’), Wish (‘ichchhaa’), Request (‘winantii’), Permission (‘anudnyaa’), Conditional (‘sanketa’), Advice (‘upadesha’), Imperative (‘aadnyaa’), Consent (‘sammati’), Blessing / Curse (‘aashirwaada / shaapa’), Indifference (‘audaasinya’), Suspicion (‘sambhrama’)
- **Gender(G):** Masculine (‘pullinga’), Feminine (‘striillinga’), Neuter (‘napunsaakalinga’), Masculine + Feminine (‘ubhayalinga’) (this is not the grammatical gender either of Marathi or Hindi or English)
- **Number(N):** Singular (‘ekawachana’), Plural (‘anekawachana’)
- **Person(P):** First (‘prathama’), Second (‘dwitiya’), Third (‘tritiya’)

¹ A more detailed explanation of ‘aakhyaata’ theory and its application in Marathi grammar can be found in standard Marathi grammar texts. This simplified explanation provides a basic understanding of the concept.

201
All tenses, aspects, moods, genders, persons and numbers listed above are active while translating from Marathi into Hindi or English. In other words, corresponding realization of these tenses, aspects, moods, genders, persons and numbers are different in these three languages. For example, translation of Marathi sentence ‘to kaama karato’ into Hindi is ‘wo kaama karataa hein’ and in English it is ‘he works’ while ‘tii kaama karate’ in Marathi means ‘wo kaama karatii hein’ in Hindi and ‘she works’ in English. Thus, gender difference is marked while translating from Marathi into Hindi and English.

2.1.1 Theoretical Paradigm of ‘aakhyaata’ Theory

The theoretical paradigm of Marathi verb morphology explains the noticeable characteristics of Marathi verb morphology in terms of agreement (section 2.1.1) of verbs (‘kriyaapada’) with nominal (nouns and pronouns) serving as grammatical subjects or grammatical objects in the sentence, types of verbs (‘kriyaapada’) (section 2.1.2), rules of attachments of ‘aakhyaata pratyaya’ (section 2.1.3) and classification of ‘aakhyaata pratyaya’ (section 2.1.4).

2.1.1.1 Agreement

Marathi grammar has three grammatical genders, two grammatical numbers, and three grammatical persons. ‘kriyaapada’ (verb form) agrees in gender, number, and person with grammatical subjects or grammatical object in a sentence. One can assign six labels to a kriyaapada that attribute TAM and GNP information respectively. A ‘kriyaapada’ may correlate with different grammatical subject and grammatical object as shown in table 2 on the next page. (Because of the size of the table it is shown on next page.)

From table 2 it is clear that egoitic / egoistic / self referring, formal honorific and informal honorific reference to a single person is done by the use of plural pronouns in Marathi and so the use of the corresponding verb form is also in plural form. This phenomenon becomes crucial while using such verb forms.

2.1.1.2 Types of ‘kriyaapada’s

‘kartarii kriyaapada’, ‘karmaNii kriyaapada’ and ‘bhaawe kriyaapada’ (verb form) are three classes of Marathi ‘kriyaapada’s made on the basis of the agreement of ‘kriyaapada’ in gender, number, and person with grammatical subjects or grammatical objects in a sentence.

- ‘kartarii kriyaapada’ agrees in gender, number, and person with that of grammatical subject. (‘to khaato’ & tii ‘khaate’ / ‘wo khaataa hai’ & ‘wo khaatii hai’ / he eats & she eats)
- ‘karmaNii kriyaapada’ agrees in gender, number, and person with that of grammatical direct object. (‘laaDuu khaallaa’ & ‘barfii khaallii’ / ‘laaDDuu khaayaa’ & ‘barfii khaayii’ / ate laaDuu & ate barfii)
- ‘bhaawe kriyaapada’ does not agree in gender, number, and person either with that of grammatical subjects or grammatical objects. It is always in
neuter gender, singular number and third person. ('tyaane khaawe’ (M/N) or ‘tine (F) khaawe’ / ‘use (M/F) khaanaa chaahiye’ / he should eat or she should eat) (M: masculine, F: feminine and N: neuter)
the procedure of formation of stems. ‘aakhyaata pratyaya’s are attached to these stems. These rules can be represented as shown in table 3.

<table>
<thead>
<tr>
<th>BC</th>
<th>AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>e</td>
<td>aaya</td>
</tr>
<tr>
<td>i</td>
<td>aaii</td>
</tr>
<tr>
<td>aaya</td>
<td></td>
</tr>
</tbody>
</table>

The above table demonstrates that the changing part (BC) of the ‘dhaatuu’ (root) changes to changed part (AC) and thus the root is transformed to stem. For example, consider the verb ‘karaNe’ to do. ‘kara’ is the ‘dhaatuu’ (root) of the verb ‘karaNe’ to do. The word ‘kara’ is formed of two consonants (k & r) and two vowels (a & a). We can write equation as ‘kara’ = ‘k+a+r+a’. The changing part in ‘kara’ is ‘a’ after ‘r’. It changes to ‘a’ as in ‘karato’ do/does, to e as in ‘karela’ will do, to ii as in ‘kariila’ will do and to ‘aaya’ as in ‘karaayachaa’ used to do.

Over 35 verbs have irregular conjugations. There are verb paradigms with different order. In other words, some verbs allow attachments of less number of ‘aakhyaata pratyaya’s. For example, the paradigm of verb root (‘saanjaawaNe’ get atmosphere as at the time of beginning of evening) conjugates only for neuter gender and third person such as ‘saanjaawale’, ‘saanjaawate’ carrying ‘aakhyaata pratyaya’s, ‘le’ and ‘te’. Similarly transitive verbs conjugate to agree with the grammatical subject or the grammatical object whereas intransitive verbs do not conjugate to agree with grammatical objects. Hence, the number of members of the paradigm differs.

2.1.1.4 ‘aakhyaata pratyaya’

Each ‘aakhyaata pratyaya’ has its own phonemic shape and its own function(s). There are two ways of classification of ‘aakhyaata pratyaya’s, namely, based on their phonemic shape and the possibility of agreement with the grammatical subject or grammatical object as to gender.

2.1.1.4.1 Classification of ‘aakhyaata pratyaya’

‘aakhyaata pratyaya’s can be classified in two ways.
2.1.1.4.1.1 Classification Based on Phonemic Shape

The phonemic shape of the ‘aakhyaata pratyaya’s serves as the basis for the classification of ‘aakhyaata pratyaya’s. The word ‘pratyaya’ is not included in the names of such classes. There are eight such classes as stated below. The ‘aakhyaata pratyaya’s included in each class is given under the name of the class.

- **prathama taakhyaata**
- **dwitiya taakhyaata**
- **laakhyaata**
  - ‘laasa’
- **waakhyaata**
- **ii-aakhyaata**
- **uu-aakhyaata**
  - ‘uu’ / ‘o’, ‘aa’ / ‘o’, ‘ota’
- **iila-aakhyaata**
- **chaakhyaata or aayachaakhyaata**

It is interesting to note that the members of the same class of Marathi inflectional suffixes based on their phonemic shape share the information about TAM. For example, ‘to’ and ‘te’ belong to ‘prathama taakhyaata’ and they share information about TAM, say, simple present tense and indicative mood. Thus, we can say that there is a set of information about TAM corresponding to each class of Marathi inflectional suffixes based on their phonemic shape. Further, a set of information about GNP varies with the particular suffix of that class. Here, the set of information refers to a set of units of information about tense, aspect, mood and gender, number, person separately.

2.1.1.4.1.2 Classification Based on Gender Agreement

Possibility of inflection for gender is another basis of classification of ‘aakhyaata pratyaya’s.

- ‘salinga aakhyaata pratyaya’ agrees in gender with that of grammatical subject or grammatical object.
- ‘alinga aakhyaata pratyaya’ does not agree in gender with that of grammatical subject or grammatical object.
‘ii-aakhyaata’ and ‘iila-aakhyaata’ are ‘alinga aakhyaata pratyaya’s while rest are ‘salinga aakhyaata pratyaya’s.

A traditional description of Marathi verb morphology, that is, the ‘aakhyaata’ theory thus explains various tenses, aspects, moods, genders, numbers and persons which can be denoted by a given verb form. It also discusses the relation between GNP-information attributed to grammatical subjects / objects and ‘aakhyaata pratyaya’. The types of verb forms (‘kriyaapada’) based on agreement of verb form with the grammatical subject or grammatical object in gender, person and number and the rules for stem formation and the classification of ‘aakhyaata pratyaya’ are also described in the ‘aakhyaata’ theory. This helps one to locate a given inflectional verb suffix on the map of Marathi verb forms.

With the knowledge of TAM & GNP information attributed to inflectional suffixes attachable to Marathi verbs (which can be drawn through the ‘aakhyaata’ theory for Marathi inflectional verb suffixes), one can assign a tag to a given verb form. Each such tag is necessarily made up of six labels, namely, T, A, M, G, N, and P. Thus, we can say that there is a particular set of information about TAM-GNP corresponding to each class of Marathi inflectional suffixes based on their phonemic shape.

Now we will see how one can disambiguate the Marathi inflectional verb suffix ‘to’ on the foundation of ‘aakhyaata’ theory.

3. THE INFLECTIONAL VERB SUFFIX ‘to’

With the help of the discussion on ‘aakhyaata’ theory as above we can now locate the Marathi inflectional verb suffix ‘to’ on the map of Marathi inflectional verb suffixes in particular and Marathi verb forms in general. (Henceforth ‘to’ will be referred to simply as the suffix instead of Marathi inflectional verb suffix.) In other words, each such suffix is loaded with information about TAM-GNP. Among all such suffixes we can establish the all possible and correct information about TAM-GNP related to the suffix ‘to’. This can be done more successfully with the help of Marathi corpus. Here, it is worth while noting that almost all Marathi inflectional verb suffixes are ambiguous, that is to say that, they are loaded with more than one set of information about TAM-GNP.

It is found that the suffix ‘to’ is included in two classes based on the phonemic shape of suffixes, namely, ‘prathama taakhyaata’ and ‘dwitiya taakhyaata’ (see section 2.1.4.1). So, the suffix ‘to’ is loaded with two sets of information about TAM. There also exist various subsets of information about GNP that can be assigned to the suffix ‘to’ respectively. The table 4 can guide one to fix these sets.
4. TECHNIQUES FOR DISAMBIGUATION OF THE SUFFIX ‘to’

When a human translator translates a text, he struggles for resolving linguistic ambiguities at the word level and the sentence level. The expertise of the human translator can be measured in terms of the ease with which the translator can resolve ambiguities. Ambiguities at the morphological level are not even noticed by human translators. But when it comes to machine translation the programmer needs to teach the machine to resolve linguistic ambiguity even at the morphological level. In the context of the machine, teaching a machine refers to writing a suitable program for reaching the predefined goal.

Linguistic ambiguity at the morphological level refers to the multiple meanings of a morpheme. The meaning of a morpheme refers to its function in the word. The function of a morpheme in the words is interpreted / represented in terms of the information about TAM-GNP coded in the morpheme when it is inflectional verb suffix. Thus linguistic ambiguity at the morphological level, in the case of inflectional verb suffixes, can be resolved by assigning a correct single tag to the verb form. This tag should essentially denote the exact information about TAM-GNP related to that suffix in a given context.

Evidently, linguistic ambiguity at the morphemic level affects the meaning of the word, especially, in the case of verb forms. For example, consider the sentences, “mii yeto’I come’ (‘mii’I, ‘yeto’come) and “to yeto’He comes’ (‘to’He, ‘yeto’comes). In these sentences neither ‘mii’I, nor ‘to’He nor ‘ye’(root form of ‘yeto’come(s)) is ambiguous. But, the culprit is the suffix ‘to’. The suffix ‘to’ can have several context-independent functions. In other words, the suffix ‘to’ is loaded with a number of sets of information about TAM-GNP whenever the context is not specified. Several possible interpretations of a suffix in a word in isolation (without specifying context) may lead to several translations of that word. Thus, an ambiguous suffix contributes to the linguistic ambiguity at the word level. (See section 2.7.3 in chapter 3.)

‘mii’I and ‘to’He define the context in the above sentences respectively. The translation of ‘mii yeto’ in Hindi is ‘main aataa huun’. It is translated in English

<table>
<thead>
<tr>
<th>Aakhyaaata</th>
<th>Tense</th>
<th>Aspect</th>
<th>Mood</th>
<th>Gender</th>
<th>Number</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘prathama taakhyaaata’</td>
<td>Present</td>
<td>Habitual, Simple, Perfective,</td>
<td>Indicative</td>
<td>Masculine (M) and Feminine (F)</td>
<td>Singular (S) and Plural (P)</td>
<td>First and Third</td>
</tr>
<tr>
<td>Past</td>
<td>Immediate</td>
<td>Indicative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Future</td>
<td>Immediate</td>
<td>Indicative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘dwitiiya taakhyaaata’</td>
<td>Past</td>
<td>Imperfective</td>
<td>Conditional</td>
<td>M &amp; F</td>
<td>S &amp; P</td>
<td>First</td>
</tr>
</tbody>
</table>

Table 4: Relation between the suffix ‘to’ and TAM-GNP
as ‘I come’. The string ‘to yeto’ is translated in Hindi as ‘wo aataa hai’. Its translation in English is ‘He comes’. Thus, ‘yeto’ in isolation can mean in Hindi either as ‘aataa huun’ or ‘aataa hai’. It can mean in isolation in English either as ‘come’ or ‘comes’. So the dictionary for machine translation should necessarily include both the meanings of ‘yeto’ in the case of Hindi as well as in the case of English. This can be achieved economically by storing different functions of the suffix ‘to’ separately and storing the meanings of verbs separately; here, it is the meaning of verb ‘yeNe’ or its root ‘ye’.

One faces such problems in the case of machine translation because in this case one has to maintain transparency while teaching machine to translate. Transparency in this case points to the description of each and every minute step. This point can be made clear if we compare the style of working of the human being and the machine.

Again we will consider the sentence, ‘mii yeto’. (‘mii’ /’ main’ / I & ‘yeto’ / ‘aataa huun’ / come) A human translator will immediately translate it into the given target language, say, Hindi or English. Yet the translator may not be able to describe the real and logical procedure of such translation overtly at once. The human translator will claim that the whole string is translated at a time. (This is surveyed by the researcher for 5 non professional translators.)

When one aims to teach a machine to translate the sentence, ‘mii yeto’, (‘mii’ /’ main’ / I & ‘yeto’ / ‘aataa huun’ / come) the person has to describe the process of translation in detail. The very first step is to list the token first and then types present in the sentence. The types are understood as the comparable candidates in natural language processing. The sentence ‘mii yeto’ has two tokens namely ‘mii’ and ‘yeto’. Since there is no repetition of any token in the sentence; it has two types, ‘mii’ and ‘yeto’. The second step is to morphologize the types and to get dictionary head words, say, ‘mii’ and ‘yeNe’ or ‘ye’. (The choice between ‘yeNe’ or ‘ye’ depends on the architecture of the system.). Then one needs to find the dictionary meaning of these head words in the target language. It is followed by the synthesis of thus obtained words in the target language. The words in the target language are arranged as per the norms of ‘word order’ in the target language. At the final step, the morphology module of the target language is applied and the final translation is generated. This process becomes more complex when the machine translates complex sentences. In short, for machine translation very minute steps are necessarily needed to be considered. And so, even disambiguation of a suffix attains a great value.

The discussion on techniques for disambiguation of the suffix ‘to’ covers such topics as the preparation (section 4.1) for disambiguation of the suffix ‘to’ and the methodology (4.2) for the same. Section 4.3 portrays the task of implementation of rules to develop the software that demonstrates the procedure of disambiguation of the suffix ‘to’.
4.1 THE GROUNDWORK FOR THE DISAMBIGUATION OF THE SUFFIX ‘to’

As explained in Section 2, the ‘aakhyaata’ theory was studied carefully to know all the sets of information about TAM-GNP that can be denoted by the suffix ‘to’ in the given context. The various tenses, aspects, moods, genders, numbers and persons that can be denoted by the suffix ‘to’, were listed as shown in table 4.

The Marathi corpus of the size of about one and half million words was acquired from Central Institute of Indian Languages, Mysore. It is a good representative corpus.

This data was analyzed with the help of a computer and a programmer. For the purpose of the analysis of the data, first, the text was tokenized. This yielded the list of words (strings separated by spaces) in the text. Further, from these tokens, types were extracted. Type refers to the word once for its multiple occurrences. For example, the string, ‘the boy and the girl are tall and fair’ has 9 tokens (the, boy, and, the, girl, are, tall and, fair) and 7 types (the, boy, and, girl, are, tall, fair).

These tokens were also sorted from right to left as the suffix ‘to’ occurs at the right-most position of the word. This facilitates the study of the occurrences of the string ‘to’ at the end position of the word. Such study was carried out by examining the context of ‘to’ at the word level. This becomes easier and faster when we consider only those words which contain the string ‘to’.

The verb suffix ‘to’ need not be the closing suffix, that is, the ending part of the word. It can be followed by the emphasizing particles ‘cha’ or ‘hii’. Human beings need not realize all these minute detailed facts while using language or doing translation. But to process or to translate any piece of language-use, the machine needs to know such fine and circumstantial linguistic reality. Here, ‘to know facts’ implies to incorporate facts in the computer program.

There are two types of occurrences of the string ‘to’ as a single word in the corpus. When the string ‘to’ is a token it can mean ‘wo’/ he or ‘taba’/ at that time as per the context. It is a verb suffix when it appears at the ultimate or penultimate position of any word. Here and henceforth, the translation of any Marathi string will be given in the order of Hindi / English. Thus the string ‘to’ can be the pronoun or the conjunction when it occurs as an individual word and it can be the verb suffix when it closes the word generally.

It was assumed that the meaning-defining context (key words) of the suffix ‘to’ can be found within its sentence. Here, by ‘meaning-defining context (key words)’ we mean that linguistic entity in the context which defines the exact meaning of the term in question. Each sentence containing the string ‘to’ was listed separately. Within these sentences the string ‘to’ was high-lighted. At first, sentences containing all ‘to’, were collected. Then the suffix ‘to’, was searched for in
the sentences and marked, and the sentences containing the suffix ‘to’ (special ‘to’) were collected separately.

For this purpose, a few unfinished rules were applied heuristically. Usually the suffix ‘to’ appears at the end of words as well as at the end of sentences. The ultimate position in the word and the penultimate position in the word which is followed by the string ‘cha’ or ‘hii’ were the criteria for searching for such a special ‘to’. Additionally, the suffix ‘to’ which is followed by full stop or the words ‘na’, ‘naa’ or ‘to’ and conjunctions like ‘towara’ are treated as special ‘to’. These criteria for searching for the suffix ‘to’ were obtained by a primarily broad analysis of the corpus. The collection of sentences containing the suffix ‘to’ provided the scientific basis for forming the rules for disambiguation of the suffix ‘to’.

4.2 THE METHODOLOGY FOR THE DISAMBIGUATION OF THE SUFFIX ‘to’

The methodological procedure of disambiguation of the suffix ‘to’ starts with first deciding the labels for various tenses, aspects, moods, genders, numbers and persons (TAM-GNPs) in question. Table 4 in Section 3 of this chapter provides the list of such TAM-GNPs related to the suffix ‘to’. The labels for each bit of information about TAM-GNPs are fixed as shown below.

- Tense: Present (Pr), Past (P), Future (F)
- Aspect: Habitual (H), Simple(S), Imperfect (I), Perfect (P), Immediate (Im)
- Mood: Indicative (I), Promise (P), Motive (M), Conditional (C)
- Gender: Masculine (M), Feminine (F), Masculine-Feminine (MF)
- Number: Singular (S), Plural (P)
- Person: First (1), Third (3)

Here are some sentences which exhibit all tenses, aspects, moods, genders, numbers and persons in their possible combinations. The labels are arranged in the order of aspect, tense, mood, gender, number and person to form a tag.

- ‘mii yeta hoto’. < IPIMS1> (Marathi - M)
  ‘main aah raahaa thaa’i (Hindi)
  I coming was (I was coming).
- ‘mii udyaa yeto’. < ImFIMS1> (M)
  ‘main kala aauungaa (aataa_huun)’i (H)
  I tomorrow will come (I will come tomorrow).
- ‘to yeto aaha’. < IPrMS3> (M)
  ‘wo aataa hai’l (H)
  He coming is (He is coming).
- ‘to udyaa yeto mhaNaalaa’. < ImFIMS3> (M)
  ‘wo kala auungaa bolaa’l (H)
  He (he) tomorrow will come said
  (He said, ‘he will come tomorrow’).
- ‘aamhii nehamii yeto’. <HPrIMFP1> (M)
  ‘hama hameshaa aaten hain’l (H)
We always come. (we come always.)

- ‘aamhii mulii yeto’. <SPrIFP1> (M)
  ‘hama laDakiiyan aati hai’ (H)
  We girls come. (we girls come.)
- ‘aamhii mulamula yeta hoto’. <IPIMP1> (M)
  ‘hama laDaker laDaker aaten the’ (H)
  We boys (together) coming were
  (We boys were coming together).
- ‘aamhii mulii nakkii yeto’. <ImFIFP1> (M)
  ‘hama laDakiiyaan jaruura aayengii (aati hai)’ (H)
  We girls definitely will come
  (We girls will definitely come).
- ‘mii udyaa nakkii yeto’. <ImFPMS1> (M)
  ‘main kala jaruura aauungaa (aataa_huun)’ (H)
  I tomorrow definitely will come
  (I will definitely come tomorrow).
- ‘to nehamii yeto’ <HPMMS3>, mhaNuuna mii nehamii yeto <HPIMS1>’. (M)
  ‘to nehamii yeto’ <HPMMS3> mhaNuuna mii (M)
  ‘wo hameshaa aataa hai isa liye main’ (H)
  He regularly comes so I
  nehamii yeto <HPIMS1>’. (M)
  hameshaa aataa huun’ (H)
  regularly come
  (‘wo hameshaa aataa hai isa liye main hameshaa aataa huun’) (H)
  (He comes regularly so, I come regularly).
- ‘mii yeto <PPrCMS1> tewhaa to jaato’. <HPrIMS3> (M)
  ‘main aataa huun tabhii wo jaataa hai’ (H)
  I come / reach that time he goes
  (He goes when I come / reach).

Because of such a fixed order, duplication in the names of labels at different positions does not affect transparency, and clarity in assigning tags as well as analyzing them. The tags listed in chart 1 are used while developing software for the illustration of disambiguation of the suffix ‘to’.

<table>
<thead>
<tr>
<th>IPrIMFPI</th>
<th>IPrIMS3</th>
<th>IPIMS1</th>
<th>IPIMFP1</th>
<th>SPPrIMS3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPIMS1</td>
<td>PPIMFP1</td>
<td>IPrIMS1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chart 1: Tags for the Suffix ‘to’

As discussed in section 4.2, the processed corpus of sentences containing the suffix ‘to’ was the scientific base for the study of disambiguation of the suffix ‘to’.
Each sentence of the corpus (approximately sentences from 50 random files) was analyzed and the meaning-defining context for the suffix was underlined manually. Simultaneously the appropriate tags were also jotted down. The relation between the contexts and the tags was established to extract the rules from the corpus. Rules were written in the form as listed in chart 2 above.

Consider the rule, `<आम्नी> ('aamhii') <----- > <होलो> ('hoto') <PPIMFPI>`. This rule contains four segments of information. In above rule they are `<आम्नी>' ('aamhii'), `<----->', `<होलो>' ('hoto') and `<PPIMFPI>' . These segments are marked by boundaries as `< ' and '>' . The first segment of the rule refers to the meaning defining context, `<आम्नी>' ('aamhii') . The second segment refers to the part of the sentence that exists in between the meaning defining context and the verb form that includes the suffix ‘to’. It is marked as `<----->' by default, as there cannot be a predefined string that exists in between the meaning defining context and the verb form that includes the suffix ‘to’. The third segment refers to the verb form in the sentence that includes the suffix ‘to’, say, `<होलो>' ('hoto') . The fourth segment refers to the tag that carries the information about TAM-GNP, which the suffix ‘to’ carries in that sentence. The individual labels present in each tag refer to a particular bit of information as explained above in the same section. In this rule, the labels in the tag ‘PPIMFPI’ refer to perfective, past, indicative, masculine-feminine, plural and first person respectively. As a matter of convenience, through practice, the order of labels in the tags is maintained as aspect, tense, mood, gender number and person.

Thus the preparation for disambiguation of the suffix ‘to’ is complete. All these efforts make it possible to produce a functioning data-processing system for disambiguation of the suffix ‘to’.

Chart 2: Rules for Tagging the Suffix ‘to’
4.3 IMPLEMENTATION

Implementation in computer science refers to various steps involved in producing a functioning data-processing system or control system from design. The pre-requisites for a successful implementation are actions such as defining the task, writing algorithm, drawing a flow chart, designing modules as per the algorithm and the flow chart, writing code as per design, testing the program to make it ready for its implementation in real life.

4.3.1 The Definition of the Task for Implementation

The task was defined as below.

“Tokenize the given text and sort the corresponding types from right to left as well as from left to right along with their respective frequencies.” AND

“To develop a data-processing system that can identify and tag a verb form in a sentence in running text according to rules in the data-base.”

4.3.2 Algorithm

The algorithm for the task defined as above was written as shown in chart 3 below.
4.3.3 The Flow Chart

The above algorithm can be graphically explained by sketching a flow chart as shown in figure 1 below. The flow chart specifies the sub-tasks.
4.3.4 Design for Implementation

According to the algorithm for implementation as explained graphically in the flow chart, the functioning data-processing system for implementation of the defined task as above is designed. The design specifies various modules such as InPut, Search, All तो (‘to’), Special तो (‘to’), Token, Tagging, OutPut1, OutPut, and DataBase. These modules constitute the system.

The architecture of the system can be sketched as shown in figure 2 below.
4.3.5 Building the System

While building the functioning data-processing system for implementation of the defined task as above various steps are followed. The graphical representation of the description of these steps is presented in Figure 3 on the next page. A.

A. At the first step, the files containing Marathi text are taken as input. These input files (\textit{InPut A}) should be placed in the bin directory, inside the folder named Marathi, which is present in the project’s folder. Our first step includes reading these input files. Here, reading means browsing or uploading the files.

B. At the second step, the string ‘तो’ (‘to’) is searched in the input files (\textit{InPut A}). All the sentences containing the string ‘तो’ (‘to’) as a single word and at the ultimate, the penultimate and the middle position of the words and the sentences are searched.

Figure 2: Architecture of the System for Disambiguation of the suffix ‘to’
The output (OutPut A) of the search is directed to another file which is present in the bin folder holding the name 'all to.doc'. The output is also displayed in the rich text box in the project form (the GUI, Graphical User Interface).

The OutPut A files have the statistics about the total number of sentences in InPut A, total number of sentences containing ‘तो’ ('to') in InPut A and the total number of words containing ‘तो’ ('to') in InPut A. Additionally, the sentences containing ‘तो’ ('to') are printed along with the search letter ‘तो’ ('to') highlighted in the OutPut A file. Another file containing the same output is generated (outputnew) as text file (.txt extension), so that, it can be given as input for tokenizing.

C. The next step is to tokenize the words in a file. The input to this is the file containing all ‘तो’ ('to')s (outputnew). The input file should be
placed in the ‘token’ folder in bin. Each word is tokenized and the total number of tokens is counted along with the number of types. The output of the token and the statistics is directed to another file called ‘frequencyoutput’. Only the Types are sorted (left to right and right to left) and saved in the file ‘sort.doc’.

D. After searching for the string ‘तो’ (‘to’) in all forms, the next step is to search the string ‘तो’ (‘to’) according to some rules. The ‘तो’ (‘to’)s that are the suffixes in the words are searched for. The ‘तो’ (‘to’)s which are in the middle of the words are also searched for provided the succeeding letters are ‘च’ (‘cha’) & ‘षी’ (‘hii’). The ‘तो’ (‘to’)s followed by ‘न’ (‘na’) and ‘ना’ (‘naa’) are also searched for.

The output (OutPut A) is directed to another file which is present in the bin folder bearing the name special to.doc which is in the document format (.doc extension).

E. The next step is to tag the sentences in a file according to given rules saved in a Database. The input to this step is the file containing the search results of particular ‘तो’ (‘to’)s (outputnew1). The output file of this step should be placed in the Tagged folder in bin.

The output is directed to another file which is present in the bin folder holding the name OUTRULE.doc which is in the document format (.doc extension).

Figure 4: GUI for the System for Tagging and Tokenizing
The task is complete after testing the program on some testing data files till expected results are obtained. The software thus ready to implement is accessed by the users through GUI, the graphical user interface. Its snap shot is given above.

4.3.6 GUI, Graphical User Interface

The details of GUI are given below. ‘FORM’ refers to appearance of GUI.

FORM 1 –
Here a brief description about the project is given. There is a link to continue – “CLICK HERE”.

FORM 2 –
After clicking “CLICK HERE” at FORM 1, a form appears in which ‘Enter the Character’, ‘Select the input file’ and ‘Select the language’ options are shown.

Enter the character (window) –
Here one can enter the character which is to be searched from the given input file.

Select the input file (window) –
One can write the name and the path of the input file or brows it.

Browse (button) –
Here one needs to select the input file. The input file should support the Unicode. This browse button helps in selecting the file.

Select the language (window) –
Here one needs to select the language. Here it is Marathi language.

Search (button) –
Once the key character(s) to be searched for is entered and the input file is selected, clicking the ‘Search’ button facilitates searching the entered key character(s) from the input file and highlights that particular character(s).

Clear (button) –
It clears the contents in the text area when this button is clicked.

Token (button) –
The Output of search is taken as Input. Each sentence is split into tokens. This button will give right to left sort, left to right sort and also word frequency by clicking this button each time.

Tagged (button) –
According to the rules specified in the database the words are tagged.

Back (button) –
The control will be transferred from form2 to form1 and vis-à-vis by clicking this button.

Exit (button) –
It is used to end the process.

Text Area (window) –
There is an untitled window which shows the text as the result of any operation.

Developing this system for tagging and tokenizing was not an easy task. Several problems were faced during the development.
4.3.7 Problems in Building the System for Tagging and Tokenizing

Problem 1 --

This task is related to natural language processing and the analysis for developing rules for tagging was based on a corpus. There is always the possibility of getting new rules even after one feels that there won’t be any rule. This makes it essential to keep the repository open to edit and enhance the repository of rules as well as data. This fact was not considered at the beginning of designing this system. Rules for tagging were encoded within the code of the software. This made it clumsy and difficult to edit or enhance the repository of rules for tagging. The older code was also bigger (over 2000 lines) than the recent one (1623 lines). The solution to this problem was to keep the repository of rules for tagging outside the code; that is, in the DataBase. This allowed the code-writer to edit or enhance the repository of rules for tagging at any point.

At this point it will be interesting to see the difference in the format of the DataBase (also known as knowledgeBase) and that of the RuleBase. The DataBase is used to store any type of data, however huge it may be. It provides an option for deleting and editing the existing data and inserting new data. It is in tabular form which is available in ‘built in function’ that is provided in / by the software developer. The code returns and can call the required information from the DataBase. Any rules stored in the DataBase can be called and returned as a function in the code. While each rule in the RuleBase is necessarily needed to be encoded separately in the code and the repetitive use of a given rule in the code needs repetition of the same lines in the code. And so, it becomes bigger in size.

Problem 2

At the step of sorting of types, ‘left to right’ sorting was easy compared to right to left sorting due to the complexity of the Devnagari script. It was made easy due to a novel approach. The strings of all types in question were reversed in the order of characters in the string. For example, consider the strings of types, वाद (वा-द) (‘waada’ (‘waa-da’)) and आपण (आ-प-ण) (‘aapaNa’ (‘aa-pa-Na’)). These are reversed as दावा (द-वा) (‘dawaa’ (‘da-waa’)) and णपा (ण-प-ा) (‘Napaaa’ (‘Na-pa-aa’)). All such reversed strings are sorted from left to right (1> णपा (‘Napaaa’), 2> दावा (‘dawaa’)). Again all strings are reversed keeping the same order of the sorting (1> आपण (‘aapaNa’), 2> वाद (‘waada’)). Thus the sorting, from right to left, of types is achieved.

Problem 3

To develop a search engine is a standard job for computer programmers. Highlighting the result of a search can be done using a “rich text” property. It
was difficult to differentiate the search for all ‘तो’ (‘to’)s and particular ‘तो’ (‘to’)s and to highlight them in different ways in the same given file using “rich text’ property. For tagging purpose, special ‘तो’ (‘to’)s are important. These ‘तो’ (‘to’)s were then made distinct. This is achieved by applying rules stating constraints on the existence of special ‘तो’ (‘to’)s instead of applying the “rich text’ property’. These constraints had to do with the spaces preceding and following special ‘तो’ (‘to’)s or the position in the word and in the sentence or the following characters like ‘च’ (‘cha’) & ‘ही’ (‘hii’) within the word and ‘न’ (‘na’) & ‘ना’ (‘naa’) in the sentence.

Problem 4

To make this system user-friendly, the earlier approach for uploading files was changed. Uploading means ‘making file(s) available for any operation / processing’. Formerly, ‘n’ number of files at a time were possibly made available for tagging and tokenizing. But, for this purpose the location of such files was necessarily fixed by the writer of the code and each user of this system needed to know about this location prior to the use of the system. Further, there was no possibility of processing file(s) by choice. But in the modified version, the facility for browsing a file for tagging was embodied in the program and thus the system was made user friendly.

Problem / Issue 5

There were two possibilities of designing this system or any system, namely, top-down and bottom-up. Top-down and bottom-up approaches are the strategies of information-processing and knowledge-ordering, typically in software, but also in other humanistic and scientific theories. In practice, they can be seen as a style of thinking and teaching. In many cases, the top-down approach is used as a synonym of analysis or decomposition, and the bottom-up approach as that of synthesis.

A top-down approach is essentially breaking down a system into its compositional sub-systems. In a top-down approach an overview of the system is first formulated, specifying but not detailing any first-level subsystems. Firstly, the main function, called ‘stubs’, is written, and then these stubs are further subdivided into still smaller stubs in a greater refined detail until the entire specification is reduced to base elements. Eventually, the components are specific enough to be coded and to write program.

In the bottom-up approach the individual base elements of the system are first specified in great detail. The small components of the program are written first and these component elements are then linked together to form larger subsystems, until a complete top-level system is formed. This strategy often resembles a ‘seed’ model whereby the small beginnings eventually grow in complexity and completeness. The bottom-up programming approach is common in object-oriented languages such as C++ or Java. The bottom-up approach largely depends on
intuition while deciding the functionality that is to be provided by the module. This approach is suitable if a system is to be built from an existing system as it starts from some existing modules.

Top-down programming is advantageous in many ways. It separates the lower level and the higher level work objects, so, it leads to a modular design. Modular designing refers to the capability of self contained development. Such modular designing reduces errors as each module has to be processed separately. It is also less time consuming.

The system for tagging and tokenizing which is basically an analysis oriented task was preferably developed with the top-down approach which is reflected in the Flow-chart (see figure 1). Knowing its advantages as above it was made sure that the system will perform with the reduced errors when the top-down approach is its basis. The top-down approach is followed while developing this system.

4.4 EVALUATION OF THE SYSTEM

The evaluation of any system that processes linguistic data can be done in two ways. Statistically it is done by finding its recall value and precision value. The other way is to do error analysis.

4.4.1 Recall and Precision

'Recall' measures how well a search system finds what you want, and 'Precision' measures how well it weeds out what you don’t want. It is easier to illustrate these concepts than to explain as shown in chart 4 below.

\[
\text{Recall} = \frac{\text{Number of relevant objects found}}{\text{Total number of relevant objects that exist}}
\]

\[
\text{Precision} = \frac{\text{Number of relevant objects found}}{\text{Total number of objects found}}
\]

Chart 4: Recall and Precision

134 = total number of relevant tags that exist
126 = total number of tags found
109 = number of relevant tags found
Recall = 109/134 = 81.34%
Precision = 109/126 = 86.50%

Chart 5: Statistics of Recall and Precision
This system is mainly meant for tagging for disambiguation of the suffix ‘to’. The statistics of the testing random five files from the corpus is given in chart 5.

4.4.2 Error Analysis

The details in the chart 5 show that when the test-files were processed by implementing the software eight ‘तो’ (‘to’)s were found which were inflectional verb suffixes but they were not tagged at all during the processing. It is found during the analysis of this result that all these ‘तो’ (‘to’)s were not tagged for a single reason. Each instance of untagged ‘तो’ (‘to’) was the example of the string which contains two words with a full stop in between without any space (e.g., शक्यतो.मालाच्या (‘shakyato.maalaachyaa’)). This error can be avoided by pre-editing and/or spellchecking the text prior to tagging.

Further, there were seventeen ‘तो’ (‘to’)s which were not tagged properly. Even this inappropriate tagging was due to a single reason. All instances of wrongly tagged ‘तो’ (‘to’)s were examples of ‘तो’ (‘to’) which is tagged with a single tag, namely, <IPrIMS3>. The condition for this tag is the existence of the string of the type, ‘**त **तो’ (**ta ** to’). The suffix ‘त’ (‘ta’) is also ambiguous. It can be either attached to a verb stem or to a noun stem. When this tagging system is developed for all words and all morphemes, this error will not occur as the suffix ‘त’ (‘ta’) would also be disambiguated then. Additionally there are words which end with ‘त’ (‘ta’) like गुलगुलीत (‘guLaguLiita’). This reason can be waived off by listing all head words which is expected to be done in the case of the system for tagging all words.

There was an interesting exception. The string ‘खाऊन कक्षात्येतो’ (‘khaauuna kakshaatayeto’) is actually made up of three words, ‘खाऊन कक्षात्येतो’ (‘khaauuna kakshaata yeto’). But in the text it appeared as ‘खाऊन कक्षात्येतो’ (‘khaauuna kakshaataayeto’). Since ‘कक्षात्येतो’ (‘kakshaatayeto’) is considered as a single word by the system it returned the correct tag, <SPrIMS3> as it was considered by the system that the preceding word is not ending with ‘त’. If it would have been ‘खाऊन कक्षात येतो’ (‘khaauuna kakshaata yeto’) then the system would have returned the tag, <IPrIMS3> which would have been wrong.

It is worth noticing here that the system is built only on the basis of rules and not with the help of any kind of dictionary. It may be claimed that if the proper dictionary is used, all these errors will disappear.

4.5 APPLICATION OF THE SYSTEM
Thus the system can tag the suffix appropriately. Once the suffix and so the verb form is tagged, one can choose the appropriate meaning of the verb form. We will consider the sentences, ‘to yeto’ / ‘wo aataa hai’ / he comes and ‘miī udyaa yeto’ ‘main kala aauungaa (aataa huun)’ / I will come tomorrow. The tags for ‘to’ in ‘yeto’ are <SPrIMS3> and <ImFIMS1> respectively. So the translation of ‘yeto’ will be ‘aataa hai’ / ‘comes’ and ‘aauungaa’ / ‘will come’ respectively. Thus, the ambiguity in Marathi verb form existing due to an ambiguous inflectional verb suffix can be successfully resolved.

CONCLUDING COMMENTS

The suffix ‘to’ is an example of a Marathi verb suffix which is inflectional. It is complicated but not impossible to teach a machine to learn to disambiguate such a small string as ‘to’.

As the suffix ‘to’ is a functional verb suffix, various types of information about TAM-GNP loaded on the suffix ‘to’ are identified properly. Then one can proceed to identify the particular information about TAM-GNP coded in the suffix ‘to’ in the specific context. The rules which relate such information and its context lead to disambiguation of the suffix ‘to’. So, at first, the sets of information which are possibly expressed by the suffix ‘to’ are listed. Secondly, we specify such sets of information expressed by the suffix ‘to’ in the given context in the sentence / text.

FUTURE

This is squirrel’s work in the field of word sense disambiguation. We hope that researchers will take due interest in this field to develop more advanced applications. The ultimate aim can only be the fully automatic machine translation of any kind of text from any natural language to any other natural language.

Thanks!