TDIL PROGRAMME
VISION TO MISSION
ON
ANUVADAKSH MACHINE
TRANSLATION SYSTEM

Ministry of Communication & Information Technology
Department of Information Technology
Electronics Niketan, 6, CGO Complex, New Delhi - 110003
Telfax : 011-24363525 E-mail : tdilinfo@mit.gov.in Website : http://tdil.mit.gov.in
Dear Readers,

e-Content is the prime mover of internet growth. The major population of India needs to connect with internet through Indian Languages to become e-Citizen for which availability of multi-lingual e-content in Indian Languages is the major issue. The Machine Translation Technology can expedite the process of generation of such content. TDIL Programme through its concerted research efforts over least 10 – 15 years has catalyzed a major research initiative in this direction i.e. Anuvadksh. It is a hybrid Machine Translation System which leverages multiple Machine Translation paradigms to get the best results for the user.

Translation is a complex human task in which the consistency of translation across various translators is also a challenge. The artificial intelligence techniques used by MT researchers thus have a limitation to mimic the human mind and its creativity. Thus machine-aided Translation output produced by such systems needs to be post-edited by human to bring it to a user acceptable level. The current issue brings out the technology challenges faced in this area, the architecture design to integrate various MT engines to throw the best translation to the user. In addition, the efforts were made for user evaluation of these MT systems for comprehension and fluency. The efforts of the researchers are laudable to convert it to the web service and to take it to the users through TDIL Data Centre (http://www.tdil-dc.in/) for translation from English – Hindi, Bodo, Gujarati, Bangali, Malayalam, Marathi, Oriya, Punjabi, Tamil, Telugu and Urdu. The future is to package the technology into translation tools useable by the translation community.

Editorial Team

Swaran Lata - slata@deity.gov.in
Manoj Jain - mjain@deity.gov.in
Vijay Kumar - vkumar@deity.gov.in
Somnath Chandra - schandra@deity.gov.in
Bharat Gupta - bharatg@deity.gov.in
DC Verma - dcvdit@gmail.com

Technology Development for Indian Languages Programme

www.tdil.mit.gov.in
www.tdil-dc.in
www.ildc.in
www.w3cindia.in

ISSN No. 0972-6434 has been granted to VishwaBhartattdil. Google search engine refers to the contents of the Journal Website: www.tdil.mit.gov.in

Cover Design Shri Narendra Srivastava
1. Enabling Linguistic Idiosyncracy in ANUVADAKSH  4-27
2. Anuvadaksh : An Evaluation  28-41
3. Resources development for English to Gujarati Machine translation system  42-51
4. Multithreaded Implementation of Earley Style parsing algorithm for L-TAG  52-61
5. Anuvadaksh: A Real Integration  62-74
6. Morphology based Factored Statistical machine Translation(F-SMT) system from English to tamil  75-80
7. Transferring of prepositional phrases in English into tamil : A machine learning approach  81-88
8. Text to speech (TTS) in Indian Languages : A survey  89-99
Enabling Linguistic Idiosyncrasy in Anuvadaksh
ENABLING LINGUISTIC IDIOSYNCRASIES IN ANUVADAKSH

Hemant Darbari, Anuradha Lele, Aparupa Dasgupta, Ranjan Das, Debasri Dubey, Shraddha Kalele, Shahzad Alam, Priyanka Jain, Pavan Kurariya

Applied AI Group, Centre for Development of Advanced Computing, Westend Centre 3, 5th Floor, Aundh, New D. P. Road, Pune
E-mail: (darbari, lele, aparupa, ranjan, debasri, shraddha, shahzada, priyankaj, pavank)@cdac.in

Abstract

This paper establishes the matter of imbibing various language peculiarities and target language specific characteristics of Indian languages in Tree Adjoining Grammar framework. Enabling these language specific features in Machine Translation paradigm is a challenge in itself. ANUVADAKSH – a Machine Translation consortium generating texts from English to eight Indian languages on tourism and health domain is developed over a hybrid based translation engine. In this paper, we explained the linguistic or language specific characteristics, divergences and idiosyncrasies that are harnessed through natural language processing.

1. Overview on Machine Translation

As early as 1629, when Rene Descartes proposed Universal Language and idea of Machine Translation came into being. In 1950s fully automated machine translation on Russian sentences were experimented. In late 1980s, statistical and example based Machine Translation was engineered. During the year 1990-91, DIT (Department of Information Technology) of Government of India initiated the TDIL (Technology for Development of Indian languages) project to encourage the Indian language processing in the area of IT. The institutions namely, C-DAC, Pune (MANTRA); NCST (now C-DAC, Mumbai; MATRA); IIIT-Hyderabad (Anusaaraka, and SHAKTI) and IIT-Kanpur (Anglabharati) have taken the Machine Translation System from English to Hindi to greater height by developing applications using cutting edge technology.

2. Overview on ANUVADAKSH

ANUVADAKSH is domain specific hybrid Machine Translation system on Tourism and Health domain. A dynamic and robust instant Machine Translation system based on Ajax technology with multiple features of translation log generation, multiple soft keyboards, ranking of multiple engine outputs, feedback from user, 1000 word text size upload, grammar and spell checking and NLP component module outputs for various researches on NLP. The system is supported by various internet browsers and it is W3C compliant. Phase I of the project was developed on six language pair from English to Hindi, Urdu, Bangla, Marathi, Odia and Tamil and Phase II has inclusion of two more language pairs from English to Bodo and Gujarati. Standard communication protocol of ANUVADAKSH is based on TAG (Tree Adjoining Grammar), SMT (Statistical Machine Translation system) and EBMT (Example based Machine Translation system).

This paper will focus on development of TAG based linguistic features and inventories imbibed in ANUVADASH for eight language pairs concurrently.
3. Coverage and Distribution

Immensely, dynamic corpus possessing 14 syntactic structure types from Tourism domain; and 22 classified and 1 unclassified syntactic structure types from Health domain that are analyzed, synthesized and generated on all 8 language pairs as mentioned in the above paragraph. Corpus coverage sentences in Tourism domain (old: 15,200 & new: 12,000) are 27,200 sentences and Health domain are 15,200 sentences. Following are the pie-charts depicting the coverage and distribution of linguistic patterns and structures on (a) tourism and (b) health domain:

(a) Tourism domain with 14 structure types:

![Frequency of Structure Types in Tourism Domain](image1)

*Fig 1: Pie chart for Structure type distribution in Tourism domain*

Maximum frequency in appositional sentences (11.33%) and minimum frequency in gerund constructions (0.35%) was found in tourism domain.

(b) Health domain with 22 classified and 1 unclassified structure type:

![Frequency of Structure Types in Health Domain](image2)

*Fig 2: Pie chart for Structure type distribution in Health domain*
Maximum frequency in copula sentences (13.73%) and minimum frequency in discourse connector/comparatives (0.09% / 0.006%) was found in health domain.

### 3.1 Linguistic Idiosyncrasy in Tourism & Health

Within the TAG translation engine, 93% output accuracy of POS tagger is directly proportional to the translated output. Each phase, within the translation process, where TAG grammars in source and target data set are complemented/constituted by CSR (context sensitive rules), GDR (grammar disambiguation rule) rules for source side analysis and parsing. Transfer links (form of transfer grammar), derivation mapping and synthesizer rules of target side are mapped with source side data for generation and translated output. The dependencies of tree nodes are linked in LISP notation for multiple syntactic divergences.

### 3.2 Lexicalized Tree Adjoining Grammar in ANUVADAkhSH

Tree Adjoining Grammar [Kroch and Joshi, 1985] is implemented for all 6 language-pairs in EILMT on TAG translation engine. The JAVA based TAG parser translates English documents to Hindi, Urdu, Oriya, Bangla, Marathi and Tamil. The significant feature of this parser is incremental parser that identifies the (a) clause or phrase on the basis of probable declarative clause boundary and, (b) after identifying clause boundary the TAG tree derivation structure identifies probable parent derivation to the nearest child derivation structure to give the final integrated derivational tree to the TAG Generator. The TAG engine is enriched in such a way that it can process the parsing and generation for interrogative sentences, negation, gerundial construction, relative clause construction, and past & progressive participle etc. The pre-processing is controlled by supervised modules such as – syntactic TAG tree disambiguator module with optimized code and database-design written in regular expressions. TAG tree derivation is given in the following sub-sections of all the language verticals.

### 4. Language Specific Feature in various vertical language pairs: Idiosyncratic feature

In the following sub-sections we will explain various language specific peculiarities and features that are computationally transformed into various rules and grammars in pre-processing, parsing and generation phases in ANUVADAKSH.

### 5. Hindi Features in ANUVADAKSH

All 14 structure types as mentioned in the above sections are analyzed through lexicalized tree adjoining grammar framework. Consider an example for a structure comprising of that clause with a PP-initial. Following is the English and Hindi TAG tree derivation diagram:

Input Sentence: For two hundred years, it was the capital of a Muslim dynasty that ruled over a populace that was predominantly Hindu.
Following is the TAG derivation tree in English for That clause:

![TAG derivation tree in English](image1)

For two-hundred years, it was the capital of a Muslim dynasty that ruled over a populace that was predominantly Hindu.

Fig 3: TAG derivation tree in English

Following is the TAG derived tree in English for That clause:

![TAG derived tree in English](image2)

For two-hundred years, it was the capital of a Muslim dynasty that ruled over a populace that was predominantly Hindu.

Fig 4: TAG derived tree in English

Machine Output: दो सौ वर्ष तक, वह एक मुसलमान राजवंश की राजधानी था जिसने एक जनसाधारण को शासन करता जो मुय पर है।
Following is the TAG derived tree in Hindi for That-caluse:

![TAG derivation in Hindi](image)

**Fig 5: TAG derivation in Hindi**

Rule for Noun (Plural) synthesis in direct and oblique forms for consonant or vowel endings for abstract or non-countable nouns and vocatives.

<table>
<thead>
<tr>
<th>Base</th>
<th>Plural (direct)</th>
<th>Plural (oblique)</th>
<th>Vocative</th>
</tr>
</thead>
<tbody>
<tr>
<td>बात</td>
<td>बात</td>
<td>बात</td>
<td>बात</td>
</tr>
<tr>
<td>आशा</td>
<td>आशाओॊ</td>
<td>आशाएॊ</td>
<td></td>
</tr>
<tr>
<td>लड़का</td>
<td>ए</td>
<td>लड़के</td>
<td></td>
</tr>
</tbody>
</table>

Noun and post-positions functional categories and catered in lexicon. Declinable and indeclinable adjectives are resolved in synthesizer.

<table>
<thead>
<tr>
<th>Base</th>
<th>1st Form</th>
<th>2nd Form</th>
<th>3rd Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>सुन्दर</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>अछ</td>
<td>अछ</td>
<td>अछ</td>
<td>अछॆ</td>
</tr>
</tbody>
</table>

Finite verb forms, causative verbs, iterative forms and their conjugated forms are resolved in verb-generator of transfer link module (of Transfer Grammar). Expressive words (ठीक-ठाक, सनसनी, चीप-चीप), paired word (घन-दौलत, रात-दिन) etc., words are catered in phrasing and chunking. Example of causatives and iterative forms are:

<table>
<thead>
<tr>
<th>Verb Class</th>
<th>1st form</th>
<th>2nd form</th>
<th>3rd form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Causatives</td>
<td>बनना</td>
<td>बनाना</td>
<td>बनवाना</td>
</tr>
<tr>
<td>Iterative forms</td>
<td>बात कया करना</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>कोशिश कया करना</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Ergativity feature is catered in TAG tree embedding where sometimes, agreement of verb is with object rather subject.

6. Urdu Features in ANUVADAKSH

All 14 syntactic structures from the corpus has been catered in English to Urdu system. Following is the derivation from wh-clause for TAG description:

Input Sentence: Mumbai was the dowry of Portuguese Princess, Infanta Catherine de Braganza, who married Charles II of England in 1661.

Following is the TAG derivation tree in English for Wh-Clause:

![Fig 6: TAG derivation tree in English](image)

Following is the TAG derived tree in English for Wh-Clause:

![Fig 7: TAG derived tree in English](image)
6.2 Lexical Divergence

Lexical divergence and de-compounding is a major issue in English-Urdu system. For example, the Urdu word ‘پاکستان’ is a compound word and both the constituents ‘پاک’ ‘love’ and ‘ستان’ ‘country’ are two independently occurring words. The function of the infix -e- is to link both the constituents and give possessive meaning. It replaces the semantic genitive کب/کی/کے which cannot occur in compounds. For example:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>Gloss</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>حب الوطن</td>
<td>country-3p.sg. Infix</td>
<td>Patriotism</td>
</tr>
<tr>
<td>POS Tag</td>
<td>N.3p.sg.</td>
<td></td>
</tr>
<tr>
<td>English</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example on de-compounding:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>اگرے کا قلعہ</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Agra fort</td>
<td></td>
</tr>
</tbody>
</table>
6.3 Structural Divergence

A remarkable example for structural divergence for English infinitive clause “to have been born” gets translated in Urdu differently. In Urdu, translation of subordinate clause rather than the original (in English) infinitive clause. For example:

<table>
<thead>
<tr>
<th>Urdu:</th>
<th>جہبہء</th>
<th>تشريف لائہ</th>
<th>متھر</th>
<th>آپ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>بهگوان کرشن</td>
<td>کہ</td>
<td>یقین کیا جانا ہے</td>
<td>کے لے</td>
</tr>
<tr>
<td></td>
<td>N, 3p.sg.hon.</td>
<td>COMPLT,sub clause</td>
<td>Verb-present perfective</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English:</td>
<td>Come to Mathura, where Lord Krishna is believed to have been born.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In another example, the sentence ‘He went to Delhi’ has the PP adjucnt “to Delhi” gets mapped with NP of Urdu.

<table>
<thead>
<tr>
<th>Urdu:</th>
<th>گیا ہی</th>
<th>دبہی</th>
<th>وہ</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS Tag</td>
<td>Verb, past, perfective</td>
<td>N, direct form, sg</td>
<td>N,direct form,sg</td>
</tr>
<tr>
<td>English:</td>
<td>He went to Delhi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Following are the derivation tree in English for the above example,

![Derivation Tree in English](image)

**Fig 9: Derivation Tree in English**

Following are the derived tree in English for the above example,

![Derived tree in English](image)

**Fig 10: Derived tree in English**
Following is the derived tree in Urdu for the above example,

**Fig 11: Derived Tree in Urdu**

### 6.4 Categorical Divergence

Categorical Divergence occurs in the phrasal / sentential levels, while translating a pair of language. In this case, the POS of two or more adjacent lexical categories lose its original categories to become another lexical category. For example:

<table>
<thead>
<tr>
<th></th>
<th>Urdu1</th>
<th>Urdu2</th>
<th>Urdu3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Urdu</strong></td>
<td>بوا</td>
<td>بوا</td>
<td>بوا</td>
</tr>
<tr>
<td><strong>Gloss:</strong></td>
<td>Air-N.</td>
<td>Sea-Adj.</td>
<td>Air-N.</td>
</tr>
<tr>
<td><strong>POS Tag:</strong></td>
<td>Noun</td>
<td>Adjective</td>
<td>Noun</td>
</tr>
<tr>
<td><strong>English:</strong></td>
<td>Sea air</td>
<td>Noun</td>
<td>Adjective</td>
</tr>
</tbody>
</table>

In these examples, this may be seen in case of “سمتی دری، ثحری، دریبئی” - all become Adjectives in Urdu translation.

Sometimes an expression could appear in a single word in English but in contrast, it needs multiple words to denote the same in Urdu. For example:

<table>
<thead>
<tr>
<th></th>
<th>Urdu:</th>
<th>Gloss:</th>
<th>POS Tag:</th>
<th>English:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>埠口</strong></td>
<td>بولنا</td>
<td>To.speak</td>
<td>infinitive</td>
<td>foray</td>
</tr>
<tr>
<td><strong>حملہ</strong></td>
<td>حملہ</td>
<td>All of a sudden, 3p.sg.</td>
<td>N, direct form</td>
<td>Verb</td>
</tr>
<tr>
<td><strong>3p.گرمی</strong></td>
<td>گرمی</td>
<td>Adjective</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this pair of example, the Urdu expression initially takes an Adjective ‘اچبوک’ ‘all of a sudden’, followed by a Noun ‘حملہ’ ‘attack’ and then a Verb ‘پرولہ’ ‘to-speak’.

6.5 Null Divergence

Null Divergence has been handled through hyphen (-) in the Lexicon Database. This Null Divergence may occur in the Subject Existential position of English or the articles of English are not mapped in any of the Urdu words. For example:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>POS Tag:</td>
<td>Verb N, direct form Det.</td>
<td>English:</td>
<td>There was a king</td>
</tr>
<tr>
<td>POS Tag:</td>
<td>Verb, progressive Noun, fem.</td>
<td>English:</td>
<td>It was raining</td>
</tr>
</tbody>
</table>

Here, the English words “there and It’ do not have the mapping of Urdu while translating from Source A to Target B.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>POS Tag:</td>
<td>Noun, direct form Noun, pl</td>
<td>English:</td>
<td>The vast library</td>
</tr>
<tr>
<td>Urdu:</td>
<td>ریزارت مسافرون والا پنج</td>
<td>Gloss:</td>
<td>Resort-3p.sg Stars,3p.pl Five-adj</td>
</tr>
<tr>
<td>POS Tag:</td>
<td>Noun, sg Noun,pl</td>
<td>English:</td>
<td>a five star resort</td>
</tr>
<tr>
<td>Urdu:</td>
<td>رقبہ</td>
<td>Gloss:</td>
<td>Area, 3p.sg</td>
</tr>
<tr>
<td>POS Tag:</td>
<td>Noun, 3p.sg</td>
<td>English:</td>
<td>an area</td>
</tr>
</tbody>
</table>

In these examples, there is no one to one correspondence of the articles of English “the, a & and” from Urdu. In fact, it is being dropped while translating from English to Urdu.

6.6 Reduplicative Divergence

Reduplicative Divergence occurs very frequently in Indo-Aryan languages (north Indian). This is no special to Urdu language, this phenomena has been written off expressively by the theoretical linguists, especially by Dr. Abbi. Complete or partial reduplicative forms are catered through a lexicalization issue (stored as lexemes). For example:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>POS Tag:</td>
<td>Verb, past Adv = partial reduplicative Noun, sg, ergative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English:</td>
<td>He said it quickly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS Tag:</td>
<td>Infinitive Noun,sg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English:</td>
<td>sing Verb</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.7 Honorific Divergence

This phenomenon is not prominent in English, whereas it is quite openly used in Urdu language to make it soft or polite. For example:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>مصاحب</th>
<th>جناح</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gloss</td>
<td>Mr</td>
<td>Mr</td>
</tr>
<tr>
<td>POS Tag</td>
<td>Noun</td>
<td>Noun</td>
</tr>
<tr>
<td>English</td>
<td>Mr.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Urdu</th>
<th>اہلیہ خباتون</th>
<th>بیکم</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gloss</td>
<td>Wife-3p.sg</td>
<td>girl-3p.sg</td>
</tr>
<tr>
<td>POS Tag</td>
<td>Noun</td>
<td>Noun,sg.</td>
</tr>
<tr>
<td>English</td>
<td>Mrs</td>
<td></td>
</tr>
</tbody>
</table>

6.8 Derivation Morphology

In the ANUVADAKSH system, the derivation morphology deals with two processes, one directly put in the database and another is generally generated from the rules, which are given to the morph generator. In this case, [ -en and –ing forms ] are getting generated thoroughly rules, which are implemented in the ANUVADAKSH system. There are some cases, where the [ -en and –ing forms ] are directly put in the database. The inflected forms of the derivational morphology are generated from the base-verb forms and the inflectional parts of the derivational Morphology are attached to it. For example:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>دوڑوے والی</th>
<th>دوڑوے والی</th>
<th>دوڑوے والی</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gloss</td>
<td>Running-adj, base form</td>
<td>Running-adj, inflected form,</td>
<td>Running-adj, inflected form</td>
</tr>
<tr>
<td>POS Tag</td>
<td>ADJ</td>
<td>ADJ</td>
<td>ADJ</td>
</tr>
<tr>
<td>English</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this example, the system will pick the appropriate form of the above examples as required in the usage through the selection Restriction Module.

7. Bangla Features in ANUVADAKSH

Bangla or Bengali is a part of the Indic group of the Indo-Aryan (IA) branch of the Indo-European (IE) family of languages. The language follows a (S)ubject-(O)bject-(V)erb word order in a sentence. Copula verbs are often dropped. There are Nominative, Accusative-Dative, Genitive and Locative cases. Gender is natural and there are no masculine and feminine markers. Bengali is a classifier based language. The classifier system creates morphological alternations based on number, animacy, honorificity, location, specificity and definiteness etc. Following are the Morphological peculiarities catered in ANUVADAKSH.

7.1 Case System

Bangla is characterized by its morphological divergences with presence of the classifier system in the language plays a crucial role in the language's morphology.

Nominative case is usually a Zero marker. For example, as in Ram,
The marker for Accusative-Dative is ‘ke’. Consider,

<table>
<thead>
<tr>
<th>Bangla</th>
<th>রামকে ডাকুন</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>Call Ram</td>
</tr>
<tr>
<td>POS Tag</td>
<td>ram-acc   call-2p-Hon-ImpV</td>
</tr>
</tbody>
</table>

Genitive case marker is -ের (-er). The form alternates with -র (-r) and -এর (-Er). For example,

<table>
<thead>
<tr>
<th>Bangla</th>
<th>রামের বই</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>Ram’s book</td>
</tr>
<tr>
<td>POS Tag</td>
<td>ram-gen    book-nom</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bangla</th>
<th>সীতার বই</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>Sita’s book</td>
</tr>
<tr>
<td>POS Tag</td>
<td>Sita-gen   book-nom</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bangla</th>
<th>মৌঁ এর বই</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>Mou’s book</td>
</tr>
<tr>
<td>POS Tag</td>
<td>Mou-gen     book-nom</td>
</tr>
</tbody>
</table>

Locative case marker is -dc (-e) and the variants are -য় (-y)/ -েে (-te). For example,

<table>
<thead>
<tr>
<th>Bangla</th>
<th>জয়পুচ্ছর</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>At Jaipur</td>
</tr>
<tr>
<td>POS Tag</td>
<td>Jaipur-loc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bangla</th>
<th>বাড়িচ্ছে</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>At home</td>
</tr>
<tr>
<td>POS Tag</td>
<td>house-loc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bangla</th>
<th>রাস্তায়</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>At road</td>
</tr>
<tr>
<td>POS Tag</td>
<td>road-loc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bangla</th>
<th>মুম্বাইচ্ছে</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>At Mumbai</td>
</tr>
<tr>
<td>POS Tag</td>
<td>mumbai-loc</td>
</tr>
</tbody>
</table>

There are no morphological variants for Nominative and accusative case but both genitive and locative markers depend on the ending of the noun it is attached with. This is handled in the system through morph synthesizer via rules.

### 7.2 Classifier System

Classifier is a very significant feature of Bangla. Plurality is expressed in the language by the classifiers -ের (-der), -রা (-rA) and -গুচ্ছ঱া (-gulo). Consider the examples in 1a to 1d:
In 1a and 1b the classifiers attached with boy are -রা (-rA) and -েের (-der) respectively. The difference between the 'boys' in 1a and 1b is direct and oblique. In 1a 'boys' is marked with nominative case whereas in 1b it is accusative. In 1c and 1d the classifier is -গুচ্ছ঱া (-gulo). The difference between 1a, b and 1c, d is the feature human. In other words, the classifier marking plurality changes according to the feature ±animacy or ±human and direct or oblique position.

The singular marker is -টা (-TA)/-টি (-Ti). This also expresses definiteness or specificity.

<table>
<thead>
<tr>
<th>1a.</th>
<th>Bangla</th>
<th>ছেলেরা খেলছে</th>
<th>Equivalent in English</th>
<th>Boys are playing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b.</td>
<td>Bangla</td>
<td>ছেলেদের ডাকুন</td>
<td>Equivalent in English</td>
<td>Call the boys</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>boy-pl-nom play-3p-NHon-pres-prog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c.</td>
<td>Bangla</td>
<td>কুকুরগুচ্ছ঱া ঘুচ্ছে</td>
<td>Equivalent in English</td>
<td>The dogs are roaming</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>dog-pl-nom roam-3p-NHon-pres-prog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d.</td>
<td>Bangla</td>
<td>কুকুরগুচ্ছ঱া ডাকুন</td>
<td>Equivalent in English</td>
<td>Give food to the dogs</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>dog-pl-nom food-nom give-2p-Hon-ImpV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In 1a and b the classifiers attached with boy are -রা (-rA) and -েের (-der) respectively. The difference between the 'boys' in 1a and 1b is direct and oblique. In 1a 'boys' is marked with nominative case whereas in 1b it is accusative. In 1c and 1d the classifier is -গুচ্ছ঱া (-gulo). The difference between 1a, b and 1c, d is the feature human. In other words, the classifier marking plurality changes according to the feature ±animacy or ±human and direct or oblique position.

The singular marker is -টা (-TA)/-টি (-Ti). This also expresses definiteness or specificity.

<table>
<thead>
<tr>
<th>2a.</th>
<th>Bangla</th>
<th>ছেলেটি খেলছে</th>
<th>Equivalent in English</th>
<th>the boy is playing</th>
</tr>
</thead>
<tbody>
<tr>
<td>2b.</td>
<td>Bangla</td>
<td>ছেলেটি কে ডাকুন</td>
<td>Equivalent in English</td>
<td>Call the boy</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>boy-sg-nom play-3p-NHon-pres-prog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2c.</td>
<td>Bangla</td>
<td>কুকুরটা ঘুরে বেড়াচ্ছে</td>
<td>Equivalent in English</td>
<td>The dog is roaming</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>dog-sg-nom roam-3p-NHon-pres-prog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2d.</td>
<td>Bangla</td>
<td>কুকুরটা ডাকুন</td>
<td>Equivalent in English</td>
<td>Give food to the dog</td>
</tr>
<tr>
<td></td>
<td>POS Tag</td>
<td>dog-sg-nom food-nom give-2p-Hon-ImpV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The nominative marker in 6a and 6c is zero. In 6b and 6d the accusative marker follows the classifier.

In the ANUVADAKSH system these morphological nuances are taken care of through morphological rules, lexical features and lexical lookups. To generate the output for 5a the system will check the lexical feature of the plural noun and its position (subject/object) in the sentence. Accordingly the appropriate rule will be applied.

### 7.3 Honorificity

In verbal agreement, Bangla shows no number or gender agreement but Honorificity is present. Consider the following examples,

---

18 | Page
In the Lexicon the features for each lexical item is present. েেচ্ছ঱রা 'he' in the lexicon has the feature [+Hon]. Appropriate PNG feature of the noun is generated by mapping the PNG feature of the subject/ object noun in the lexicon.

### 7.4 Copula Drop

In copula dropping, Bangla is featured by copula dropping. For example:

<table>
<thead>
<tr>
<th>Bangla</th>
<th>জয়পুর একটি ভ্রমণস্থান</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent in English</td>
<td>Jaipur is a tourist place</td>
</tr>
<tr>
<td>POS Tag</td>
<td>JaipurNom Copula tourist-placeAcc</td>
</tr>
</tbody>
</table>

This phenomenon is handled in the system through lexicon. Implementation of various structures types in English to Bangla in TAG derivations are described below:

**Following is the example of derivation tree from copula in English:**

![Fig 12: TAG derivation in English](image-url)
Following is the example of derived tree from copula in Bangla:

![Fig 13: TAG derivation in Bangla](image)

Following is the derivation tree in English for existential nouns:

![Fig 14: TAG derivation in English](image)
Following is the derived tree in Bangla for existential noun:

![Diagram](image-url)

Fig 15: TAG derivation in Bangla

8. Marathi Features in ANUVADAKSH

8.1 Handling of Marathi Nouns (inflected forms)

Marathi being highly agglutinative Indo-Aryan language and gender system is purely intuitive where role of animacy [+/- ANIMATE] is vital in gender assignment. For example:

<table>
<thead>
<tr>
<th>English</th>
<th>Marathi</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottle</td>
<td>बाटऱी</td>
<td>Feminine</td>
</tr>
<tr>
<td>Wood</td>
<td>लाकूड</td>
<td>Neuter</td>
</tr>
</tbody>
</table>

Noun stems in Marathi undergoes a change before attaching to post-positions or case marker. To resolve such issues of stem divergence, nouns in Marathi in ANUVADAKSH is marked a noun class. Till now 17,520 nouns are marked with these noun classes. These also include the multiword expression. Example:

Marathi: मशीद (noun, Gender Feminine, Number Singular, Person III)  
English: Mosque

<table>
<thead>
<tr>
<th>Marathi</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>मशीद</td>
<td>Oblique</td>
</tr>
<tr>
<td>मशीद</td>
<td>Plural</td>
</tr>
<tr>
<td>मशीद</td>
<td>Plural oblique</td>
</tr>
</tbody>
</table>

Marathi: भीक (noun, Gender Feminine, Number singular, Person III)  
English: Alms

<table>
<thead>
<tr>
<th>Marathi</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>भीक</td>
<td>Oblique</td>
</tr>
<tr>
<td>भीक</td>
<td>Plural</td>
</tr>
<tr>
<td>भीक</td>
<td>Plural oblique</td>
</tr>
</tbody>
</table>
8.2 Generation of the gerund

The gerund is a non-finite verb form and in Marathi gerund behaves like noun and undergoes synthesis with the postposition. Gerund synthesis is also implemented in system but needs enhancement as many a time the Part of speech tagger fails to identify gerund. For example,

Input Text: It is an ideal getaway for picnic while traveling between Dalhousie and Chamba.
Machine output: हे डलहौसी आणि चंबा यांना सह तयार करताना सहजीमातील एक आदर्श ठिकाण आहे

Following is the TAG derivation tree of English for Gerund:

![Fig 16: Derivation TAG tree in English](image)

Following is the TAG derived tree of English for Gerund:

![Fig 17: Derived TAG trees in English](image)
Following is the TAG derived tree of Marathi of Gerund:

![Diagram of TAG derived tree in Marathi]

**Fig 18: TAG derived tree in Marathi**

### 9. Odia Features in ANUVADAKSH

#### 9.1 Honorificity

Features of ‘honorific’ and ‘animate’ have grammatical relevancy in Odia. They impose selectional restriction on number inflection and subject-verb agreement whereas ‘gender’ feature has no such role.

<table>
<thead>
<tr>
<th>Odia</th>
<th>Equivalent in English</th>
</tr>
</thead>
<tbody>
<tr>
<td>஬ାଲକ-ମାନେ/*ଗୁଡିକ ଯାଉଛନ୍ତି</td>
<td>boys-PL go-Pr-Prog-3Pl-Anim/Hon</td>
</tr>
<tr>
<td><strong>POS Tag</strong></td>
<td>(n, direct form, plural) (v, present, progressive, third person, plural, animate/honorific)</td>
</tr>
<tr>
<td><strong>English</strong></td>
<td>Boys are going.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Odia</th>
<th>Equivalent in English</th>
</tr>
</thead>
<tbody>
<tr>
<td>ଗାଇ-ମାନେ/*ଗୁଡିକ ଯାଉଛନ୍ତି</td>
<td>cows-PL go-Pr-Prog-3Pl-Anim/Hon</td>
</tr>
<tr>
<td><strong>POS Tag</strong></td>
<td>(n, direct form, plural) (v, present, progressive, third person, plural, animate/honorific)</td>
</tr>
<tr>
<td><strong>English</strong></td>
<td>Cows are going.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Odia</th>
<th>Equivalent in English</th>
</tr>
</thead>
<tbody>
<tr>
<td>ଇଟା-ଗୁଡିକ/*ମାନେ ଯାଉଛି</td>
<td>bricks-PL go-Pr-Prog-3Pl-(Anin)</td>
</tr>
<tr>
<td><strong>POS Tag</strong></td>
<td>(n, direct form, plural) (v, present, progressive, third person, plural, -animate)</td>
</tr>
<tr>
<td><strong>English</strong></td>
<td>Bricks are being carried away.</td>
</tr>
</tbody>
</table>
As demonstrated in the above examples, the choice of plural marker with the nouns and PNG marker in case of verbal conjugation is determined by the ‘honorific’ and ‘animate’ properties of the subject nouns. This sort of language specific properties are represented by means of some features and are kept in the lexical database.

9.2 Negation

In Odia, negation of verb form is a complex phenomenon to generate computationally. The negative particle ‘na’ is merged with the auxiliary, thus creating a negative auxiliary verb. Let us consider the following sentences.

<table>
<thead>
<tr>
<th>Odia</th>
<th>Equivalent in English</th>
<th>POS Tag</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ସେବହିପଢୁଛି</td>
<td>He book read-Pr-Prog-3Sg-(Hon)</td>
<td>(n, direct form) (n, direct form) (v, present, progressive, third person, -honorific)</td>
</tr>
<tr>
<td></td>
<td><strong>ସେବହିପଢୁନାହିଁ</strong></td>
<td>He is reading a book.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ସେବହିପଢୁନାହିଁ</td>
<td>He book read-Pr-Prog-3Sg-(Hon)-Neg</td>
<td>(n, direct form) (n, direct form) (v, present, progressive, third person, -honorific, Neg)</td>
</tr>
<tr>
<td></td>
<td><strong>ସେବହିପଢୁନାହିଁ</strong></td>
<td>He is reading a book.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ସେରାଠାରି</td>
<td>He there is-Pr-3Sg-(Hon)</td>
<td>(n, direct form) (n, direct form) (v, present, third person, -honorific)</td>
</tr>
<tr>
<td></td>
<td><strong>ସେରାଠାରିନାହିଁ</strong></td>
<td>He is there.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ସେରାଠାରିନାହିଁ</td>
<td>He there is-Pr-3Sg-(Hon)-Neg</td>
<td>(n, direct form) (adv) (v, present, third person, -honorific, Neg)</td>
</tr>
<tr>
<td></td>
<td><strong>ସେରାଠାରିନାହିଁ</strong></td>
<td>He is not there.</td>
<td></td>
</tr>
</tbody>
</table>
In the above examples, it can be seen that for negation Odia verbs in the present tense, the auxiliary element /ach-/ in the verb-form is replaced by a negative auxiliary. Again, when used as full verb this auxiliary possess the sense of ‘be’ verb in English, and its negative counterpart /nãh-/ being ‘not be’. Both have allomorphic variants too.

9.3 Copula Drop

In normal conversation and writing in Odia, the copula verb is dropped in an affirmative sentence. However, the negative copula is retained.

**Following is the description of TAG in English to Odia ANUVADAKSH system:**

![Fig 19: TAG derivation in Odia](image)

Thus the example (5) can be written as:

<table>
<thead>
<tr>
<th>Odia</th>
<th>ସେ ଜଣେ ଭ଱ ମଣିଷ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equivalent in English</strong></td>
<td>He a good man is-Pr-3Sg-(-Hon)</td>
</tr>
<tr>
<td><strong>POS Tag</strong></td>
<td>(n, direct form) (det) (adj) (n, direct form) (v, present, third person, -honorific)</td>
</tr>
<tr>
<td><strong>English</strong></td>
<td>He is a good man.</td>
</tr>
</tbody>
</table>
9.4 Existential copula

In the following sentence (7), we can find a verb representing existentiality of the subject. Its negative counterpart (8) also contains a corresponding negative verb-form. This phenomenon, at present, is handled by providing multiple options for Odia expression.

<table>
<thead>
<tr>
<th>Odia</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>ଟେବୁ଱୍ ���େବୁ଱୍ ଉପରେ ଏକ ବହି ରହିଛି</td>
<td>There is a book on the table.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Equivalent in English</th>
<th>POS Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>table on a book</td>
<td>(n, direct form)</td>
</tr>
<tr>
<td>is-Pr-3Sg(-Hon)</td>
<td>(p)</td>
</tr>
<tr>
<td>(n, direct form)</td>
<td>(det)</td>
</tr>
<tr>
<td>(v, present, third person, -honorific)</td>
<td>(v)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Odia</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>ଟେବୁ଱୍ ���େବୁ଱୍ ଉପରେ ଏକ ବହି ରହିଛି</td>
<td>There is a book on the table.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Equivalent in English</th>
<th>POS Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>table on a book</td>
<td>(n, direct form)</td>
</tr>
<tr>
<td>is-Pr-3Sg(-Hon)-Neg</td>
<td>(p)</td>
</tr>
<tr>
<td>(n, direct form)</td>
<td>(det)</td>
</tr>
<tr>
<td>(v, present, third person, -honorific)</td>
<td>(v)</td>
</tr>
</tbody>
</table>

Following is the derived TAG tree in Odia from example 7:

![Fig 20: TAG derivation in Odia](image)

Following is the derived tree in Odia from example 8:

![Fig 21: TAG derivation in Odia](image)
10. Conclusion

Above sections that explained all language specific features reveals the fact that TAG framework has been extensively enriched by linguistic intricacies on Indo-Aryan and Dravidian languages too. The linguistic feature of Bodo and Gujarati are currently being implemented into the system where Bodo is strikingly a classifier based language and Gujarati is a inflectional language. In addition to this, TAG framework is being simultaneously developed for semantic based tree adjoining grammar facilitating the semantic role assignments to the above said framework on English to Indian language translation. These syntactically and morphologically reached feature implementation on the TAG framework has yield translation accuracy (in terms of comprehension and acceptability) in the range of 70% to 80% on an average.

Reference


***
Anuvadaksh:
An Evaluation
ANUVADAKSH : AN EVALUATION

Aphale Neha, Doctor R, Shinde Babita
Centre for Development of Advanced Computing, Pune
{nehaa, rdoctor, babitag}@cdac.in

Abstract:

In order to ensure seamless transfer of Information across languages, TDIL DeitY has initiated various research areas viz. development of machine translation, cross lingual information access system, optical character recognition. This paper presents an overview of the Procedures as well as Methodologies deployed for evaluating one of the many MT Systems: Anuvadaksh. CDAC GIST has been entrusted with the task of developing an evaluation strategy and evaluating the output of the Translations. The paper is divided into 4 parts.

To set things in perspective, Part One of the paper presents a broad overview of the various machine translation systems, all the more so, since the evaluation methodology is common for all the machine translation systems. An overview of Anuvadaksh system along with other existing MT systems is presented.

The Objectives underlying evaluation are given in Part two.

Part three deals with the development and fine-tuning of the machine translation evaluation strategy and the details of evaluation procedure followed.

As a logical corollary, Part four gives a brief analysis of Anuvadaksh system. The next part provides steps for improving the overall translation quality.

It needs to be noted that, this paper deals only with machine translation evaluation strategy & evaluation procedure as pertinent to Anuvadaksh and hence the output of the system in terms of accuracy is not presented.

1. Introduction:

To build machine translation systems different institutes & organizations have been working for several years to overcome the language barriers and have generated a large repository of linguistic tools & resources, pertinent to machine translation. The Department of Information Technology (DeitY), Ministry of Communications and Information Technology, Government of India has taken the initiative and provided them a common platform in consortium mode, through which these machine translation systems are made available to public mass usage through TDIL-DC portal. (http://translation.tdil-dc.in/).

Anuvadaksh MT systems have completed One Phase of development, testing and evaluation (31/12/2007 to 30/6/2010), and are now in their second phase.

India is a multilingual country. There is a great demand for translation of documents from one language to another. This will ensure larger flow of information across different languages.

Anuvadaksh: an English to Indian language machine translation system is part of the machine translation consortia the other three being, AnglaMT, ILMT & Sanskrit to Hindi Translation. Anuvadaksh aims to develop and deploy a machine translation system from English to chosen Indian languages in Tourism and Health domains and supports the following languages.

1. English → Hindi
2. English → Marathi
3. English → Bangla
4. English → Urdu
5. English → Tamil
6. English → Odia
7. English → Gujarati (newly added in phase-II)
8. **English → Bodo (newly added in phase-II)**

Anuvadaksh is developed by EILMT consortium. C-DAC Pune is a Consortium Leader in association with 12 institutes - IIT- Hyderabad, C-DAC Mumbai, IIT- Bombay, Jadavpur - University, IIIT- Allahabad, Utkal University - Bhubaneswar, Amrita- university, Banasthal Vidyapeeth, North Maharashtra University, Jalgaon, Dharamsinh Desai University, Nadiad and North Eastern Hill University, Shillong. This is a multi engine; multi output machine translation system which has TAG based MT (TAG), Example Based MT (EBMT) & Statistical Machine Translation (SMT) systems. Whereas the TAG system supports all the 8 indian languages, SMT supports three languages Hindi, Marathi & Bangla and EBMT supports only Hindi.

**2. Objectives of Evaluation:**

The aim of the evaluation is to evolve a strategy for evaluating output of the different Translation Engines and provide a methodology at the national level for machine translation evaluation, which will focus on linguistic analysis of machine translation system as well as to provide end to end system performance in quantitative measure. There exist international standards for evaluation like BLEU, NIST & Meteor. But these do not cater to Indian language complexities. However in close consultation with consortia leaders it was felt that in the initial stage the tried and tested approach of manual testing be carried out to ensure that the translation system meets the main requirements: usability & native user expectations. Our objective has been therefore to develop in consultation with the Consortia an evaluation procedure which evaluates output of the translation system in terms of comprehensibility, fluency and linguistic integrity also, which can handle the morphological complexities of Indian languages.


Once the objective is set in place, Evaluation in conformity with the Objective needs to be defined. Evaluation is necessarily a two pronged process. Creating a Strategy: A strategy for evaluation needs to be defined in terms of what is being targeted. Implementing the Strategy: Once this strategy is formulated and finalized, a method for implementing the strategy needs to be developed. Each of these will be treated in what follows:

**3.1 Development of a Strategy:**

The Evaluation strategy evolved has undergone mutations in the various stages of its development, as and when successive evaluations have taken place and feedback from the consortia team has been received. In what follows a diachronic development of this strategy is presented. In short the strategy has moved from Linguistic Testing to Sprachgefühl i.e. native speaker’s acceptance of the output in terms of comprehensibility and fluency.

**3.1.1 Phase 1: Linguistic testing**

Initially the strategy proposed was that of Linguistic Testing through a test bed of patterns based on Spelling, Morphology, Syntax, Semantics, lexicology and norms. Stress was also laid on whether the systems produced text which was in compliance with the Akshara theory. A short synopsis of the major parameters of this approach is given below:

- **Orthography:**
  1. Misspellings: Spelling correction: e.g. strenght for strength, dias for dais
  2. Spelling variants: American vs. British: e.g. Organise vs. Organize

1 All examples are provided in roman for intelligibility
3. Abbreviations and Truncations: e.g. it’s, ‘tis I’d ain’t
4. Acronyms: e.g. IT: Income Tax or Information Technology
5. Misused terms: e.g. loose for lose

• **Morphology:**

1. Moving from a low level morphological language to a high level one e.g. ENGLISH to TAMIL will test the translation systems ability to lemmatize data as well as the POS Tagging and parsing “skills” of the MT system.
2. Morphological variants: E.g. Urdu: joining the helper verb or keeping the helper verb distinct kiaagayaa written together instead of separately.

• **Morphotactics:**

1. Suffix ordering: EN->MAR From this very table: tebl+aa+var+caa+c in Marathi EN->GUJ From the boy’s side (with stress): chokr+aa+vaaLaa+o+maaN+thi+ya En->MAL maram+il -> marattil (geminated sandhi)
2. Sandhi: e.g. Vowel harmony in Bangla verbs e.g. khaa vs. khe

• **Syntax:**

1. Handling of word order: SVO->SOV.
2. Use of correct negation: mat and nahiN in Urdu.
3. Handling dummy constructs: e.g. DO in English “I DO not understand”, “DOES he know?” vs.. He DID the work.
4. Anaphora and cataphora interpretation from Source to Target Language.
   a. E.g. It’s well known that Rajasthan has forts. The dog came. Its tail was wagging.
5. IF constructs (conditionals) If I were..., If I win..., If he went..., If he had gone....
6. POS typologies and their translations: Pronouns, Adjectives (Non-qualifiers), Adverbs
7. Negation: He has NOT come, he does NOT know

• **Semantics:**

1. Here stress will be basically on Semantic competence. Some test cases are provided below.
2. Semantic Ambiguity:
   a. E.g. American head seeks arms Identification and explanation: The homograph "head" can be interpreted as a noun meaning either chief or the anatomical head of a body.
3. Lexical Ambiguity:
   a. E.g. Teacher strikes idle kids.
   b. Ambiguity type: Lexical (part of speech or category ambiguity). Identification and explanation: "strikes" can occur as either a verb meaning to hit or a noun meaning a refusal to work.
4. Structural Ambiguity:
   a. E.g. stolen painting found by tree Ambiguity type: Structural.
5. A tree found a stolen painting. A person found a stolen painting near a tree. (* Indicates an unacceptable sentence)

- **Lexicology:**
  1. Borrowing: A word taken directly from another language e.g. Strawberry, Hadran Collider in Target languages Loan vs. Source translation: Installments vs. kishton
  2. Calque: SL foreign word/phrase is translated and incorporated into TL free verse
  3. Transliteration & Named Entity Recognition: Taj Mahal, Eiffel Tower Translation of “function words” such as for: Rajasthan is famous for its castles.
  4. Translation of Phrasal verbs: go in for, go for, go into,
  5. Divalence of POS: yellow (Adj., Noun, Verb)

- **Norms:**
  1. Spelling Norms: Compliance with spelling norms of the respective target languages:
     i. e.g. Urdu: Imlaanaamaa Bangla: Bangla Akademi
  2. Storage Norms: Compliance with Unicode Based on the above parameters the following 7-POINT Russian Grading scale (slightly modified) which deals with translation as a process of visibility of text was proposed. Usability and Transmission of Information is the prime criteria on which this grading scale was based on.
7-POINT Grading scale

1. **Opacity**
   The rendering is absolutely useless for any purpose. Such a rendering shall be deemed as of **UNACCEPTABLE** quality.

2. **Semi-Opaque**
   Some parts are comprehensible, but on the whole the picture still remains difficult to get and the text evades the target user. Such a rendering shall be deemed as **POOR** quality.

3. **Part Visibility**
   The user can get a grasp of the over-all intention of the text, but on the whole, the user has to work hard to get at the meaning of the text and large fragments are practically opaque and incomprehensible. Such a rendering shall be deemed as **LOW** quality.

4. **Half Visibility**
   The rendered text is quite comprehensible to the target reader and can be used by him/her as can be used as a rough draft for improvement. Such a rendering shall be deemed as **DRAFT** quality.

5. **Near Visibility**
   Text is clear enough and all pertinent information can be drawn from it. However, the text is hard to read due to language errors and require further filtration. Such a rendering shall be deemed as of **ACCEPTABLE** quality.

6. **Near-total visibility**
   The rendering has stylistic errors and also some difficult grammatical, syntactic, lexical issues are not clarified. However, it transmits the information needed to the target user. Such a rendering shall be deemed as of **SATISFACTORY** quality.

7. **Total visibility**
   The rendering passes muster though not stylistically perfect. Such a rendering shall be deemed as of **HIGH** quality.

However, as the Anuvadaksh machine translation system is a research project it was felt that these parameters could be applied at a later stage. Moreover it was observed that the more the number of scales, the more will be errors in human judgment and more training is required for human evaluators. Hence this strategy based on linguistic testing was replaced by one based on the native speaker’s evaluation.
3.1.2 Phase 2: Comprehensibility

A new approach was formulated with stress on Sprachgefühl i.e. focus on usability and the native speaker’s expectations and the translation quality was provided in terms of comprehensibility of output. With this in view, the following 4 point grading scale was finalized.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade 0</td>
<td>No output provided by the engine concerned.</td>
</tr>
<tr>
<td>Grade 1</td>
<td>The translated output is not comprehensible.</td>
</tr>
<tr>
<td>Grade 2</td>
<td>Comprehensible after accessing the source text.</td>
</tr>
<tr>
<td>Grade 3</td>
<td>Comprehensible with difficulty.</td>
</tr>
<tr>
<td>Grade 4</td>
<td>Acceptable since the text is comprehensible.</td>
</tr>
</tbody>
</table>

For calculating the ratings, the following formula was deployed:

\[
\text{Final Evaluation Mean For Each Sentence} = \frac{1}{3} \sum_{T=1}^{T=3} \text{Grade Given By Tester T} \\
\text{Total Average} = \frac{1}{S} \sum_{S=1}^{S=100} \text{Final Evaluation Mean Of Sentence S} \\
\text{Accuracy Of Engine} = \frac{\text{Total Average} \times 25}{\text{Number Of Sentences}} \quad (\text{in } \%)
\]
### 3.1.3 Phase 3: Present Evaluation Strategy: Fluency And Comprehensibility

Based on the learning experience from evaluation and suggestions from consortia the grading scale was changed. And this because of the following reason: There was confusion in the above grading scale, where "Grade 2" - "Comprehensible after accessing the source text" was useful only in Open testing and had no significance in Blind testing. This made a difference in results of MT systems. So in consultation with all the MT consortia leaders it was decided to change the grading scale for testing of MT systems when Phase-II was deployed.

Also it was decided that, NO OUTPUT by system and buffer clearance issue will be graded as "-1". As per previous grading scale this was graded as "0" which largely affected the performance of the system. Performance of the systems will be given on two parameters, (a) comprehensibility and (b) fluency.

This led to the following rating system:

<table>
<thead>
<tr>
<th>Grade 0</th>
<th>Nonsense (if the sentence doesn’t make any sense at all - it is like someone speaking to out in al language you don’t know)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade 1</td>
<td>Some parts make sense but is not comprehensible over all (e.g., listening to a language which has lots of borrowed words from your language - you understand those words but nothing more)</td>
</tr>
<tr>
<td>Grade 2</td>
<td>Comprehensible but has quite a few errors (e.g., someone who can speak your language but would make lots of errors. However, you can make sense out of what is being said)</td>
</tr>
<tr>
<td>Grade 3</td>
<td>Comprehensible, occasional errors (e.g., someone speaking Hindi getting all its genders wrong)</td>
</tr>
<tr>
<td>Grade 4</td>
<td>Perfect (e.g., someone who knows the language)</td>
</tr>
</tbody>
</table>

The results are thus calculated on two parameters, (a) comprehensibility and (b) fluency. Both (a) and (b) are be calculated by considering the average of the score given by all the evaluators for every sentence.

(a) Comprehensibility is calculated by taking out the percentage of the number of sentences getting an average score bet 2-4 out of the total number of sentences in the set. Specifically, let \( S_i \) be the number of sentences with a grade of \( i \) (\( i=0, 1, 2, 3, 4 \)). Then comprehensibility \( C = \frac{\text{sum}_{i=2,3,4}(S_i)}{\text{sum}_{all_i}(S_i)} \).

(b) For fluency, the average scores will be measured against \([4 * \text{total number of sentences in the set}]\). Specifically, let \( S_i \) be the number of sentences with a grade of \( i \) (\( i=0, 1, 2, 3, 4 \)). Then fluency \( Q = \frac{\text{sum}_{all_i}(i * S_i)}{[4 * \text{sum}_{all_i}(S_i)]} \).

#### 3.2 Development of an Evaluation Procedure:

Once these testing parameters were set in place, the following step-by-step procedure was adopted. The main features of the evaluation procedure are laid down point wise for the purpose of clarity.
3.2.1 Sampling of Test Data:

Test data for both tourism & health domain was provided by consortia. Out of the provided test data, randomly selected sentences were used for evaluation. For the initial round of testing 100 sentences were used and later number of test sentences were increased to 1000 for extensive testing. All these sentences were randomly selected from the test data provided by consortia. Phase-II evaluation of Anuvadaksh system is carried out on the 100 sentences for tourism & health domain. All these sentences were categorized grammatically structure wise by consortia.

3.2.2 Identification of Evaluators:

Once the data was finalized, the choice of evaluators was determined. This was conditioned by the strategy. As user expectations is a main criteria of machine translation evaluation, instead of linguists 3 to 5 native users were used for evaluation. Since all evaluation of a translation is subjective; to bring in a modicum of objectivity, 3 -5 evaluators were used. Evaluators were selected on the basis of their proficiency in the source and target language. To ensure a broad sampling spectrum, they were chosen from different professions and age-groups.

Thus for English → Marathi, the following sampling of professionals was chosen:

<table>
<thead>
<tr>
<th>English → Marathi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluator ID</td>
</tr>
<tr>
<td>Qualification</td>
</tr>
<tr>
<td>Profession</td>
</tr>
</tbody>
</table>

3.2.3 Training To Evaluators:

Once the evaluators were determined, a workshop for formal training of the evaluators was conducted to make them aware about the grading scale, along with the sample examples of each grading scale. This was followed by a Hands-on Training session to make them familiar with the procedure and also to resolve all doubts. The evaluation was a double-bind test: Open vs. Blind testing. A double-bind testing was proposed i.e. Blind vs. Open testing. Initially the evaluators were supposed to evaluate the output alone in terms of the grading scale and in the second half, propose a grading after seeing both source and target. The grading may change in Open Testing for following two reasons

a. Some translations deemed as incomprehensible will become “comprehensible with great difficulty” i.e. after accessing the source text

b. However it may also be that the translation deemed correct will be proved incorrect, since a wrong choice of word is involved.

E.g. Translation for: I saw the man may be rendered in output as: मैं को आदमी काटा which may be judged as partly comprehensible when in fact it has distorted the meaning of the English and the rating will be changed to Incomprehensible.
3.2.4 Evaluation by Evaluators:

Each evaluator was provided two sheets. One sheet for Blind Testing and one for Open testing. Each evaluator first undertook blind testing and subsequently Open Testing. Along with the testing sheets, an instruction sheet was provided to each evaluator for their guidance and to avoid the confusion in grading the sentences. In case of Blind Testing, the evaluator did not have access to Source sentence. And in Open Testing s/he was provided machine translated output along with the Source sentence. To ensure objectivity, if multiple outputs were obtained from the System, the top five outputs were considered for evaluation, from which the evaluator selected the better machine translated output for each input sentence. The evaluators were given enough time to grade the sentences. Comprehensibility & fluency are calculated on the basis of grading given by evaluators. The grading provided by the evaluators was furnished to the Consortia for their feedback and for bettering the system.

4. Analysis of Anuvadaksh:

Various rounds of evaluation have been carried out with test data provided by consortia for all the language pairs. The quality of output has improved over each round of testing. Our major observations are provided in what follows:

4.1 General Observations:

The Overall performance of Tourism domain is better than health domain. This is possibly because the initial phase was for Tourism and the Health domain was added on later within a comparative perspective, TAG & SMT has better translation quality than EBMT engine. Performance of system in terms of quality of translation is increased from Phase-I, for English to Hindi (for both TAG & SMT engine), English to Bangla (TAG) and English to Urdu (TAG). But the quality of translation is a bit less for English to Marathi (TAG), English to Odia & English to Tamil.

4.2 Syntactic Issues:

The following observations were noted for syntactic evaluation of the data: Irrespective of language, it was found that in Tourism domain, the system performs well for the following sentence structures "Simple, co-ordinate, copula, pp initial, participle, relative clause, that clause"; However, improvement for structure type "Conditional, wh-clause, adverb initial, apposition, discourse connector" needs to be implemented.

In Health domain it is observed that system perform better for the following sentence structure types “Simple, Relative clause, participial/gerund, compound, pp initial, adverb initial, copula, that clause, infinitive constructs, appositional, wh-clause” and needs improvement for the structure type "Discourse connector, imperative, complex, multiple verb, participle constructs, conditional clause".

4.3 Error Diagnostics

On the basis of evaluation, we have analyzed the following most prominent errors. Quality of the translation can be considerably increased if these certain issues are addressed.

1. Ranking Module - As Anuvadaksh is multi engine translation system, if three engines (TAG, SMT & EBMT) are integrated along with the ranking module, better output can be provided to the end user.

2. WSD module - Improvement in WSD module will help to avoid problem of wrong selection of words in target language, where a word has multiple meanings.
3. **Preprocessing Module** – Cases of Acronyms and short abbreviations need to be addressed in the preprocessing module – Thus 12th is glossed as १२वीं, instead of १२वीं.

4. **NER module** – This is a complex area and all MAT systems encounter difficulties in correctly identifying Named Entities. Certain named entities are not identified and transliterated correctly. The quality of transliteration is quite good but if improved will provide a boost to rating.

5. Sometimes Indian language text is not rendered properly
5. Conclusion

The Road Ahead Evaluation in terms of both procedure and strategy and its implementation has been a huge learning experience. Fine tuning the Procedure will better the evaluation approach. From the feedback we have received from the evaluators, it was felt that within the present system more clarity is needed for differentiating Grade 2 and Grade 3. While at present test data is proposed by the consortia for testing he output, in subsequent testing grounds, the test data will be provided by the testing team itself. Since the engines are available for testing on the TDIL-DC website, a large sampling of sentences has been provided by users and its is suggested that this valuable user data be used for bettering the quality of translation and also for testing purposes. At present no module-wise analysis of the output is being tested. Reports for such and evaluation will enable the developers to pinpoint with accuracy which modules are efficient and which needs to be fine-tuned. Machine Translation is a major challenge worldwide and Anuvadaksh’s contribution towards bridging the language divide between English and Indian Languages cannot be understated. Within the perspective of the evaluation carried out, we are confident that the MT System will provide near-native translation output.

6. Acknowledgement

We would like to express our special gratitude and thanks to TDIL-Deity officials for supporting and guiding this activity. We would also like to thank all the consortia leaders of machine translation systems Dr. R.M.K. Sinha, Dr. Rajiv Sangal, Dr. hemanr Darbari and MR. V.N. Shukla for their valuable and constructive suggestions during the evolution of machine translation evaluation strategy, In addition, we would also like to thank chief investigator of the Testing, benchmarking & Evaluation project, Mr. Mahesh Kulkarni for his continuous support and linguistic Resource Cell at C-DAC GIST for getting the evaluators on board.

References:


17. Kay, Martin, Jean Mark Gawron and Peter Norvig; 1994; Verbmobil: A Translation System for Face-to-Face Dialog, Stanford, CA: CSLI.


23. Nida E. 1969, The Theory and Practice of Translation -Brill,


***
Resource Development for English to Gujarati Machine Translation system
RESOURCES DEVELOPMENT FOR ENGLISH TO GUJARATI MACHINE TRANSLATION SYSTEM

C. K. Bhensdadia, Brijesh Bhatt Jatayu Baxi, Kirit Patel and Dinesh Chauhan Department of Computer Engineering, Faculty of Technology Dharmsinh Desai University, Nadiad, Gujarat.

Abstract

This article describes various linguistic resources created to develop English to Gujarati Machine Translation system. The work includes, parallel corpus creation, English-Gujarati Lexicon building and development of grammatical resources such as Transfer Grammar and Morph Synthesizer. The resources are used to develop Tree Adjoining Grammar based English to Gujarati Machine Translation system. The system shows 50% accuracy on gold data.

1. Introduction

Machine Translation refers to build software system which translates text from one natural language to another natural language. In a multilingual nation like India with 22 official languages, it is important to translate and share information across languages. Most of the information available online is in English. In order to make this information available to common Citizen of India, it is desirable to translate this information in vernacular languages. With this aim, English to Indian Language Machine Translation System (EILMT) project is initiated by Department of Information Technology, MCIT, Government of India. Aim of EILMT project is to design and deploy a Machine Translation System from English to Indian Languages. The project started from September 2006. Figure 2 shows abstract view of E-ILMT system. Work for Gujarati language is initiated in 2011. Remaining of the paper is organized as follows, section 2 describes basic overview of Machine Translation system and various approaches for the development of Machine Translation system. Section 3 provides brief description of EILMT system and interface for Gujarati language. Section 4 and 5 describe development of resources for Gujarati language.

Figure 1: Basic Architecture of Machine Translation (Image source: http://www.linguatec.net/products/tr/information/technology/mtranslation)
2. Overview of machine translation

2.1 Various Approaches of Machine Translation

‘A Rule-Based Machine Translation (RBMT) system consists of collection of rules, called Grammar rules, a bilingual or multilingual lexicon, and software programs to process the rules has discussed knowledge based machine translation in which system rely on set of language pair-dependent rules to carry out translation. Shortcoming of this approach is insufficient amount of really good dictionaries.

2.1.1 Rule-based Approach

In this section we discuss various approaches to build Machine translation system describes survey done on various approaches for machine translation system.

2.1.2 Statistical Machine Translation

Statistical machine translation tries to generate translations using statistical methods based on bilingual text corpora proposed an effective frame-work for English-Hindi phrase-based SMT. In this work With only a small amount of bilingual training data and limited tools for Hindi, reasonable performance and substantial improvements over the base-line phrase-based system is achieved. Shortcoming of this approach is Corpus creation can be costly for users with limited resources.

2.1.3 Example Based Approach

‘An EBMT system is given a set of sentences in the Source Language and their corresponding Translations in the Destination Language, and uses those examples to translate other, similar sentences Makoto Nagao proposed this method and pointed out that the Example-based Machine Translation is especially adapted to the translation between two totally different languages.

2.1.4 TAG Based Approach

Tree-adjoining grammar (TAG) is a grammar formalism defined by Aravind Joshi. Tree-adjoining grammars are similar to context-free grammars, but the elementary unit of rewriting is tree rather than the symbol. Whereas context-free grammars have rules for rewriting symbols as strings of other symbols, tree-adjoining grammars have rules for rewriting the nodes of trees as other trees. A TAG based system system has three phases i) Analysis ii) Transfer and iii) Generation. In the first stage source language parser generates syntactic representation of a sentence. In next stage result of first stage is converted into target language oriented representations. In the final step of this translation approach, a Target Language morphological analyzer is used to generate the final Target Language output.

3. E-ILMT System

Figure 2 shows basic architecture of E-ILMT system. Input is first pre processed with help of modules like Morph analyzer, Named entity recognizer, word sense disambiguation etc. After that there are three approaches which are tried in this system: Example based machine translation, Statistical machine translation and tag based translation. For EBMT model, training examples are provided to the system, for SMT appropriate language model is prepared. For tag based approach parser and generator module is implemented. Post processing modules like Morph synthesizer, multiple output selector, Synonym
selector are also implemented. The ranking module is also implemented which gives rank to each output generated with different approaches.

![Figure 2: Basic Architecture of E-ILMT System](image)

### 6. Corpus Development for Gujarati

In order to integrate Gujarati language support in E-ILMT framework various resources and tools are developed for Gujarati language. The existing tourism and health corpus is translated into Gujarati to create parallel corpus. The corpus is XMLized and a bilingual English-Gujarati lexicon is created. The morphological structure of Gujarati is investigated and accordingly morph synthesizer for Gujarati is developed. This section describes various tools and resources created for Gujarati language.

#### 4.1 Translation

<table>
<thead>
<tr>
<th>Domain</th>
<th>No of Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health</td>
<td>15000</td>
</tr>
<tr>
<td>Tourism</td>
<td>12000</td>
</tr>
</tbody>
</table>

*Table 1: Corpus Creation*
The tourism and health corpus which were available in other languages were used for the system development. The first step was to translate English tourism and health corpus in Guajarati language to construct parallel corpus. The sentences are categorized based on their structure into Simple, Complex, Copula, Adverb-Initial, Gerund etc.

As shown in Table 1, we translated 15000 sentences of health domain and 12000 sentences of tourism domain. The sentences are grouped into files each containing 100 sentences. With each English sentence we put corresponding Guajarati translation of the sentence and put it in the same file. Figure 3 shows sample parallel corpus file.

<table>
<thead>
<tr>
<th>S.N.</th>
<th>English Sentences</th>
<th>Gujarati Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A hot Epsom salt bath, twice a week will be highly beneficial in all cases of acne.</td>
<td>अच्छी शरीर या आयुक्त या सम्यक स्थिति में गर्म पानी का सेवन बहुत फायदेमंद है।</td>
</tr>
<tr>
<td>2</td>
<td>A person can, through chronic stress, become sensitive to common foods or commonplace substances like petrol fumes.</td>
<td>एक लोग रुकता है जब तक कि उन्होंने गंभीर तंत्रिका से बनी आकर्षण या गुणों जैसे पेट्रोल की फूंसों के सिम्पल उपजोड़ों को होते हूँ।</td>
</tr>
</tbody>
</table>

Figure 3: Parallel Corpus Creation

4.2 XMlization

E-ILMT project uses XML as a standard language to represent parallel corpus. In order to represent Guajarati corpus into XML, we used XMlization tool developed at Banasthali Vidyapith. The XMlization tool takes parallel corpus file as an input and generates two XML files, one for source (English) language and other for target (Gujarati) language. Figure 4 shows an example XML file.

Figure 4: XML file Creation
4.3 Lexicon Building

The Lexical transfer phase of machine translation finds a target language word for the given source language word. English to Gujarati lexicon is constructed to perform this task. The Linguistic Resource Management Tool (LRMT tool) developed by IIIT-Allahabad is used to construct lexicon.

The LRMT tool opens XML file and displays English and Guajarati sentences parallaly. The words which are not present in dictionary are highlighted. We need to select source word and corresponding destination language word and click Add word button. After we add word, it will be added into database. Figure 5 shows a snapshot of the LRMT tool. Also we can add features with Noun and add synonyms of given noun. Following features are identified and added with each Noun.

- Proper Noun : Indicates weather Noun is proper noun or simple Noun.
- Gender : Indicates gender of Noun. Gujarati has three genders so it can take one of GM,GF or GN value.
- Number : Indicates weather Noun is singular or Plural.
- Person : Indicates person of Noun. Either PI,PII or PIII.
- Animate : Indicates weather Noun is Animate or Inanimate.
- Human : Indicates nature of noun as Human or Non-Human.
- Abstract : Indicates Abstract or Non-Abstract Noun.
- Honorific : Indicates Honorific nature of Noun.
- Temporal : Indicates Temporal or locative nature of Noun.
- Countable : Indicates weather Noun is countable or not.

Figure 5: Lexicon Creation using LRMT Tool We have done lexicon building and feature addition for around 3200 words
5. Tool Development

5.1 Transfer grammar

Transfer grammar module chooses appropriate Gujarati word for the given English verb. English verbs are categorized into 6 parts. Transfer Grammar rules are developed to category wise relates English and Gujarati verbs. In Gujarati, verb shows gender inflection, which is not the case with English. As shown in Table 2, the noun shows no inflection for gender but in Table 3, verb shows inflection for gender male and female. Transfer grammar reflects these inflection accurately in translation. Figure 6 shows an example of transfer grammar entry for verb type Appoint. We have built around 900 rules for transfer grammar.

<table>
<thead>
<tr>
<th>Verb Type</th>
<th>English Sentence</th>
<th>Gujarati Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE APPOINT</td>
<td>A boy did the work.</td>
<td>છેકરાએ કામ કયુલો.</td>
</tr>
<tr>
<td>TYPE APPOINT</td>
<td>A Girl did the work.</td>
<td>છેકરીએ કામ કયુલો.</td>
</tr>
</tbody>
</table>

Table 2: Verb type Appoint

<table>
<thead>
<tr>
<th>Verb Type</th>
<th>English Sentence</th>
<th>Gujarati Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE ALLOW</td>
<td>A boy came.</td>
<td>છેકરાએ આષો.</td>
</tr>
<tr>
<td>TYPE ALLOW</td>
<td>A Girl came.</td>
<td>છેકરી આવી.</td>
</tr>
</tbody>
</table>

Table 3: Verb type Allow Example

Figure 6: Sample Transfer Table Entry
5.2 Morph Synthesizer

A morph synthesizer is a tool which synthesizes output word according to its grammatical features. As destination Gujarati language is inflectional, we need to identify various inflections and also build the rules regarding when to apply which kind of inflection. In this system the root word is stored in database and while translating, based on grammatical features of source word, destination word is inflected and hence synthesized output is produced. In this section we describe rules for each of above mentioned synthesizer and implementation details for the same.

For this project we have built rules and implemented those rules for following synthesizers:

5.2.1 OF Synthesizer

The task of OF synthesizer is to replace corresponding target language inflected word for “OF”. In this synthesizer, English sentence having following structure is scanned:

NN1+OF+NN2

NN1 and NN2 are Nouns. These nouns have some features associated with them. Based on feature of NN1, the rule is applied on NN2 and output word is synthesized.

For example if English sentence is: Book of Ram. So here NN1 is Book, NN2 is Ram. Feature of NN1 is GF (Gender Female), so according to table 4 we apply the rule and append ન to NN2 and hence translation turns out to be રામન  ચ પડી.

<table>
<thead>
<tr>
<th>Feature of NN1</th>
<th>Rule applied on NN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>NN2+નુ</td>
</tr>
<tr>
<td>GF</td>
<td>NN2+ન નુ</td>
</tr>
<tr>
<td>GN</td>
<td>NN2+નુ</td>
</tr>
<tr>
<td>NP</td>
<td>NN2+નલ</td>
</tr>
</tbody>
</table>

Table 4: Rules For OF Synthesizer

5.2.2 Adjective Synthesizer

The task of Adjective synthesizer is to synthesize adjective in target language based on the noun that an adjective follows. The English sentence is in following format:

ADJ+NN
Table 5 shows the rules applied on Adjective based on feature of NN. As an example if the sentence is Good boy then ADJ is good and NN is Boy. So as Boy is having feature GM(Gender Male), the corresponding rule is applied and output is सारे छिड़के.

**5.2.3 Apostrophe Synthesizer**

The purpose of Apostrophe synthesizer is to synthesize output for the sentences which includes Apostrophe s. The format of such sentences are:

<table>
<thead>
<tr>
<th>Feature of NN</th>
<th>Rule applied on ADJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>(ADJ-2)+◌૦</td>
</tr>
<tr>
<td>GF</td>
<td>(ADJ-2)+◌૦</td>
</tr>
<tr>
<td>GN</td>
<td>ADJ</td>
</tr>
<tr>
<td>NP</td>
<td>(ADJ-2)+◌૦</td>
</tr>
</tbody>
</table>

*Table 5: Rules For Adjective Synthesizer*

NN1+Apostrophe s+NN2

Table 6 shows the rules applied on NN1 based on features of NN2. For example English sentence is Ram’s Book. So NN1 is Ram and NN2 is Book. As features of NN2 is GF so Rule 2 is applied and output is given as રામના ચોટી.

<table>
<thead>
<tr>
<th>Feature of NN2</th>
<th>Rule applied on NN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>NN1+નન</td>
</tr>
<tr>
<td>GF</td>
<td>NN1+નન</td>
</tr>
<tr>
<td>GN</td>
<td>NN1+એલ</td>
</tr>
<tr>
<td>NP</td>
<td>NN1+એલ</td>
</tr>
</tbody>
</table>

*Table 6: Rules For Apostrophe Synthesizer*

**6. Observation**

After above synthesizers are implemented in the system, we have prepared 100 Gold sentences which contains different types of sentences and tested the system for Gujarati. We kept manual translation as reference and evaluated accuracy of output on the scale of 0 to 5. The average rating of output out of 5 is around 2.5. So we can conclude that accuracy of the system is approximately 50%.
7. Conclusion

Gujarati language is successfully included in EILMT system. The system is tested for TAG based approach. It shows around 50% accuracy on the gold data. We aim to further improve the performance of the system by increasing lexicon size and investigating TAG structure for Gujarati. We also aim to build statistical Machine Translation system for English to Gujarati machine translation.

8. Acknowledgements

English to Gujarati Machine Translation system is developed as a part of ‘Anuvadaksha’ project. The support of Ministry of Communication and Information Technology, Government of India is gratefully acknowledged.

References


***
Multithreaded Implementation of Earley Style Parsing Algorithm for F-LTAG
MULTITHREADED IMPLEMENTATION OF EARLEY STYLE PARSING ALGORITHM FOR FOR F-LTAG

Ramchandra P. Bhavsar*, AkshayDesale*, B. V. Pawar*,
Associate Professor  Software Developer  Professor
*rpbhavsar,addesale,bvpawar@nmu.ac.in
School of Computer Sciences,
North Maharashtra University,
Jalgaon(MS) 425001

Abstract

Lexicalized Tree Adjoining Grammar (LTAG) is a leading formalism in Generative Enterprise. Out of different approaches for parsing LTAG, Earley Style LTAG parsing as proposed by Joshi & Schabes is considered as favorite and popular approach for parsing LTAG. Practical implementations of this parsing algorithm have been used in LTAG based practical NL systems such as XTAG, MANTRA-MT platform and ANUVADAKS HTAG MT engine. In order to leverage the benefits of multithreaded processor architecture, multithreaded version of the algorithm along with optimizations and other issues incurred during actual implementation of algorithm for F-LTAG(Feature based LTAG) parsing have been described in the present article. Our article has been divided into four parts, Part-I describe the parsing process and FLTAG in simple words along with gist of Earley style parsing, while Part-II presents the insights on Recognizer routine along with proposed multithreaded implementation philosophy. Optimizations & their impact on time & space complexity have been discussed in Part-III followed by concluding remarks.

Keywords: FLTAG, Earley Style LTAG parser, multithreaded parser implementation.

Part-I

1. Introduction

Tree Adjoining Grammar (TAG) proposed by Prof. Arvind Joshi[5][6] is considered to be leading grammar formalism in Generative Enterprise[1]. The major strength of this formalism lies in its ability to perform constituency analysis due to its' two operations viz. Adjunction and Substitution. In LTAG grammar, lexical categories are represented using tree data structure. The elementary LTAG trees are divided in two classes viz. Initial (alpha) tree & Beta tree. Every tree is instantiated by lexicalizing it with lexical item (word) at special node in elementary called anchor. Beta tree get adjuncted on elementary trees (alpha or beta). Sentence analysis in LTAG involves combining the tree structures associated with lexical items(words) either by Adjunction or Substitution operations, in a special tree called sentential tree(generally tree anchored at verb). These operations are performed on Nodes of elementary trees which are marked with special constraints for adjunction and substitution. After combining all lexical items the resulting hierarchical structure results into complete tree for sentence, which is also called as 'parse tree' or 'derived tree' for the sentence. The yield of this tree i.e. leaf nodes(words) in left to right order represent the analyzed sentence. The parsing process actually is responsible for combining constituent
tree structures into parse tree(s) using some trivial procedure. The Parsing algorithm formally defines this procedure for checking membership of given sentence to given grammar and presents the proof of this membership in the form of derivation tree & derived tree. A derivation tree records the sequence of adjunction and substitution operations amongst various participating elementary trees, in the context of sentential (main) tree. The LTAG parsing process is achieved through ‘dot traversal’, which defines order of visiting each node in the elementary tree. Dot represents the progress of parsing at node level, for an elementary tree under parsing. Considering the structure of an elementary tree, a node can have four possible dot positions viz. la(left above), ra(right above), lb(left below), rb(right below). Fig.1 below depicts the dot traversal.

![Fig. 1 dot traversal.](image)

From fig. 1 it can be noted that the dot traverses through all four positions for intermediate nodes while for leaf node(except Foot node of beta tree), we visit only top dot positions i.e. la and ra.

Feature based LTAG(F-LTAG) is special version of LTAG in which feature structures are defined on each node of LTAG tree and feature unification takes place during completion (also referred as recognition in literature) of adjunction and substitution operations[4]. Two feature structures (top & bottom) are associated with each node. Feature unification process guarantees basic feature(syntactic and semantic) agreement between sentence constituent viz. subject-verb, object-verb, adjective-noun, aux verb-verb etc. This type of checking helps improves the correctness of output as it not checked merely on POS level. The feature unification process is encoded during parsing process while performing adjunction and substitution operations.Fig 2 depicts the adjunction and substitution operations in the context of feature unification.

![Fig 2 adjunction and substitution operations in the context of feature unification](image)
Feature unification process checks compatibility of two feature structures. Compatibility is measured on equality of values for similar features in two feature structures, while non-similar features are simply included (united) in resultant feature structure.

2. Gist of Earley style parsing

J. Earley invented the Earley Parsing algorithm for Context Free Grammar (CFG) formalism[2]. His algorithm defines three main operations namely Predictor, Scanner and Completer. The algorithm uses dot symbol in the right hand side (RHS) string of CFG production for marking the parsing progress. The Predictor gives fair chance to all possible CFG productions, matching selection criteria (mostly syntactic) for substitution, while Scanner operation matches terminal symbols (after dot) in current state with current symbol in input string. On matching Scanner recognizes the input symbol and moves to next symbol in input string. Completer recognizes complete CFG production upon visiting last symbol in RHS of production. This parsing process involving these three operations was then popularly known as Earley style parsing.

Joshi & Schabes proposed Earley style parsing algorithm for parsing LTAG[3]. However due to anatomy of LTAG in terms of tree structure representation of lexical categories and two operations viz. Substitution and Adjunction, in contrast to single operation (substitution) and linear representation of grammar rule (production) in CFG respectively, their algorithm defines three operations in Earley’s using 09 operations. The detail algorithm has been described in following section.

Part-II

Earley style LTAG parsing algorithm defines 09 primitive operations along with recognizer routine. The recognizer routine of the algorithm drives the parsing process by invoking these operations based on dot position in the state under processing. The algorithm stores the parsing progress using State Chart data structure which contains state sets which in turn stores states. A state tuple represents configuration (node in tree, position of dot, return address etc.) of a tree under operation at particular time instance of parsing process. More than one state may get added to state chart as all operations defined at that dot position are tried, out of these one or more (in case of structural ambiguity) may lead to success in entire
parsing while others can’t proceed and get halted. The formal description of the Recognizer routine is given in below.

**Following Fig. 3 summarizes primitive operations defined in the Recognizer routine above**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Node Dot Position</th>
<th>Node Constraint/Property</th>
<th>Operational Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitution Predictor</td>
<td></td>
<td>Substitution</td>
<td><em>Case 1:</em> Predict all the possible substitutions and add state configuration to the State Chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><em>Case 2:</em> Substitution node: If no trees are available for substitution, report failure. No beta Trees: If on intermediate/foot node simply move dot to ‘lb’ of this node and add state configuration to the State Chart</td>
</tr>
<tr>
<td></td>
<td>LeftAbove</td>
<td>Non leaf node or Foot Node</td>
<td><em>Case 1:</em> Predict all the possible adjunctions and add state configuration to the State Chart, <em>Case 2:</em> If not beta trees available for adjunction simply move dot to the ‘lb’ of this node and add this configuration to the State Chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left Predictor</td>
<td>LeftAbove</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node must be terminal symbol(word) or epsilon node(epsilon node)</td>
<td>Match the current symbol in input string symbol with the node symbol(word), if they match move dot to ‘ra’ of terminal node and this configuration to the State Chart, move the pointer to next word in input string for epsilon node simply move dot to ‘ra’ of epsilon node and add this state configuration to the State Chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scanner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left Completer</td>
<td>LeftBelow</td>
<td>FootNode of beta tree</td>
<td>Jumps the predicting trees node’s ‘lb’ position (which predicted this beta tree) and add the state denoting that configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move Down</td>
<td></td>
<td>Intermediate node</td>
<td>Move the dot to ‘la’ position of first left child and add the state for this configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitution Completer</td>
<td></td>
<td>RootNode of Initial(alpha) tree</td>
<td>Complete the substitution operation on tree which predicted this tree for substitution, move dot to ‘ra’ of initial tree node marked with substitution constraint, on successful feature unification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>State has Substitution flag ON</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no. of tokens recognized = no. of tokens available &amp;&amp; dot is on initial tree</td>
<td>Show the successful parsing output i.e. derivation tree(s) produced</td>
</tr>
<tr>
<td></td>
<td>RightAbove</td>
<td>no. of tokens recognized != no. of tokens available</td>
<td>Report failure</td>
</tr>
</tbody>
</table>
The Recognizer

Let $G$ be a TAG.

Let $a_1...a_n$ be the input string.

Program recognizer

Begin

$S_0 = \{ [a, o, left, above, o, -, -, -, -] \}
| a$ is an initial tree $\}$

For $i := 0$ to $n$ do

Begin

Process the states of $S_1$, performing one of the following seven operations on

each state

$s = [a, dot, side, pos, l, fl, fr, star, t_1^*, b_1^*]$

until no more states can be added

1. Scanner
2. Move dot down
3. Move dot up
4. Left predictor
5. Left completor
6. Right predictor
7. Right completor
8. Substitution predictor
9. Substitution completor

If $S_{i+1}$ is empty and $I < n$, return rejection.

End

If there is in $S_n$ a state

---

**Fig. 3** Earley Style LTAG parsing algorithm operations
\[ S = \{ a, o, \text{right, above, o, } -, -, -, -, - \} \]

Such that \( a \) is an initial tree

Then return acceptance.

End.

3. Insights on Recognizer:

The recognizer algorithm described above reveals that the states in the StateChart are processed in linear order & multiple states may be added to the StateChart during processing of each state in the StateChart. Addition of multiple states hint forking of multiple paths in the parsing out of these paths one or more may lead to success or all failure. The process of forking multiple parsing paths can go up to any depth resulting in parse forest. The recognizer halts whenever all such paths are tried. The linear state processing order of the StateChart defeats the parallelism achieved due to ability of applying multiple operations at the dot position. The theoretical time complexity of the algorithm has been reported as \( O(|G|^2 n^9) \), while space complexity to be \( O(|G| n^6) \) in [3].

From the above discussion it is apparent that the algorithm is very compute intensive and as its' implementation would make it impractical to use for web based applications, which are tightly constrained by the response time factor (time out period), in order to overcome this problem, we decided to adopt multithreading paradigm for our implementation. The details of our multithreading paradigm are explained in following section.

4. Multithreaded implementation

If the problem at hand is large and can be divided into comparatively easy subparts which can be performed in parallel fashion, it's a good idea to use multithreading. Multithreading support is available in modern programming languages as well as present day processors. Multithreading promotes parallel execution. In multithreaded applications, the execution starts with a main thread and the main thread can recursively spawn new threads. This execution model is lucrative option for the problems in which subtasks can be carried in parallel fashion. These parallel threads execute independently in their own sealed space however the main programs data space can be shared amongst executing threads. Our deep study of the recognizer algorithm hinted to apply multithreading during state processing operations i.e. whenever we are processing a state, we primarily examine the dot position on the node of processing tree and depending on the dot position recognizer suggests addition of one or more states to the State Chart, it is this point where we can spawn a new thread for each new state and leading to parallel execution. Each of the newly spawned thread needs resources, for further carrying the parsing process ahead. Considering these resource requirements, we have divided the total resources in two classes viz. mutable resources & immutable resources. Mutable resources are those data structures which drive the parsing process while immutable ones aid the parsing process. The specific mutable resources are intermediate derivation strings, State Chart, newly added state and Feature Vector. Immutable resources include Tree Vector and Token vector. Whenever a new thread is spawned, the mutable resources from the parent thread are cloned and passed to child thread. The immutable resources work as global resources, which are shared amongst all executing threads avoiding duplication of resources. The cloning of mutable resources may seem to be overhead at first sight but the efficiency achieved through this parallelism outweighs this duplication. Also this approach controls the size of State Chart as it splits at each spawning operation. Technological advances in handling of multithreading implementation further helps us to reduce the
thread instances through thread pooling. The following block diagram (Fig. 4) shows the schemata for multithreaded implementation.

![Fig. 4 Multithreaded Implementation using stacks](image)

**Part-III**

5. **Optimization in Implementation**

5.1 **Removal of State Chart**

The LTAG parsing process in nutshell can be described as jumping *to-and-fro* amongst the LTAG trees in Tree Vector. Although the parsing algorithm stores all states generated in the process but in the practical sense only the ones generated as consequence of tree jumps are relevant from future point of view. This observation in turn directs us in a new and innovative idea of storing only those states and that too using data structure which will give the required data i.e. *state* in single lookup instead of searching linearly into the *State Chart*. This will also help to reduce the State data structure's size from 11 tuples to 5 tuples as other state variables are used for tacking the jumping addresses. The following diagram (Fig. 5) shows the jumping operations.
Due to inherent non-overlapping nature of tree jumps, the states required for backward jump(from predicted tree to predicting tree) are stored in last in first out order hence, this behavior can be computationally captured using Stack data structure. Considering the tree jumping operations Left Predictor, LeftCompleter, Right Predictor, three stacks viz. predStack, lcStack and rpStack were introduced for storing return states, thus eliminating the need for State Chart.

### 5.2 Stack based implementation explanation:

1. 3 stacks each of size \( n \), where \( n \) is no. of tokens in the input string are allocated.
2. States are pushed on respective stack, while left prediction, left completion and right prediction operations.
3. For other operations, the current state is replaced by new one for further processing.
4. The states are popped from predStack, lcStack and rpStack when the operation in progression right predictor, left completer and right completer respectively.

---

<table>
<thead>
<tr>
<th>Operation</th>
<th>Tree Jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subst. (S.P.)</td>
<td>Predictor Predicting (Initial) Tree -&gt; Predicted (Auxiliary) Tree</td>
</tr>
<tr>
<td>Subst. (S.C.)</td>
<td>Completer Predicted (Auxiliary) Tree -&gt; Predicting (Initial) Tree</td>
</tr>
<tr>
<td>Left (L.P.)</td>
<td>Predictor Predicting (Initial) Tree -&gt; Predicted (Auxiliary) Tree</td>
</tr>
<tr>
<td>Left (L.C.)</td>
<td>Completer Predicted (Auxiliary) Tree -&gt; Predicting (Initial) Tree</td>
</tr>
<tr>
<td>Right (R.P.)</td>
<td>Predictor Predicting (Initial) Tree -&gt; Predicted (Initial) Tree</td>
</tr>
<tr>
<td>Right (R.C.)</td>
<td>Completer Predicted (Initial) Tree -&gt; Predicting (Initial) Tree</td>
</tr>
</tbody>
</table>

---

**Fig5. Tree Jump Operations**

---

Repository: 

- [GitHub](https://github.com/example-repo)
6. Benefits of new implementation

6.1 Space Complexity

c

6.2 Lookup Time

The worst case lookup time for State Chart and Hash Map is $O(n)$ in case of $n$ token sentence, whereas for stack lookup time is precisely $\Theta(1)$, which results in faster execution.

7. Summary

In this paper, we have described the F-LTAG Parsing using Earley style parsing algorithm as proposed by Joshi & Schabes along with FLTAG and gist of Earley style parsing. After deep study of this parser, we got the idea of adopting multithreading paradigm for the proposed practical implementation of this parser using JAVA for ANUDKSH’s TAG MT Engine. We have described the details of our multithreaded implementation along with its advantages. Further have presented important optimization of replacing the StateChart data structure which is key data structure, with stack and still guarantee the correctness of algorithm. Space requirements and time complexity have also been discussed and it is proved that use of stack reduces the space requirement and lookup time hence the parser built using this approach is more efficient and it is suitable to use for Internet based MT application like ANUVADKSH.

References:


***
Anuvadaksh
: A Real
Integration
ANUVADAKSH : A REAL INTEGRATION

AAI Group, C-DAC Pune, EILMT Consortia

With national level objective, DeitY has put a step forward for collaborating all research across the India in the direction of machine translation and language technology. Resulting formation of consortium with various academic research institutes using their best expertise in the domain. We, the consortia of 13 institutes under TDIL, are part of this mission for "EILMT -English to Indian languages machine translation system": pertaining to English to Hindi, Marathi, Urdu, Bangla, Oria, Tamil, Gujrathi and Bodo Language, covering the selected domains of Tourism & Health care. This is a Multi-Lingual, Multi-platform & Multi engine hybrid System targeted to serve the nation by breaking the barrier of languages through its services.

Technology Development for Indian Languages (TDIL) Program initiated by the Department of Electronics and Information Technology (DeitY), Ministry of communications & Information Technology, Government of India has the objective to make common people available with machine interface in one’s own language.

At a early glance, the task seemed little difficult to integrate the long year of experiments and researches of all academic institutes into a single system. There was a effort to have minimal changes into basic operating of the system(Fig.1Architecture Layer).

The cooperation from consortia played a great team work to collate all the research components and put things as a compiled model. This approach have 4 layers at architecture level: Communication Layer, Application Layer and Business Logic Layer.

Fig. - 1 Architecture Layer
1. COMMUNICATION LAYER

1.1 NETWORK ARCHITECTURE- Anuvadaksh acts as a peripheral on machine translation platforms. This system is used as a Plug and Play in various broad level Applications. In this thin-client/thick-server design, users of this system (clients) shall use a standard browser to access the translation services of server. Clients submit the Text/Documents to the server, the job of translation is carried out at the server and the server renders translated text/documents to the clients. The EILMT application software and database consisting of Grammar Database and Lexicon Database is residing on EILMT server.

![Network Architecture Diagram](image)

Centralized design of EILMT is ideal for Internet clients who submit their English documents to a powerful EILMT server where the job of translation is performed and translated Indian Language document is sent back to the clients. Therefore, even low-end system with Internet connectivity can also avail the facility to translate the documents from remote place. This seems as an optimal solution for sharing translation-system resources.

1.2 COMMUNICATION INTERFACES TDIL- Unified User Interface Act as a leading and frontline participator and contributor for the implementation of Unified User Interface for three consortia machine Translation Projects under Technology Development for Indian Languages (TDIL) DIT. Here, Anuvadaksh plays a vital role as an application resource. We have integrated Anuvadaksh with eight Indian languages to get the translations in this common unified interface.

![Communication Interface Diagram](image)
Communication protocols are maintained for interaction with TDIL Common GUI. TDIL-DC is portal where all consortia based engines are placed together as a consolidated Multilingual Machine Translation System. An http connection protocol are established to have an interaction between the client and the server.

1.3 HTTP PROTOCOL:

Hypertext Transfer Protocol (HTTP) is the communication protocol used to exchange information between a client system and a Web server across a TCP/IP connection. We have referred this interchange in the system. Anuvadaksh is an application-level protocol used by the client and server. We have staggered the requests instead of transmitting all at the same instant. Anuvadaksh is ported on real server by same means.

1.4 WEB SERVICES:

In order to make Anuvadaksh ubiquitous we have provided it in the form of Web Service. Web Services are designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine process able format. Due to this now Anuvadaksh can be used at all major and different platforms and also as an open protocol. Using system as a Web service presents with an opportunity to take advantage of services offered by others and the opportunity to make our application available to others as a Web service very easily. Anuvadaksh as a web service have given benefits in exposing the existing functions on to network. It has increased the interoperability and it is cognized as a Standardized Protocol at every platform.

2. APPLICATION LAYER

System is implemented with J2EE environment with JBOSS Application Server. J Boss Application Server is the open source implementation of the Java EE suite of services. It is easy-to-use server architecture and high flexibility with a customizable middleware platform. System is on struts framework. Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model view-controller (MVC) architecture. Running TAG Parser as a web application may cause huge resource & memory requirement so system is experimented with multithreaded and multi-core environment to improve response time and throughput of system. System is ported to JVM with 64 bit Operating System and J Boss server with 8 GB heap space size so the execution of complex and lengthier jobs also is simpler to run.

2.1 TRANSLATION SYSTEM ARCHITECTURE

The Workflow of English to Indian Languages Machine Translation consists of five main modules namely User Log Module, Pre-Processing Module, Parsing and Generation Module, Collation and Ranking Module and Post Processing Module. EILMT, being a web based Project, has a responsibility to work with multiple users and multiple requests simultaneously. As an application Server, J Boss 5.0.0 is chosen which is devoted to the efficient execution of procedures. Proper session handling is a big task, when the system is working in multi-user scenario, so we have used a robust Database I/O instead of memory blocking file/exe. System is working with EJB 2.0 (Enterprise Java Bean) which enables rapid and simplified development of distributed, transactional, secure and portable applications. System moved to Struts frame 1.1, it encourages consistent use of MVC throughout the application and provides utility
classes to handle many of the most common tasks in Web application development.

Fig. 4 TRANSLATION SYSTEM ARCHITECTURE
Centralized design of EILMT is ideal for Internet clients, who submit their English documents to a powerful multi-core EILMT Server, where the job of translation is performed. The translated document is sent back to the clients.

The System is designed to use three-translation engines working in parallel. The pre processed text is the input to the four translation engines and generated output by the all the engine goes to collation and ranking module for evaluation.

One of the translation engines of EILMT system is Example-Based Machine Translation (EBMT). This approach is characterized as it uses a bilingual corpus as its main knowledge base, at run-time. EBMT is integrated with number of knowledge resources, such as linguistics and statistics, symbolic and numerical techniques, for integration into one framework. Another translation engine being used is Statistical Translation Model (SMT), which is a mathematical model. SMT correspondences between the words in the source and the target language, which are learned from bilingual corpora on the basis of so-called alignment models.

The Tree Adjoining Grammar (TAG) is one of the four engines of the EILMT system. The TAG consists of a set of elementary trees, divided into initial and auxiliary trees. This works on tree-to-tree translation model and makes use of syntactic tree for both the source and target language. Parser and Generator modules recognize various grammatical entities in the English sentence, analyze them, represent them in different tree structures and synthesize equivalent Indian Language sentences on the basis of the derivational tree structure and the Transfer Grammar. It has two layers of multi-threaded embedding. The Server handles each client’s request and maintains its session. It generates the threads to call all the engines in parallel. One of the outer layer thread calls the inner layer of multi-threading, that improves the heart of the TAG translation engine (the parser) by spawning a new thread for each sentential initial tree of a sentence, thereby building several derivational trees simultaneously.

The other outer layer threads makes a connection with SMT server and another communicates with EBMT engine. Initially EBMT engine was implemented in per with Linux platforms, which have been ported to windows platform. SMT is in Java, works with Linux Operating system. There was experiment to call some engine as a web service form but due to avail more performance efficiency it is decided to keep these engines on a separate physical server having peer-to-peer connectivity. An http request made a call to these engines to perform their operations.

![EILMT Process Diagram](image)

*Figure 5 EILMT http communication protocols*
The Collation & Ranking Module collates and ranks the translated output generated by different translation engines associated with the system. While preparing a complete integrated system of EILMT, options such as – SOAP/RPC, calling a DLL/EXE, intermediate File I/O, http protocol request and web service has been taken into consideration. A Standard has been set to communicate through Database for intermediate I/O transfer. Post Processing Module is targeted to provide additional features for EILMT Translation engine like multiple output selection, Synonym Replacement and Format re-builder.

2.2 DATA HANDLING

As being a Multilingual Translation System, Database Data stored in the EILMT System is having eight clusters of data (Each per Language) and some common data tables required for common purpose.

The Database being used in EILMT System is MySQL 6.0. MySQL is one of the most popular databases on the Internet, which is a multithreaded, multi-user Relational database management system (RDBMS). Besides its undoubted advantages such as ease of use and relatively high performance, MySQL offers simple but very effective security mechanisms.

MySQL 6.0 supplies more reliability, performance, and ease-of-use enhancements, starting with the new Falcon transactional storage engine. MySQL 6 features ACID transaction compliant, Crash recovery, User defined table spaces, High-speed data caches, Advanced B-Tree indexes, Performance/diagnostic monitoring tables and Simplified configuration, among others. Falcon utilizes table spaces for user data storage with there being no practical limit to how many table spaces can be created and used to manage tables, indexes, and BLOB data. All table spaces feature auto-extending data files, automatic space reclamation, and compaction of data pages.

![Fig. 6 Internal Intermediate Communication](image)

3. BUSINESS LOGIC LAYER

3.1 CODE IMPLEMENTATION

The EILMT system is J2EE applications are comprised of components, containers, and services. Components are application-level components. Web components, such as Servlets and JSPs, provide dynamic responses to requests from a Web page. EJB components contain server-side business logic for enterprise applications. Web and EJB component containers host services that support Web and EJB
modules. The An Enterprise Java Bean (EJB) is a reusable, portable J2EE component, which consist of methods that encapsulate business logic. Enterprise Java Beans (EJB) is managed, server-side component architecture for modular construction of enterprise applications. Struts frame 1.1 encourages consistent use of MVC throughout the application and provides utility classes to handle many of the most common tasks in Web application development.

4. USER INTERFACES

The EILMT system provides User Log module to facilitate with user friendly Graphical User Interface (GUI). This GUI eases the registration process for new users to get registered with the system. It also provides login facility where the user is secured with his/her personal information. The user has to select the Language Pair [English-Hindi, English-Marathi, English-Bangla, English-Oriya, English-Tamil, English-Urdu, English-Gujarati & English-Bodo].

Only registered users can login into the system. After login, the GUI provides browse option through which user can upload the file for which he/she needs translation in .rtf, .xls, .txt and .html format. To start translation the user should click ‘Proceed’ button, which gives the translated output in the corresponding language selected by the user. This module also provides the facilities to maintain the format of the user input, which shall be used for format extraction and rebuilding in later stage. Since the communication protocols make request to EILMT Server for translation, it automatically takes care of login user and session provided by caller portal. Apart from that, a registered user can directly login to system by submitting respective user-name and password. On successful login, a translation screen appears where the facility of language and domain selection has been provided. There is a facility to enter input text in two forms. User can upload a file for translation along with entering input text in text area. On pressing the translate button, the interface makes a call to translation server and provide translated output to user sentence by sentence in his desired language. Translated outputs have various facilities for post processing like – Multiple output selection, Synonym selection, care marker and typing facility. On click over the "more" button, user may get more translations, which are the outcome of various engines. The highlighted color tokens in translated output represents synonym options, clicking on which it provides multiple meaning of corresponding tokens.

![Fig. 7 Login Interface](image1)

**Fig. 7 Login Interface**

![Fig. 8 - Upload Interface](image2)

**Fig. 8 - Upload Interface**

![Fig. 8 - Translation Interface](image3)

**Fig. 8 - Translation Interface**
User is presented with the above screen with the final output. Help link provides information of all buttons and links existing in the system. Home link takes user to home page where user can input new sentences for translation. Download XML log downloads input and output in XML format.

Download XLS log downloads input and output in XLS format. Feedback Button allows user to provide modification or control in the process of system by its results or effects. NLP Components button provides user with different levels of output generation. Logout link automatically logs out the user from system. Our Translation System is capable of providing output of intermediates module in form of NLP components in xml standard format. It can provide tool support and facilitate consortium to access our website to download the NLP components as if when needed. Different modules which comes under NLP components are: Preprocessed Output, POS Final Output, TAG Parsed Derived Output, TAG Generated Derived Output and Translated Output. Derived and derivation Output is used to form the Tree view of the source and translated output.

5. PERFORMANCE:

To speed up the online translation process we have introduced a TM, a database of derivation structure of source and target sentence, integrated at the backend of the MT system. We have implemented a fully automated TM as a part of research from a RBMT using Tree Adjoining Grammar (TAG). This TM can be reused by the same RBMT system when same set of sentences or structure is encountered. Comparing new sentence for translation to already done translations improves consistency and saves time: you can
reuse previous translations or adjust them to create new, more contextually appropriate translations. This maintains the consistency and increases the efficiency of the system.

6. **HIBERNATE ON ANUVADAKSH:**

Hibernate is an open source mapping tool. Hibernate maps the objects to the relational model where entities/classes are mapped to tables, instances are mapped to rows and attributes of instances are mapped to columns of table. We have implemented Hibernate framework in Anuvadaksh to increase maintainability and portability of the system.

Tools provided by the community helps generate or develop hibernate application very fast and easy. Main architecture is divided into two parts as – web side and engine side. Web configuration is done with struts framework where model-view architecture is enriching the application server JBOSS. This is handling the client request, session and servers response to the user. Engine side work is being handled by Enterprise Java Beans which pre-processes the source text, translates the pre-processed text and then morph synthesized, collated and ranked.

The users uses the Internet client applications to feed the source document to the EILMT system and get them machine-translated. Expected skill-set of such users is familiarity with PC usage in an office environment i.e. they have some basic computer experience, such as surfing the Internet or using a word processor.

7. **SECURITY**

Penetration testing web applications is not an easy task. So understanding possible attack scenarios when it comes to securing the web application is obligatory to get a flawless system. To find vulnerabilities in our web application we have undergone a standard for performing application-level security verifications. OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. We used this open source web application security project to implement against the top 10 security threats. Servers along with Database are secured with password authentication. Basic password authentication and role based security mechanisms are being used to protect system from unauthorized access. Other security mechanisms are handled by the parent system (i.e. TDIL Data centre).

Web applications present a complex set of security issues for architects, designers, and developers. The most secure and hack-resilient Web applications are those that have been built from the ground up with security in mind. Deploying web based applications over the Internet can be a challenging and intimidating experience. Additional care have to be taken for securing web application from unauthorized access and load unbalancing.

The stateless nature of HTTP means that tracking per-user session state becomes the responsibility of the application. As a precursor to this, the application must be able to identify the user by using some form of authentication. Given that all subsequent authorization decisions are based on the user's identity, it is essential that the authentication process is secure and that the session handling mechanism used to track authenticated users is equally well protected.

- User registration form is been secured with a service known as CAPTCHA, which is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot. It is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a person.
- Handling Session Management. To secure the communication protocol on informative basis and add efficiency some security related issues were handled such as Injection, Broken Authentication and Session Management, Cross-Site Scripting, Insecure Direct Object References, Security Misconfiguration, Sensitive Data Exposure, Missing Function Level Access Control, Cross-Site Request Forgery, Using components with Known Vulnerabilities,
Invalidated Redirects and Forwards. These aspects are important for authentication, session management and secure consequential data in application which leads our system flawless from vulnerabilities.

- Providing Auditing and Logging functionality which helps to spot the signs of intrusion, inability to prove a user's actions, and difficulties in problem diagnosis.
- Capture of session identifiers resulting in session hijacking and identity spoofing.
- Identifying Security Policies and Procedures which helps Security policy determines what your applications are allowed to do and what the users of the application are permitted to do. More importantly, they define restrictions to determine what applications and users are not allowed to do.
- Use of Connection Pooling which allows multiple users (clients) to make use of a cached set of shared and reusable connection objects providing access to a database. Opening/Closing database connections is an expensive process and hence connection pools improve the performance of execution of commands on a database for which we maintain connection objects in the pool. It facilitates reuse of the same connection object to serve a number of client requests. The application server (JBoss Server) handles the responsibilities of creating connection objects, adding them to the pool, assigning them to the incoming requests, taking the used connection objects back, returning them back to the pool, etc. When a dynamic web page of the web-based application explicitly creates a connection (using JDBC 2.0 pooling manager interfaces)

8. SCENARIOS

Once the code implementation and Installation is done, system is operated as Web application. Main integrated system with windows operating system are placed on main application server having User log Module, Input Format Extractor, Pre-processing, TAG Parser & Generator, EBMT translation engine, morph synthesizer, post processing, collation and ranking modules. Server 2 is situated with Fedora operating system running SMT engine on it. Clients make a request to main server for translation; main server makes intermediate call to respective engines for translation, collates the outputs, ranks and sent back to user.

9. CONFORMANCE WITH STANDARDS –

System is compatible with W3C Consortium. http://www.w3.org/Consortium/

10. DEPLOYMENT SCENARIO

It is a web based translation system deployed at TDIL-Data center. The JBoss Application Server (AS) is the leading open source J2EE container today. Fully compliant with the J2EE 1.4 specification, it offers a container for Servlets and JSPs; an EJB container for Entity Beans, Session Beans, and Message Driven Beans; Web Services, database connection pooling, and JavaMail support. JBoss Application Server ships with best-of-breed applications including Apache Tomcat for the web tier, Hypersonic for embedded database services, and Hibernate for object-relational mapping. A J2EE certified platform, JBoss Application Server provides a useful tool for developing and deploying Java applications, Web applications and Portals.

11. DEPENDENCIES

EILMT Phase-II shall be able to communicate with the internet communications architecture.
12. **EXPANDABILITY**

Being in plug and play architecture, system is expandable.

13. **DEBUGGING**

The system using JBOSS application server which generates the logs of processes being handled on it. The log can be stored and explored for debugging purpose.

14. **AVAILABILITY**

The system shall be available 24 X 7.

15. **PORTABILITY**

The web application is coded in J2EE, Struts and hibernate, therefore, it should be transferable between different OS and Java, System can port with any database due to hibernate implementation

16. **SALIENT FEATURE**

In order to achieve the goal, the system is facilitated with features such as –

- User Log module with
  - User friendly Graphical User Interface
  - New User Registration Module
  - The user can either upload and edit a file/document or type directly into the text area of the system, for translation

- Pre-Processing module to prepare input text into engine suitable form with the help of
  - Input Format Extractor for extracting text from uploaded files and translating for the formats.rtf, .xls, .txt and .html
  - Morphological Analyzer
  - Part of Speech Tagger
  - Named Entity Recognizer including Name, Place, Date, Act & Rules
  - Word Sense Disambiguator
  - Noun/Phrase Chunking, Clause Identification

- The System is designed to use three-translation engines working in parallel namely EBMT, SMT & AG, which would facilitate the translation for all the eight language pairs.

- The Collation & Ranking Module which is responsible for collating translated outputs of all the engines for a given language pair and rank them on the basis of translation accuracy.

- Post processing module provides additional features for EILMT Translation engine like
  - Morph Synthesizer for smoothening the translated output
  - Multiple translation options
  - Synonym selection option
  - Typing facility for Target Languages
  - Transliteration Facility
  - Retaining the original format of English text
- System shall be compatible with W3C Consortium
- Browser compatibility shall be provided for popular browsers such as Google Chrome, Mozilla Firefox, Internet Explorer, Opera and Apple Safari.

17. GLOSSARY

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAI</td>
<td>Applied Artificial Intelligence</td>
</tr>
<tr>
<td>Client</td>
<td>DIT</td>
</tr>
<tr>
<td>DIT</td>
<td>Department of Information Technology</td>
</tr>
<tr>
<td>EBMT</td>
<td>Example Based Machine Translation</td>
</tr>
<tr>
<td>EILMT</td>
<td>English to Indian Languages: Machine Translation</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext markup language</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronic Engineers</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Page</td>
</tr>
<tr>
<td>LRMT</td>
<td>Language Resource Management Tool</td>
</tr>
<tr>
<td>MO</td>
<td>Multiple Outputs</td>
</tr>
<tr>
<td>MT</td>
<td>Machine Translation</td>
</tr>
<tr>
<td>NER</td>
<td>Named Entity Recognizer</td>
</tr>
<tr>
<td>POS</td>
<td>Part of Speech</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>SMT</td>
<td>Statistical Machine Translation</td>
</tr>
<tr>
<td>SSSF</td>
<td>Shakti Standard Format</td>
</tr>
<tr>
<td>STM</td>
<td>Statistical Translation Model</td>
</tr>
<tr>
<td>TAG</td>
<td>Tree Adjoining Grammar</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>Web Site</td>
<td>A place on the World Wide Web</td>
</tr>
<tr>
<td>WSD</td>
<td>Word Sense Disambiguation</td>
</tr>
</tbody>
</table>

System is available at:
http://temp-eilmt.cdac.in/eilmt/login.jsp

***
Morphology based Factored Statistical Machine Translation (F-SMT) system from English to Tamil
MORPHOLOGY BASED FACTORED STATISTICAL MACHINE TRANSLATION (F-SMT) SYSTEM FROM ENGLISH TO TAMIL

M. Anand Kumar, S. Rajendran, Soman K.P

Amrita Vishwa Vidyapeetham,
Coimbatore

Abstract

This paper presents a novel preprocessing methodology in factorized Statistical Machine Translation system from English to Tamil language. SMT system considers the translation problem as a machine learning problem. Statistical machine translation system for morphologically rich languages is a challenging task. Moreover it is very complex for the different word order language pair. So a simple SMT alone would not give good result for English to Tamil, which differs in morphological structure and word order. A simple SMT system performs only at the lexical level mapping. Because of the highly rich morphological structure of Tamil language, a simple lexical mapping alone will suffer a lacuna in collecting all the morphological and syntactic information from the English language. The proposed SMT system is based on factored translation models. The factored SMT uses machine learning techniques to automatically learn translation patterns from factored corpora. Using the learned model FSMT predicts the output factors for the given input factors. Using the Tamil morphological generator the factored output is synthesized.

1. Introduction

Statistical approach to machine translation learns translation patterns directly from training sentences and generalized them to handle new sentences. When translating from simple morphological language to the rich morphological language, the SMT baseline system will not generate the word forms that are not present in the training corpora. For training the SMT system, both monolingual and bilingual sentence-aligned parallel corpora of significant size are essential. The corpus size decides the accuracy of machine translation. The limited availability of parallel corpora for Tamil language and high inflectional variation increases a data sparseness problem for phrase-based SMT. To reduce the data sparseness, the words are split into lemma and their inflected forms based on their part of speech. Factored translation models [Koehn and Hoang, 2007] allow the integration of the linguistic information into a phrase-based translation model. These linguistic features are treated as separate tokens during the factored training process.

\[ P(T|E) = \frac{P(T) \ P(E|T)}{P(E)} \]

\[ T^* = \text{argmax} \ P(T) \ P(E|T) \]

\[ T \]
SMT works on the above equation. Where T represents Tamil language and E represents English language. We have to find the best Tamil translation sentence \((T')\) using \(P(T)\) and \(P(E|T)\), Where \(P(T)\) is given by the Language model and \(P(E|T)\) is given by the translation model.

## 2. Factored SMT for Tamil

Tamil language is morphologically rich language with free Word order of SOV pattern. English language is morphologically simple with the word order of SVO pattern. The baseline SMT would not perform well for the languages with different word order and disparate morphological structure. For resolving this, we go for factored SMT system (F-SMT). A factored model, which is a subtype of SMT [Koehn and Hoang, 2007], will allow multiple levels of representation of the word from the most specific level to more general levels of analysis such as lemma, part-of-speech and morphological features. A preprocessing module is externally attached to the SMT system for Factored SMT.

The preprocessing module for source language includes three stages, which are reordering, factorization and compounding. In reordering stage the source language sentence is syntactically reordered according to the Tamil language syntax using reordering rules. After reordering, the English words are factored into lemma and other morphological features. A compounding process for English language is then followed, in which the various function words are removed from the reordered sentence and attached as a morphological factor to the corresponding content word. This reduces the length of English sentence. Now the representation of the source syntax is closely related to the target language syntax. This decreases the complexity in alignment, which is also a key problem in SMT from English to Tamil language.

Parallel corpora and monolingual corpora are used to train the statistical translation models. Parallel corpora contains factored English sentences (using Stanford parser) along with its factored Tamil translated sentences (using Tamil POS Tagger [V Dhanalakshmi et.al, 2009] and Morphological analyzer [M Anand Kumar et.al, 2009]). Factorized monolingual corpus is used in the Language model.

The parsed source language is reordered according to the target language structure using the syntax based reordering system. A compounding process for English language is then followed, in which the various function words are removed from the reordered sentence and attached as a morphological factor to the corresponding content word. This reduces the length of English sentence. Now the representation of the source syntax is closely related to the target language syntax. This decreases the complexity in alignment, which is also a key problem in SMT from English to Tamil language.
The factored SMT system’s output is post processed, where the Tamil Morphological generator is pipelined to generate the target sentence. Figure.1 shows the architecture of the prototype factored SMT system from English to Tamil.

3. **Morphological models for Tamil language**

Morphological models for target language Tamil are used in preprocessing as well as post processing stage. In preprocessing, Tamil POS tagger and Morphological analyzer are used to factorize the Tamil parallel corpus and monolingual corpus. Morphological generator is used in the post processing stage to generate the Tamil words from Factored SMT output.

4. **Tamil POS tagger**

Parts of speech (POS) tagging means labeling grammatical classes i.e. assigning parts of speech tags to each and every word of the given input sentence. POS tagging for Tamil is done using SVM based machine learning tool [V Dhanalakshmi et.al, 2009], which make the task simple and efficient. The SVM
Tool is used for training the tagged sentences and tagging the untagged sentences. In this method, one requires Part of speech tagged corpus to create a trained model.

5. **Tamil Morphological Analyzer**

The Tamil morphological analyzer is based on sequence labeling and training by kernel methods. It captures the non-linear relationships and various morphological features of natural language in a better and simpler way. In this machine learning approach two training models are created for morphological analyzer. These two models are represented as Model-I and Model-II. First model is trained using the sequence of input characters and their corresponding output labels. This trained model-I is used for finding the morpheme boundaries [M Anand kumar et.al,2009].

Second model is trained using sequence of morphemes and their grammatical categories. This trained Model-II is used for assigning grammatical classes to each morpheme. The SVM Tool is used for training the data. Generally SVM Tool is developed for POS tagging but here this tool is used in morphological analysis.

6. **Tamil Morphological Generator**

The developed morphological generator receives an input in the form of lemma word class Morpholexical Information, where lemma specifies the lemma of the word-form to be generated, word class specifies the grammatical category (POS category) and Morpholexical Information specifies the type of inflection. The morphological generator system needs to handle three major things; first one is the lemma part, then the word class and finally the morpholexical information. By the way the generator is implemented makes it distinct from other morphological generator [M Anand kumar et.al,2010].

The input which is in Unicode format is first Romanized and then the paradigm number is identified by end characters. For sake of easy computation we are using romanized form. A Perl program has been written for identifying paradigm number, which is referred as column index. The morpholexical information of the required word class is given by the user as input. From the morpholexicon information list the index number of the corresponding input is identified, this is referred as row index. A verb and noun suffix tables are used in this system. Using the word class specified by the user the system uses the corresponding suffix table. In this two-dimensional suffix table rows are morpholexical information index and columns are paradigm numbers.

7. **Conclusion**

In this paper, we have presented a morphology based Factored SMT for English to Tamil language. The morphology based Factored SMT improves the performance of translation system for morphologically rich language and also it drastically reduces the training corpus size. So this model is suitable for languages which have less parallel corpus. Tamil morphological models are used to create a factorized parallel corpus. Source language reordering module captures structural difference between source and target language and reorder it accordingly. Compounding module converts the source language structure to fit into the target language structure. Initial results obtained from the Factored SMT are encouraging.
References


***
Transferring of Prepositional phrases in English into Tamil: A machine learning approach
1. Introduction

Ambiguity is the core problem in computational linguistics. Ambiguity arises at almost every level of language processing, from word level processing tasks like part of speech tagging to high-level tasks like discourse planning. A machine has to be able to resolve these ambiguities and combine the information from different levels into an unambiguous meaning representation in order to understand a human language.

Preposition is one of the word classes which is both frequent and highly ambiguous. The different senses of a preposition express different relations between the preposition complement and the rest of the sentence. The sense is related to the semantic role of the dominating prepositional phrase. To understand the semantics of a prepositional phrase, a system would need access to both the higher level semantics of the semantic role and the finer word-token level semantics of the preposition.

Prepositions are not given the attention they deserve in the earlier studies on resolution of ambiguity. Even in lexicographic works including dictionaries, prepositions are not elaborately discussed explicating the ambiguity they carry along with them. Presuppositions are not deeply studied in the corpus analysis unlike the other parts of speech. Though prepositions are only a closed set of words exhibiting certain grammatical functions, their polysemous nature is comparable to other parts of speech. Similar to the major parts of speech like noun and verb, prepositions also creates problem in their interpretation. The interpretation of prepositions becomes a challenge to the computational community who are involved in natural language processing. They are closely related to verbs as the indicators of their internal arguments.

Preposition is a term used in grammatical classification of words referring to the set of items which typically precede NP (often single nouns or pronouns) to form a single constituent of structure. Prepositions normally precede nouns or pronouns. For example

The cow is grazing in the field.

The preposition ‘in’ shows the relationship between cow and field. In the above sentence object of the preposition field comes after the preposition in. Hence the noun or pronoun which is used with a
preposition is called its object. It is in the accusative case and is said to be governed by the preposition. In the above given sentence the noun field is in accusative case and is governed by the preposition in. A preposition may have two or more objects as in the following sentence.

The road runs over hill and plain.

It has to be noted here that prepositions can also be an adverb and that is they can be used without an object. If personal pronouns I, we, he, she, they etc are used as the object of a preposition, then their objective form me, us, him, her, them, have to be used.

Tamil makes use of postpositions instead of prepositions; these postpositions could be affixes or free forms or a combination of both.

avan meecai meel puttakattai vaittaan

he table on book-ACC keep-PAST-3PS

‘He kept the book on the table’

Though we can draw one to one correspondence between English prepositions and Tamil postpositions in many instances, there are drastic differences between them in some instances.

This paper aims at transferring of prepositional phrases in English into Tamil by making use of a machine learning approach. We are considering only two prepositions ‘for’ and ‘with’ to substantiate our arguments.

2. Ambiguity in prepositions

Actually prepositions themselves are ambiguous at the source language level. One can expect ‘transfer ambiguity’ at the transfer level when we move from English to Tamil. Take for example the following sentences with the preposition ‘for’.

1. He boarded the train for Jaipur.
2. I waited for you.
3. Ram has sympathy for the poor.

Look at their translations into Tamil.

1a. avan jaipUrukku rayil ERinAn
2a. wAn unakku vENti/Aka kaththiruuthEn

3b. rAmukku EzaikaL itam irakkam irukkiRathu.

In the first sentence the preposition ‘for’ is matched to dative suffix –ukku in Tamil; in the second sentence ‘for’ is matched against the benefactive vENti/Aka and in the third sentence ‘for’ is matched with the receiver postposition iTam ‘to’. This three way distinction of ‘for’ in Tamil exhibit the three-way ambiguity at the source language which is reflected at the transfer level.

Now let us look at the sentences with ‘with’ preposition.

4. He apologized with her.
   4a. avan aval-itam mannuppu kEttAn

5. Blend water with milk.
   5a. taNNIrutan pAlaik kala

6. We walk with legs.
   6a. wAm kAlkaLAl natakkinROm

In the first sentence the preposition ‘with’ is matched with the addressee marker itam in Tamil; in the second sentence ‘with’ is matched against the sociative postposition utan and in the third sentence ‘with’ is matched with the instrumental case suffix Al ‘by’. This three way distinction of ‘with’ in Tamil exhibit the three-way ambiguity at the source language which is reflected at the transfer level.

Proposition sense disambiguation requires lexical and world knowledge, and contextual information as generally with word sense disambiguation. For example, consider the sentences ―Ram built a house with a magic wand‖ and ―Ram built a house with a bed room‖. In the first example, ‘with’ is used to indicate the instrument that is used in building, while in the second example ‘with’ is used to indicate the part of a house that has a bed room built-in. To distinguish these two senses of ‘with’, we need to understand the difference between ‘magic wand’ and ‘bed room’. This requires world knowledge or the understanding of word meanings.

3. Earlier works

A number of researchers including Alam (2004), Harabagiu (1996), O’Hara and Wiebe (2002, 2003), Sopena, LLoberas, and Moliner (1998), Sablayrolles (1995), Saint-Dizier and Vazquez (2001), Litkowski (2002), and Boonthum, Toida and Levinstein have studied recently disambiguation of the preposition. Alam (2004) studied the disambiguation of the preposition ‘over’ by taking into account two things: one is the meaning expressed by the complement noun phrases of the prepositions and the other is the head of the prepositional phrases (verb or noun phrases). Harabagiu (1996) made use of WordNet to
disambiguate prepositional phrase attachment. A special issue of *Computational linguistics* (Baldwin et al., 2009) was devoted to discuss about the issues on preposition. Preposition sense disambiguation was one of the SemEval 2007 tasks (Litkowski and Hargraves, 2007), and was subsequently explored in a number of papers using supervised approaches. O’Hara and Wiebe (2009) presented a supervised preposition sense disambiguation approach which explores different settings. Tratz and Hovy (2009), and Hovy et al. (2010) make explicit use of the arguments for preposition sense disambiguation, using various features. Rudzicz and Mokhov (2003) and O’Hara and Wiebe (2003) have studied the constraints of prepositional constructions to annotate the semantic role of complete prepositional phrases with FrameNet and Penn Treebank categories.

The present study is much rarer of the kind of studies mentioned above as it aims to resolve the prepositional ambiguity at the transfer level.

**4. Methodology**

English sentences with the prepositions “for” and “with” are collected and manually tagged with equivalent postpositions in Tamil. We divided the annotated English preposition corpus into train and test sets. The training dataset consists of 430 sentences and testing set has 100 sentences Sample English sentences with preposition tag are given in Table.1. The preposition tag contains the Tamil postposition information. By using this method, the prepositions of English are disambiguated and translated into Tamil postpositions.

To disambiguate a preposition $p$, our system uses the bag-of-words and linguistic information as features. The essential linguistic information for each English sentence is extracted using the Stanford parser toolkit. This linguistic information such as lemma, POS tag, and dependency tag is used as features in machine learning based disambiguation model. For training the model we utilized SVM-Light, a public distribution of SVM (Support Vector Machines) by (Joachims, 1999). The preposition disambiguation framework is shown in the Figure.1.
Table: 1 Sample English Sentences and its Prepositional Tag

<table>
<thead>
<tr>
<th>S.No</th>
<th>English Sentences</th>
<th>Preposition Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dussehra is celebrated <em>for</em> ten days.</td>
<td>NUL</td>
</tr>
<tr>
<td>2</td>
<td>I have played outside <em>for</em> an hour.</td>
<td>DAT-Aka</td>
</tr>
<tr>
<td>3</td>
<td>She does not bring water <em>for</em> me.</td>
<td>DAT</td>
</tr>
<tr>
<td>4</td>
<td>I had headache <em>for</em> two or three days.</td>
<td>Aka</td>
</tr>
<tr>
<td>5</td>
<td>His body is covered <em>with</em> hair.</td>
<td>Al</td>
</tr>
<tr>
<td>6</td>
<td>Peter has fallen out <em>with</em> his boss.</td>
<td>il-iruwthu</td>
</tr>
<tr>
<td>7</td>
<td>I have lived <em>with</em> my parents for over 10 years.</td>
<td>Aka</td>
</tr>
<tr>
<td>8</td>
<td>She did not come <em>with</em> me.</td>
<td>utan</td>
</tr>
</tbody>
</table>

Figure: 1 Framework for English Preposition disambiguation
Our investigation revealed that machine learning method for solving the preposition disambiguation problem using existing knowledge sources as features improved the performance of preposition translation. In future, the feature set can be further tuned with the use of English Word Net and the Semantic role’s of verb in the sentences.

5. Conclusions

It can be inferred by linguistic analysis that within the context for prepositions, the governor (head of the NP or VP governing the preposition), the object of the preposition (i.e., head of the NP to the right), and the word directly to the left of the preposition have the highest influence. While disambiguating prepositions the maximal accuracy can be achieved by considering the context, features, and granularity. Though the preliminary result is encouraging, various issues still need to be addressed, i.e. improving the result by disambiguating noun senses or using world knowledge or context information.

Preposition sense disambiguation has many potential uses. For example, due to the relational nature of prepositions, disambiguating their senses can help with all-word sense disambiguation. In machine translation, different senses of the same English preposition often correspond to different translations in the foreign language. Thus, disambiguating prepositions correctly may help improve translation quality.

6. References


***
Text
To speech
(TTS) Indian Languages: A survey
TEXT TO SPEECH (TTS) IN INDIAN LANGUAGES: A SURVEY

Ranjeet Singh, Santosh Kumar Upadhyay
Ranjeet.sobi@gmail.com, ersk2006@gmail.com
M.Tech Scholar (CSE) Mewar University Chittogarh (Rajasthan), Assistant Prof. Galgotia College of Engineering & Technology Greater Noida

Abstract—Text to speech in Indian language survey paper is, a research overview of work done by the researchers, scholars and Engineers in the field of the processing and building of Speech synthesis. This paper gives an idea of the different approaches, methodologies and techniques taken by them to convert Indian language text data into speech with their respective tools. The quality and methodology of TTS systems can be effectively assessed and test on the basis of reliable, verified and valid listening tests. These works done by various researchers have contributed highly in the technical world and helped building the knowledge of the common people; mainly for visually impaired people. In this paper a detailed overview survey of challenge and technology used by them for Indian language Text to Speech systems is given.

1. INTRODUCTION

A Text to Speech is a computer based speech Synthesis System, abbreviated as TTS, is a form of speech synthesis that converts digital input text into spoken voice output.

The Text to Speech System can be used for various applications. TTS Software embedded with Screen Reader help the visually challenged people to read the text on the computer screen and they would be able to perform the computer operations. Another crucial use is in reading any online service / application for the blind, where a system would process text from an online source and convert it into a speech format. Nowadays, quite sophisticated and high end systems exist that facilitate human–computer interaction for the blind, in which the TTS can help the user, navigate around a desktop system. Apart from this, TTS have been used for weather forecasting, navigation, education and other wide variety of applications.[1,2]

2. Players in TTS field

There are few organizations working in the area of Indian language TTS. Some of the major players in this field are:

- Indian Languages (Hindi, Bengali, Marathi, Tamil, Telugu and Malayalam) Screen Reader application Developed by the following organizations in consortium mode: IIT Madras, IIT Hyderabad, IIT Kharagpur, CDAC Mumbai, CDAC Thrivananthapuram
- JNU University Language (Sanskrit)
- Media Lab Asia Shruti-Drishti (Text-to-Speech and Text-to-Braille)
- HP Lab –Hindi TTS based on Festival framework
- Hyderabad Central University (HCU) -Vaani -Telugu
- IIT Kharagpur - An Indian language TTS synthesizer (named SHRUTI) that accepts text inputs in two Indian languages namely Hindi and Bengali
- Simputer Trust – Dhvani TTS [Dhvani, 2001], [Hariharan, 2002]
- C-DAC Bangalore - Matrubhasha API [Raman, 2004]
- IISC Bangalore-Thirukkral & Vaachaka,
- The TTS and screen reader technology may also be deployed in CSCs and E-Governance.
There are some other institutions also, where the research on speech synthesis systems is going on, it includes – HCU, IISc Bangalore, Utkal University, TIFR Mumbai, C-DAC Noida and College of Engineering, Guindy etc. [3,4,5]

3. Devanagari based Text to Speech Softwares:

The following three Text to Speech software are available on http://tdil-dc.in/,

3.1 Text to Speech screen reader application, for both Windows (NVDA) and Linux (OCRA) based operating systems available in six IL like Hindi, Marathi, Bengali, Tamil, Telugu and Malayalam. Text to speech as a screen reader application which works with different editors like MS -Word, Notepad, Word Pad but it does not support PDF. Currently this software is monolingual. TTS system is independent of any font. TTS system simply reads out what is written in the text editor and it does not support proof reading. [5]

3.2 “Text to Speech Mozilla Browser Plugin”: Text to speech software is available freely in the form of a Mozilla plugin. TTS Mozilla plug-in is available in 8 different Indian languages (Hindi, Marathi, Bengali, Gujarati, Malayalam, Tamil, Telugu, Kannada) and also includes English in 3 different accents (Hindi, Tamil, Telugu). Speech change features also provided, with this listener can adjust the reading speed as per his listening. 5 options of voice speed: normal, slow, slower, fast, faster [6].

3.3 Text to Speech Chrome Browser Plug-in: Text to Speech Chrome Plug-in is similar to Text to Speech Mozilla Plug-in. After successful installation of chrome plug-in an icon will appear on top right corner of the browser. Click on the icon after selecting the text and speech file (mp3) will get played. [7]

3.4 Text to Speech android app: SMS Reader on android platform for 5 Indian Languages namely Hindi, Marathi, Tamil, Telugu and Gujarati have been developed and made available through E-gov mobile seva gateway (http://mgov.gov.in). [9]

3.5 JNU Sanskrit Tool [8]: The "Samvacaka - A Speech Synthesis System for Sanskrit Prose" is a result of the research carried out by Diwakar Mishra (Ph.D. 2009-2013) under the supervision of Dr. Girish Nath Jha and Dr. Kalika Bali. The application takes simple Sanskrit text and returns synthesized speech or Sanskrit voice output. It does not take into consideration prosodic features and can also be used for Hindi with reasonable success.

3.6 Issues

a. Currently TTS is in development stage, if compared to human voice the speech output of TTS is quite robotic. There is a need for improvement in the naturalness of the TTS.

b. TTS failed to pronounce the percentage sign, Decimal, Hindi numerals, Roman numbers, double quotes.

c. Abbreviations was not handled, e.g. 10gm,1kg etc.

d. Mathematical equation are not handled.

e. Currency symbol like ₹, $ are being skipped by the systems.

4. Challenges in TTS

4.1 Text Preprocessing And Text normalization Challenges: Text preprocessing is a difficult task for Devanagari Text to Speech. And it is more complex in other IL text to speech.
Following Challenges we face in Text preprocessing for Indian language

a. Digits and numerals: एक इंच मे 254. cm होते है . Here [इंच] is read as [Inch], If [in] is there it should also be read as [Inch] and in second verse cm is read as ['सेंटीमीटर']

b. Fraction/Dates are problematic. Examples are:
   - 23/24 can be read as [तेइस बटे चौबीस ]

c. Roman Numerals can be a bit problematic. Examples are:
   - एलिजाबेथ IV can be read as [एलिजाबेथ चतुथथ ]
   - And IV can be read as चतुथथ अध्याय ]Prefix and postfix will make it more complicated

d. Abbreviations
   - 1st /2nd/3rd as प्र. / दि / तृ

4.2 Phonetic Analysis Challenge: Phonetic Analysis converts the orthographical symbols into phonological using Phoneme Conversion. Two types of approaches are used in the pronunciation of the word in speech synthesis

A. Dictionary based: A large dictionary is used to store all kinds of words and its pronunciation. The System looks for the word and its respective pronunciation in the dictionary one by one. This type of approach is very fast and the result came, is of better quality. The system has drawback also, like if the word is not found in the dictionary, the system throws an error and the system will stop working.

B. Rule based: In Rule based approach, there is some rule for the letter sounds for a word, letter sounds are blended together to form a pronunciation based on some rule. The System has advantages over previous approach that it requires no database and it can work on any type of input. The system has complexity for the irregular inputs In Indian Language. Aksharas are used to form a word and hence speech data. The Aksharas has following properties
   1. An Akshara is an orthographic representation of speech sound in Indian language
   2. Akshara are syllabic in nature
   3. The Typical forms are Aksharas are V, CV, CCV, CCCV or we have the general form as C*V

4.3 Prosodic Modeling and Intonation: Prosodic Modeling and Intonation describes the Pitch, Stress Pattern, rhythm and intonation in the output speech. Prosodic modeling describes the human emotions. If the system identifies emotions in the vocal and it is very natural synthesized speech. Capturing emotions, stress, rhythm is no doubt a challenging task of voice syntheses for Indian Languages. And it’s vital for smart and naturalness of the speech.

Following are the Challenges in the Prosodic Modeling and Intonation:

In voice recording of voice samples. Suppose the person who is recording is not smiling or stressed as required by the sentence so sample voice may not record the smiling emotion or stress emotion.

Example:

a. भारत मा ता की जय !! – This recording is recorded in full emotion. If the same is recorded in Smiling emotions, then the effect is completely different

b. आप कमरे से बहार जाए – In this the sample is recorded with angry emotion, if the system doesn’t catch and it will not up to the output as desired. The modeling of the intonation is to take
care of the modification of the One of the related issues is modification of the pitch contour of the sentence, depending upon whether it is an affirmative, interrogative or exclamatory sentence. [2]

5. Some basic TTS frameworks
This section gives details of the generic frameworks available for the development of a TTS synthesizer. Some of these are as back-end engines and others are full-featured commercial TTS frameworks. [3]

a. Festival TTS Framework The Festival TTS synthesizer was developed in CSTR at the University of Edinburgh by Alan Black and Paul Taylor and in co-operation with CHATR, Japan [Black et al., 2001]. Festival is multi-lingual (currently English, Spanish and Welsh) and is freely available complete diphone concatenation and unit selection TTS synthesizer. It is the freeware synthesis system and it includes a comprehensive manual. Festival offers a general framework for developing speech synthesis systems as well as including examples of different modules. As fully, it offers full TTS synthesizer through a number of APIs. The English version of current Festival version is more advanced and the developments for this version are very fast. The synthesizer is written in C++ and uses the Edinburgh Speech Tools for low-level architecture and has a Scheme (SIOD) - based command interpreter for control. The latest details and a full software distribution of the framework are available through the Festival Website [Black et al., 2001]. [3,10,11,12,17]

b. MBROLA SYNTHESIZER MBROLA is a better-quality, diphone-based speech synthesizer and is available in the public domain. It is provided by the TCTS Lab of the Faculty Polytechnique de Mons (Belgium) whose main objective it to obtain a set of speech synthesizers for as many languages as possible. The MBROLA speech synthesizer is free of charge for non-commercial, non-military applications. MBROLA database is prepared using any of the recordings in user’s speech. Presently there are diphone databases existing for several languages: American English, Brazilian Portuguese, Breton, British English, Dutch, French, German, Greek, Romanian, Spanish and Swedish. TCTS also provides speech database labeling software: MBROLIGN, a fast MBROLA-based TTS aligner. MBROLIGN can also be used to produce input files for the MBROLA v2.05 speech synthesizer. Demo, comparison of different voices and languages between MBROLA and other synthesis methods can be found on the MBROLA project home page [MBROLA, 1998]. [3]

c. Flite (Festival-lite) is a smaller, faster alternative version of Festival designed for embedded systems and high volume servers.

More information is available at: http://www.speech.cs.cmu.edu/flite/

6. Tools available for development of a TTS synthesizer
Different API’s are available for developing a TTS synthesizer [3] provided by different vendors, and different markup languages. There exist many different APIs for speech output but the Microsoft API for synthesizers running on Windows is getting popularity. Another API that is not so frequently used is the Sun-Java Speech API. These two are described below.

a. The Java Speech API: is being developed to allow Java applications and applets to incorporate speech technology. The API defines a cross-platform API to support command and control recognizers, dictation systems and speech synthesizers. Java Speech Grammar Format provides a cross-platform control of speech recognizers. Java Speech Markup Language provides a cross-platform control of speech synthesizers. Text is provided to a speech synthesizer as a Java
String object. The Java Platform uses the Unicode character set for all strings. Unicode provides excellent multi-lingual support and also includes the full International Phonetic Alphabet (IPA), which can be used to accurately define the pronunciations of each syllable. More information can be found on the Java homepage. http://java.sun.com/products/java-media/speech/.

b. Sapi Microsoft's speech API: The major software and speech building Systems vendors are beginning to support Microsoft's Speech API, or SAPI, which is based on the COM specification and is being adopted as the industry standard. The motive of SAPI is to eventually allow interoperability between the speech engines. The Microsoft Speech API provides applications with the ability to incorporate speech recognition (command & control dictation) or TTS, using either C/C++ or Visual Basic. SAPI follows the OLE Component Object Model (COM) architecture. It is supported by many major speech technology vendors. The major interfaces are:

Voice Commands: high-level speech recognition API for command and control. Voice Text: simple high-level TTS API. The Voice Text object is available in two forms: a standard COM interface IVoiceText and companion interfaces, and also an ActiveX COM object, VtxtAuto.dll Multimedia Audio Objects: audio I/O for microphones, headphones, speakers, telephone lines, files, etc. With the Microsoft Speech SDK, and in particular, the TTS VtxtAuto ActiveX COM object, any developer can create a TTS-enabled application using a few simple commands, such as register and speak. More information can be found on the website.

c. Markup Language: The Speech Synthesis Markup Language (SSML) specification is the W3C markup language specification that defines XML tags to be used in the speech synthesis system to be used in the different speech parameters. SSML Tag also used to define the information like language, metadata for improving the synthetic speech quality in the voice enabled applications. SSML is a markup language of W3C used to create voice enabled applications with email programs and internet browsers.[3]

6.1 Uses of SSML
i. SSML is also used to develop standalone applications and allow users to use voice commands with various online tasks such as searching the Internet, receiving and responding to emails.
ii. SSML is also used in with Spoken Text Markup Language (STML) and Java Speech Markup Language (JSML).

Example of SSML:

```xml
<?xml version="1.0"?>
<speak version="1.0"
xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/TR/speech-synthesis/synthesis.xsd" xml:lang="hi-IN">
<lexicon uri="http://www.somelexiconfile.com/lexicon.file"/>
<voice gender="female">
<p> <s>I speak <emphasis>Hindi</emphasis></s> <s>I also speak <emphasis>Marathi</emphasis></s> </p>
<sub alias="International Phonetic Association">IPA</sub>
</voice>
<audio src="royal.wav">आप का नाम कौन सा है</audio>
</speak>
```
There are many tags used in the current version 1.1 (2010) of SSML defined under W3C. The two tag extensions to the existing SSML specification in the context of Indian languages are:

**Transliterate tag**

Text input to be used in the most of the Text to Speech systems are either an English transliteration of the Indian language script or is in Unicode. As there is no standard or uniform transliteration scheme to represent different Indian language. Although there is a popular scheme use is IRANS package. To use this, it is very important to specify whether the input text is Unicode or is transliterated using some transliteration scheme.

Transliterate tag has two attributes

a. Codepage
b. Uri

Example of this tag is

```xml
<? xml version="1.0"?>
<speak version="1.0" xml:lang="hi-IN">
  <transliterate codepage="1137">मेरा नाम रणजीत है।</transliterate>
</speak>
```

```xml
<? xml version="1.0"?>
<speak version="1.0" xml:lang="en-US">
  <transliterate codepage="1252" uri="http://www.example.com/trans.file">mera nam ranjeet hai</transliterate>
</speak>
```

**Foreign tag**

The current version of SSML Specification has two attributes (<lexicon>, <phoneme>) that can be used to find the pronunciation of words or phrases.

a. The lexicon tag is used to reference external pronunciation dictionaries that are applicable to entire the document
b. In phoneme tag the pronunciation for the word or phrase is specified explicitly.

Therefore, there is a concern to define a tag that can be used to indicate that a certain word or phrase needs to be pronounced using a different pronunciation scheme without having to specify its exact phone sequence.

In this case, the tag would point to a lexicon which is different from the globally specified lexicon for the whole document. Such a tag would be helpful when dealing with foreign language words/phrases embedded in a given language text or even in the case of loan words.

Therefore, there is a tag called <foreign>_tag that has two attributes "lang" and "uri".

Example:

```xml
<? xml version="1.0"?>
```
I greeted her with a "Namaste" and showed her where she could get a ticket for the movie "Jaane bhi do yaaron".

6.2 Pronunciation Lexicon Specification (PLS)

PLS is created by Voice Browser Working Group of W3C and is a standard of W3C. Current version of PLS 1.0 (2008). The aim of PLS is to design an interoperable specifications of pronunciation information which then can be used for speech technology development. PLS provides the facility of mapping between the words or short phrases, their written representations and pronunciation to be use by speech engines and other applications uses speech engines. XML format is used by PLS and for specific language using the baseline PLS specification of the W3C. This specification provides the possibility of providing multiple pronunciations for the same orthography as well as multiple orthographies against an entry of single pronunciation in the PLS. This will almost cover all homophones and homographs. There is a possibility of incorporating acronyms and abbreviations also by providing them as an alias. PLS specification provides a framework and guideline which can be tailored to the needs of a specific language and consequently the XML tag set can be defined to build the PLS data using IPA as UTF 8 representation. PLS can be used by Text to Speech (TTS) and Automatic Speech Recognition (ASR) Engines and can have a wide variety of applications like voice browsers, pedagogical tools etc.

Multiple Pronunciations for the same Orthography in Hindi: For ASR systems it is common to rely on multiple pronunciations of the same word or phrase in order to cope with variations of pronunciation within a language. In the Pronunciation Lexicon language, multiple pronunciations are represented by more than one <phoneme> (or <alias>) element within the same <lexeme> element. In the following example the word "pran [प्राण]" has two possible pronunciations.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
alphabet="ipa" xml:lang="en-GB">
<lexeme>
<grapheme> pran </grapheme>
<phoneme> praan </phoneme>
</lexeme>
</lexicon>
```

Homograph

Some languages have words with different meanings but the same spelling (and sometimes different pronunciations), called homographs.
For example, in Hindi the word हार (पराजित होना) and the word हार (गहना) have identical spellings but different meanings. It is recommended that these words be represented using separate <lexeme> elements that are distinguished by different values of the role attribute, if a pronunciation lexicon author does not want to distinguish between the two words they could simply be represented as alternative pronunciations within the same <lexeme> element. In the latter case the TTS processor will not be able to distinguish when to apply the first or the second transcription. In this example the pronunciations of the homograph “हार ” are shown.

<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
alphabet="ipa" xml:lang="hi-IN">
<lexeme>
<grapheme> हार </grapheme>
<phoneme> हार </phoneme>
<phoneme> हार </phoneme>
</lexeme>
</lexicon>

Pronunciation by Orthography (Acronyms, Abbreviations, etc.)

For some words and phrases, pronunciation can be expressed quickly and conveniently as a sequence of other orthographies. The developer is not required to have linguistic knowledge, but instead makes use of the pronunciations that are already expected to be available. To express pronunciations using other orthographies the <alias> element may be used. This feature may be very useful to deal with an acronym expansion.

<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
alphabet="ipa" xml:lang="hi-IN">
<!-- Acronym expansion -->
<lexeme>
<grapheme> रू </grapheme>
<alias> रुपये </alias>
</lexeme>
<!-- number representation -->
<lexeme>
<grapheme>500</grapheme>
</lexicon>
Conclusion

There are only a few players in the field of IL TTS and most of the product are in beta (under development) version. In this survey we found that there are many issues that need to be addressed to improve the quality and usability of Indian Languages TTS from user point of view. At present TTS voice is quite robotic in comparison to the human voice. There is a need for improvement in the accuracy & naturalness of TTS.

Acknowledgment

I would like to thank all my seniors and colleagues for their direct or indirect help and inspiration, without which this work cannot be completed.

References

[8] “Sanskrit Text to Speech (TTS) System- संस्कृत संवाचक ” [http://sanskrit.jnu.ac.in/samvacaka/index.jsp]


