MaTra: A Practical Approach to Fully-Automatic Indicative English-Hindi Machine Translation

Ananthakrishnan R, Kavitha M, Jayprasad J Hegde,
Chandra Shekhar, Ritesh Shah, Sawani Bade, Sasikumar M
Centre for Development of Advanced Computing (formerly NCST),
Juhu, Mumbai-400049
India
{anand,kavitham,jjhegde,shekhar,ritesh,sawani,sasi}@cdacmumbai.in

Abstract

MaTra is a fully automatic system for indicative English-Hindi Machine Translation (MT) of general-purpose texts. This paper discusses the strengths of the MaTra approach, especially focusing on the robust strategy for parsing and the intuitive intermediate representation used by the system. This approach allows convenient enhancement of the linguistic capabilities of the translation system, while making it possible for us to produce acceptable translations even as the system evolves. This paper also presents encouraging results of automatic evaluation using BLEU/NIST and subjective evaluation by human judges.

1 Introduction and Background

MaTra is a fully automatic system for indicative English-Hindi Machine Translation (MT) of general-purpose texts. This paper discusses the strengths of the MaTra approach, especially focusing on the robust strategy for parsing and the intuitive intermediate representation used by the system.

It is well accepted that Fully-Automatic, High Quality, General-Purpose MT is not achievable in the foreseeable future – more so for widely divergent languages like English and Hindi. Existing MT systems work by relaxing one or more of these three dimensions. They do this by: (i) focusing on a fairly small subset of the language (in domains such as weather forecasting and official gazettes), or (ii) using human assistance during or post translation, or (iii) producing indicative rather than perfect translations.

MaTra is designed to be fully automatic, and is geared towards translating real-world, general-purpose texts. As a tradeoff, MaTra makes the last of the aforementioned approximations.

The primary testing ground for MaTra is the Web, where many sentences are grammatically incorrect or incomplete (fragments) containing unknown words and abbreviations. In such a noisy environment, with incomplete knowledge, it is impractical to work with perfect models and grammars. In this scenario, the design of MaTra represents a pragmatic approach to engineering a usable system, which aims to produce understandable output for wide coverage, rather than perfect output for a limited range of sentences.

MaTra achieves this by using a judicious mix of (i) corpus-based or statistical tools and techniques for shallow parsing, word-sense disambiguation, abbreviation handling, and transliteration, and (ii) rule-based techniques for lexical and structural transfer.

The structural transfer component has at its core a relatively simple and intuitive intermediate representation (which we call MSIR – MaTra Structured Intermediate Representation) that can accommodate most types of sentences that are found in real-world texts. The simplicity and generality of the MSIR is an important aspect of our approach. The level of detail, both syntactic and semantic, is just enough to capture the divergence between English and Hindi.

In keeping with our engineering outlook, MaTra does not attempt either a deep parse or an elaborate semantic analysis of the English sentence. The parsing strategy is hybrid – a combination of statistical and rule-based techniques. The parsing component is modular by design, with each stage working on a logically separate aspect of structural analysis. One of the primary goals of the design is graceful degradation from full sentence structures all the way down to word-by-word structures.

The design of the MSIR and the parsing algorithm is the focus of this paper.

This paper is organized as follows: the next section presents the overall architecture of MaTra and describes the various components briefly. Section 3 discusses the intermediate representation used by MaTra (MSIR). The parsing algorithm used to obtain the MSIR is presented in section 4. Section 5 provides evaluation details. Section 6 concludes the paper.

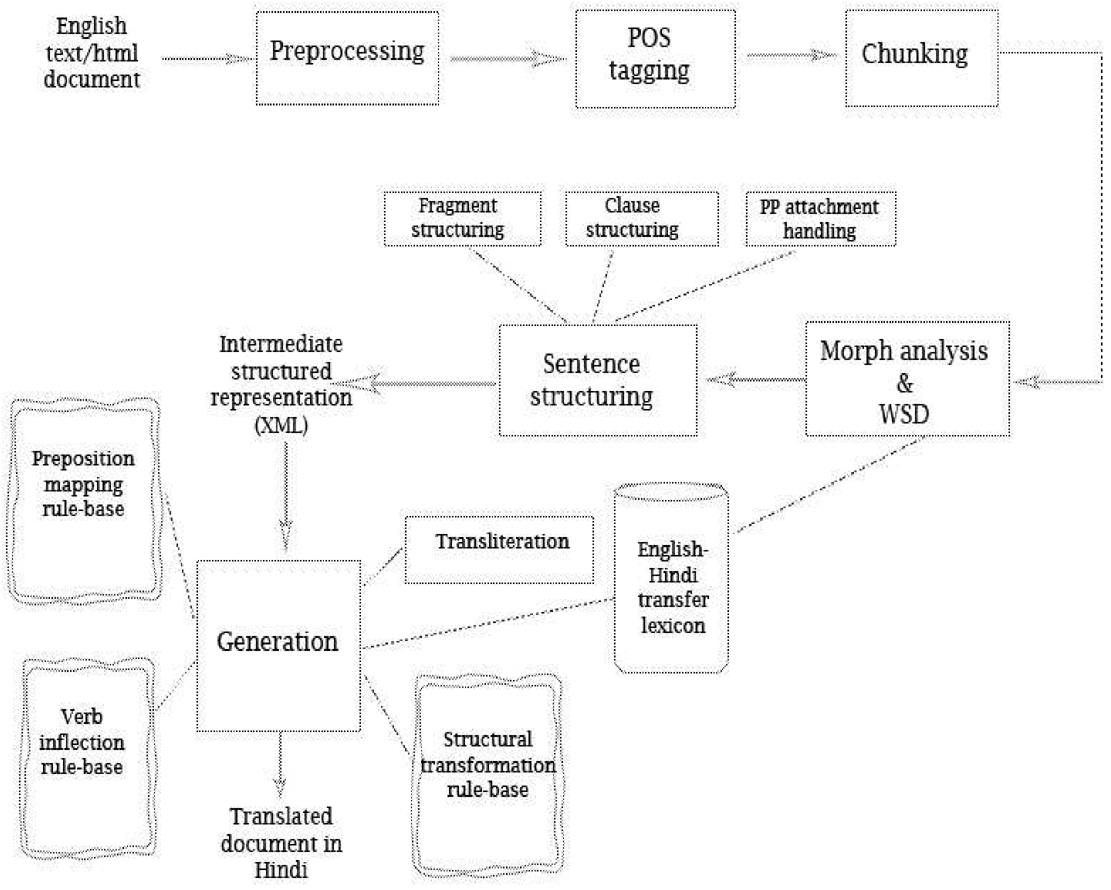
English text/html document
Preprocessing
POS tagging
Chunking
Fragment structuring
Clause structuring
PP attachment handling
Intermediate structured representation (XML)
Sentence structuring
Morph analysis & WSD
Preposition mapping rule-base
Transliteration
English-Hindi transfer lexicon
Generation
Verb inflection rule-base
Structural transformation rule-base
Translated document in Hindi

Fig 1: MaTra Architecture

```
Sentence      → IndepClause | Conjunct | Fragment | MixedBag
Fragment      → FragClause | Conjunct
Conjunct      → Word, Sentence+
IndepClause   → VerbFrame, Modifier*
VerbFrame     → VerbGroup, Subject*,Object*,Modifier*
VerbGroup     → Word+
FragClause    → FragFrame, Modifier*
FragFrame     → VerbGroup*, Subject*, Object*, Modifier*
Subject       → NounPhrase | Sentence, Modifier*
Object        → NounPhrase | Sentence, Modifier*
Modifier      → PrepPhrase | DepClause
DepClause     → Word*, FragFrame, Modifier*
NounPhrase    → Word+
PrepPhrase    → Preposition, NounPhrase
MixedBag      → (Word | NounPhrase | PrepPhrase | DepClause | Fragment | IndepClause)+
```

Fig 2: Grammar for the MaTra Structured Intermediate Representation (MSIR)

2 MaTra Architecture

Essentially, MaTra follows a structural and lexical transfer approach, using semantic information only when required. Figure 1 shows the overall architecture of the system.

The ***preprocessing*** component splits the input text into sentences. Abbreviations, acronyms, dates, numeric expressions, etc. are identified at this stage. ***Part-of-speech tagging*** and ***chunking*** are done using the fast Transformation Based Learning tool, fnTBL [Ngai and Florian, 2001]. The ***word sense disambiguation*** component, then, chooses the appropriate Hindi mapping for English words and phrases. Next, the ***sentence-structuring*** component converts the chunked input into the ***MSIR*** that facilitates Hindi generation. A rule-based ***generation engine*** [Rao et al., 1998; Rao et al., 2000; Mehta and Rao, 2003] generates the Hindi translation from the MSIR using rule-bases for preposition-mapping, noun-inflection, verb-inflection, and structural transformation. The ***transliteration*** component handles unknown words. This component uses Genetic Algorithms to learn transliteration rules from a pronunciation dictionary. [Mishra, 2004].

3 MaTra Intermediate Structured Representation (MSIR)

The core part of our transfer-based framework for translation deals with the transfer of a single clause, which is adequate for handling mono-finite sentences [Rao et al., 2000]. This has then been systematically extended to handle multi-finite sentences and non-finite clauses, thus covering compound and compound-complex sentences [Mehta and Rao, 2003], and further, to fragments and incomplete sentences. However, the framework, as of now, does not handle imperative and interrogative sentences. Figure 2 shows the grammar for the MSIR based on this framework.

In this section, we first discuss how various types of clauses are represented in the MSIR. Then, we look at the MSIR interpretations of *Subject*s, *Object*s, and *Modifier*s, which are simplifications of the traditional definitions. Finally, we look at how incomplete sentences and fragments are represented in the MSIR.

3.1 Clauses

A clause is the basic unit of predication in any language, and forms the basis of the MSIR too. A clause consists of a single verb group, which represents an action, event or state change. The verb group may consist of one or more verbs, including auxiliaries and pre-modifying adverbs, and may be finite or non-finite. Every verb has certain sub-categorization features, which define the number and nature of other constituents that attach with the verb to form the clause, namely, the subject, objects, complements, and modifiers.
Each clause modifier may itself be a separate clause. This recursion can be used to create sentences with a hierarchy of clauses.

Clauses may be classified, based on the types of verbs that they contain, into the following three categories:

- Finite Clauses: clauses containing a finite verb phrase. E.g., *Jayshankar` has visited Delhi*
- Non-finite clauses: clauses containing a non-finite verb phrase but no finite verb

phrase. E.g., *Having visited Delhi, Jayshankar …*

- Verbless clauses: clauses with no verb phrase. E.g., *Jayshankar, then at Delhi, …*

Clauses are represented in the MSIR as either *independent* or *dependent* clauses. In our representation, independent clauses are always finite, whereas dependent clauses may be finite, non-finite or verbless.

Sentences are classified as Simple, Complex and Compound based on the types of clauses that they contain. Simple sentences contain a single independent clause; complex sentences contain one independent and at least one dependent clause connected by subordinators; and a compound sentence contains more than one independent clause connected by coordinating conjunctions.

In the following three subsections, we look at how the MSIR represents simple, complex and compound sentences respectively.

3.1.1 Representing an Independent Clause – Simple Sentences

The fundamental unit that represents a single independent clause is a frame (*VerbFrame*) that captures:

(i) the action that is conveyed by the sentence (*VerbGroup*, which includes a single verb group and auxiliaries),
(ii) the entities that are involved in the action (*Subject* and *Object* – see subsection 3.2)
(iii) any adverbials (*Modifier* – see subsection 3.3)

Simple sentences, which have a single independent clause, are represented by an *IndepClause*, which in turn contains a *VerbFrame*.

3.1.2 Representing Dependent Clauses – Complex Sentences

The *DepClause* modifier when combined with an *IndepClause* allows us to represent complex sentences (see Figure 3).

DepClause differs from *IndepClause* in the following ways: (i) *VerbFrame* is replaced by *FragFrame* to indicate the fact that the verb group is optional, and (ii) it allows a subordinator (*Word*) in front of the *FragFrame* -- subordinators are words such as *wh*-words, *although, because*, etc.

3.1.3 Representing Compound Sentences

Compound sentences are represented by connecting clauses with a *Conjunct*. *Conjunct*s include *and*, *but*, *or*, etc., and also correlative conjunctions such as *not only-but also* and *if-then* (see Figure 4).

3.2 *Subject* and *Object*

Subject and *Object* in our representation have a more general interpretation than usual. *Subject*s are phrases (*NounPhrase*) or clauses with nominal function, which occur before the verb group (*VerbGroup)* in the sentence. Similarly, *Object*s are phrases or clauses with nominal function, which occur after the *VerbGroup* in the sentence. Thus, complements are also represented as *Subject*s or *Object*s. These are simply syntactic entities. We do not attempt to determine semantic roles here.

*Subject*s and *Object*s may themselves be clauses. Figure 5 shows such an example.

3.3 *Modifier*s

*Modifier*s can be attached to any component of a clause (*Modifier* within *Subject*, *Object*, *VerbFrame*, and *FragFrame*) or to a complete clause as a whole (*Modifier* within *IndepClause* and *DepClause*). This recursive nesting of clauses can be used to represent complexity of any arbitrary level.

For simplicity, we club all modifiers except prepositional phrases as *DepClause*, in which phrases are represented as verbless clauses. Prepositional phrases are represented using *PrepPhrase*.

3.4 Representing Incomplete Sentences and Fragments

We define *Fragment* to represent incomplete sentences which end at chunk boundaries. This is used to handle captions, titles, news headlines etc., which are usually not full sentences.

Incomplete sentences that do not end at chunk boundaries usually cannot be parsed in full by the sentence-structuring algorithm. *MixedBag* is a fallback option that allows us to represent such structures. Any identified clause will be structured in the usual manner, while other parts of the sentence will be represented as chunks (see Figure 6).

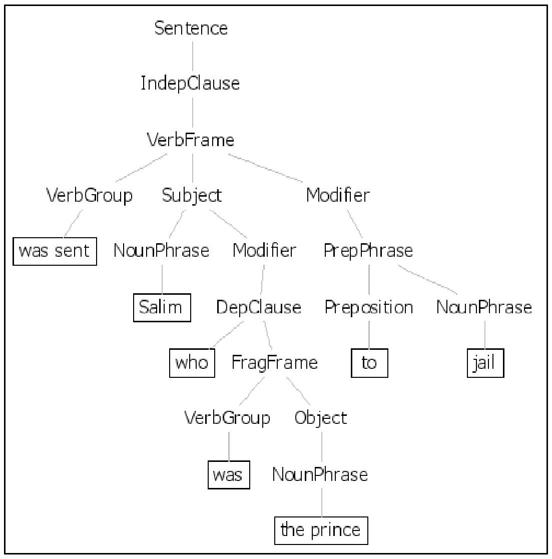

Fig 3: MSIR representation for a sentence containing a dependent clause: *Salim, who was the prince, was sent to jail*

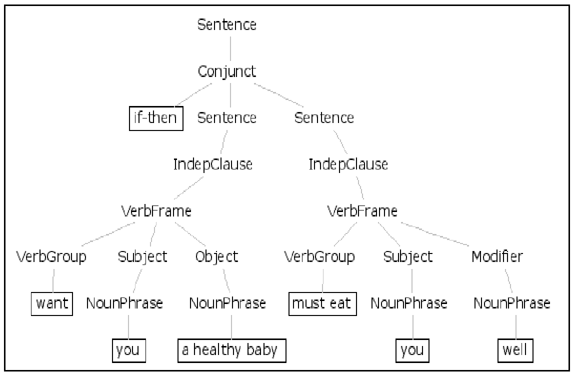

Fig 4: MSIR representation for a compound sentence: *If you want a healthy baby, you must eat well.*

The MSIR is a simple representation, which is reasonably easy to attain for most sentence types, using a straightforward parsing algorithm as discussed in the next section. This representation is admittedly rather coarse, and supplies much less syntactic and semantic information to the

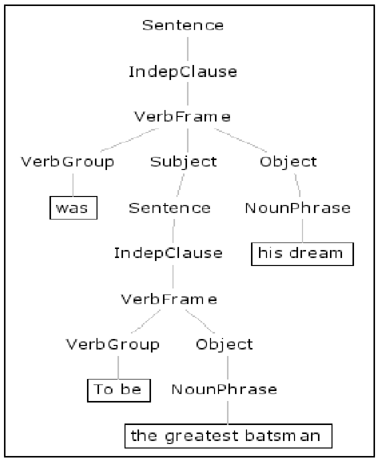

Fig 5: MSIR representation for a sentence containing a clause as a subject: *To be the greatest batsman was his dream*

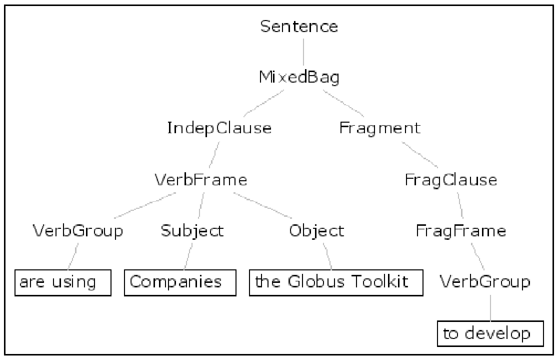

Fig 6: MSIR representation for a fragment not ending at a chunk boundary: *Companies are using the Globus Toolkit to develop.*

generation component than most existing grammars. As described, many categories of sentence elements have been clubbed together for simplicity of representation and parsing. For instance, except for prepositional phrases, modifiers are not distinguished, and complements are clubbed with *Subject* or *Object*. Semantic roles are largely ignored. However, based on a preliminary evaluation of MaTra, it is our contention that most of these details are not essential to English-Hindi indicative MT (see section 5 for evaluation details). We are working on larger scale and more thorough evaluation to substantiate this claim.

Hindi generation from the MSIR (structural and lexical transfer) is described in detail in [Rao et al., 1998; Rao et al., 2000; Mehta and Rao, 2003].

4 The Parsing Algorithm

The source sentence is first processed for punctuations, numerals, contractions, acronyms, and other abbreviations. The sentence is then chunked. Since the clause is the basis of the MSIR, the various clauses are then identified; this is done by the clause boundary detection component. Each clause is then structured independently as discussed in section 3.1. The MSIR for the whole sentence is finally obtained by putting these structured clauses together in a manner that adheres to the MSIR.

Thus, the parsing stage can be divided into three major components: (i) the POS tagging and chunking component, (ii) the clause boundary detection component, and (iii) the sentence-structuring component.

4.1 POS Tagging & Chunking

POS tagging and chunking are done using the fnTBL POS tagger and chunker [Ngai and Florian, 2001], which is based on rules learnt using Transformation Based Learning (TBL) on the Wall Street Journal corpus. The POS tagger chooses the most probable part-of-speech tag (from the UPenn tagset) for each word. The text chunking component of fnTBL groups the words into one of Noun, Verb, Adjective, and Preposition groups.

The following is an example of fnTBL output:

(Salim$_{NNP\ NP}$) (, , -) (who$_{WP\ NP}$) (was$_{VBD\ V}$) (the$_{DT}$ prince$_{NN\ NP}$) (, , -) (was$_{VBD}$ sent$_{VBN\ VP}$) (to$_{TO\ PP}$) (jail$_{NN\ NP}$)

4.2 Clause Boundary Detection

The clause boundary detection works on the chunked sentence. It identifies the various clauses present in a well-formed sentence based on verbs and clause boundary markers called sentinels (*who, whom, which,* etc.) [Narayana Murthy, 1996] as described below:

```
Begin
        if verb not present
            label as Fragment
        End if

        if sentence is well-formed
            for each coordinating conjunction
                label left clause of coord conjunction
                  as IndepClause
            End for
           label the rest of sentence as IndepClause
            for each sentinel
                extract subordinate clause and label
                  as DepClause
                mark the position where the the
                  subordinate clause was originally
                  present
            end for
        End if
End
```

The following is an example of a clause boundary output:

{(Salim$_{NNP\ NP}$) (, , -) DepClause#1 (, , -) (was$_{VBD}$ sent$_{VBN\ VP}$) (to$_{TO\ PP}$) (jail$_{NN\ NP}$)}

DepClause#1={(who$_{WP\ NP}$) (was$_{VBD\ VP}$) (the$_{DT}$ prince$_{NN\ NP}$)}

4.3 Sentence Structuring

The last stage of parsing is the sentence-structuring component. This takes the clause boundaries and builds the MSIR from it.

The algorithm for structuring the clauses and putting these structured clauses together to obtain the MSIR is as described below.

```
SentStruct()
Begin
        For each clause
            ClauseStruct()
        End For
        If no well-formed combination of clauses and
        phrases
            Create a tree with an unstructured
              collection of clauses and
              independent phrases

        Else

           If Conjunct present
               Root ← Conjunct
           End If
           Attach each Structured Fragment or
             IndepClause to Root/Conjunct
          Attach dependent clauses to their
             respective IndepClauses
         Attach Clause Modifiers to respective
             clauses
       End if
End
```

Each clause (with a single verb group) is then structured in the following way:

```
ClauseStruct()
Begin
    If not Fragment
        Attach the verb group to the root

        Search and attach Noun group occurring
          before the verb group as Subject

        Search and attach prepositional/noun groups
        as modifiers to the Subject
```

```
            Search and attach Noun groups occurring
              after the verb group as *Object*. The
              ordering determines whether the object is
              direct or indirect.

            For each object identify and attach
              prepositional groups as
              modifiers to the object

            The remaining groups are attached as
              modifiers to the verb group
    else
            If verb group present
                Root← verb group
                Search and attach verb
                      group modifiers
                Search and attach noun
                      group as object
                Search and attach prepositional
                      groups as modifiers
            else
                Root ← object
                Search and attach noun
                      group as object
                Search and attach prepositional
                      groups as modifiers
End
```

The parsing algorithm currently does not support imperative and interrogative sentences. Also, the implementation for non-finite clauses is not complete.

5 Evaluation Strategy & Results

A preliminary test set of 315 sentences has been created by selecting sentences from news archives and other sources. This test set is in two parts:

The first part of the test set contains declarative sentences and fragments exhibiting certain basic grammatical phenomena, for example, sentences with: (i) different types of clauses in various roles (ii) different types of phrases, and (iii) all tense-aspect-modality combinations. Evaluation shows that the MSIR and parsing algorithm can handle such sentences well.

The second part includes sentences and fragments with (i) interrogative, subjunctive, and imperative mood, (ii) phrasal verbs, (iii) elliptical clauses (iv) idioms, etc. This part is intuitively more difficult for MT. We are currently working on extending the MSIR and parsing algorithm to accommodate these phenomena.

The complete test set of 315 sentences was used to evaluate the translations produced by MaTra. This serves as an indication of the effectiveness of the MSIR, parsing algorithm, and Hindi generation component.

BLEU [Papineni et al., 2002] and NIST evaluations [Doddington, 2002] were done using the NIST evaluation toolkit[1]. One reference translation was used for each sentence. N-grams (up to 4-grams) were matched after stemming.

Table 1 shows the scores for the current version of MaTra (called MaTra2) and the previous version. The previous version, in addition to imperative and interrogative sentences, also did not have support for fragments and incomplete sentences. Non-finite clauses were also not supported. The scores suggest that substantial improvement has been made in MaTra2 over the previous version.

	Matra (Feb 05)	*Matra2* (Mar 06)
BLEU	0.0377	0.0534
NIST	2.1261	3.1494

Table 1: BLEU and NIST scores

Manual inspection indicates that there are differences in lexical choice in the reference and system translations. Structural differences also exist due to the free word-order of Hindi. These are partly the reason for the low absolute scores. To improve the automatic evaluation process, we are increasing the number of reference translations for BLEU/NIST. We are also looking at other automatic evaluation strategies that perform matching of linguistic phrases rather than n-grams. Such a strategy would perhaps be more suited for a free word-order language like Hindi.

Subjective evaluation was performed by two native Hindi speakers, who are also proficient in English. These judges were able to identify most transliterated words, which may not be true for a person not knowing English. They graded translations on the following 4-point scale [Sumita et al., 1999]. Translations marked as one of A, B, and C can be considered acceptable:

A) Perfect: no problems in both information and grammar
B) Fair: easy-to-understand with some unimportant information missing or flawed grammar

[1] http://www.nist.gov/speech/tests/mt/resources/scoring.htm

C) Acceptable: broken but understandable with effort
D) Nonsense: important information has been translated incorrectly

Grade	% of sentences
A	12.7 %
(A + B)	37.1 %
(A + B + C)	65.4 %

Table 2: Subjective evaluation scores

Table 2 shows the results of the subjective evaluation. Though the number of perfect translations is low (12.7%), it is highly encouraging that more than 65% of the translations were rated as acceptable.

6 Conclusion

In this paper, we have described a practical strategy for designing an English-Hindi machine translation system. The system is geared towards translating real-world, general-purpose texts with a high level of noise, such as those on the Web. The design of the system represents a pragmatic approach to engineering a usable system, which aims to produce understandable output for wide coverage, rather than perfect output for a limited range of sentences. The design is based on an intuitive intermediate representation (MSIR) that especially simplifies the parsing algorithm, as described in the paper. The approach allows convenient enhancement of the linguistic capabilities of the translation system, while making it possible for us to produce acceptable translations even as the system evolves.

Though at an early stage of implementation, preliminary evaluation of the system is very encouraging -- more than 65% of translations were rated as acceptable by human judges. The paper also reports automatic evaluation results using BLEU and NIST.

Future work will look at handling interrogative, subjunctive and imperative moods, phrasal verbs, idiomatic usages, etc. Larger scale and component-wise evaluation of the system is also being planned.

References

Doddington, G. 2002. *Automatic Evaluation of Machine Translation Quality Using N-Gram Co-Occurrence Statistics*, Proceedings of the Second International Conference on Human Language Technology, 2002.

Mishra A., *Discovering Rules for Transliteration from English to Hindi: A Genetic Algorithms approach*, MCA thesis, SSIT Orissa, 2004.

Narayana Murthy, K., *Universal Clause Structure Grammar*, Ph.D. thesis, Dept. of Computer and Information Sciences, University of Hyderabad, 1996.

Mehta V. and Rao D., *Natural Language Generation of Compound-Complex Sentences for English-Hindi Machine Aided Translation*, Proceedings of the Symposium on Translation Support Systems (STRANS) 2003.

Mohanraj K., Hegde J., Dogra N., Ananthakrishnan R., *The MaTra Lexicon*, Technical Report, CDAC Mumbai, 2003.

Ngai G., and Florian R., *Transformation-Based Learning in the Fast Lane*, Proceedings of North American Association for Computational Linguistics (NAACL), 2001.

Papineni, Kishore, Salim Roukos, Todd Ward, and WeiJing Zhu, 2002. *Bleu: A Method for Automatic Evaluation of Machine Translation*, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), 2002.

Rao D., Bhattacharya P. and Mamidi R., *Natural Language Generation for English to Hindi Human-Aided Machine Translation*, Proceedings of the International Conference on Knowledge Based Computer Systems (KBCS), 1998.

Rao D., Mohanraj K., Hegde J., Mehta V., and Mahadane P., *A Practical Framework for Syntactic Transfer of Compound-Complex Sentences for English-Hindi Machine Translation*, Proceedings of the International Conference on Knowledge Based Computer Systems (KBCS), 2000.

Sumita E., Yamada S., Yamamoto K., Paul M., Kashioka H., Ishikawa K., and Shirai S., *Solutions to Problems Inherent in Spoken-Language Translation: the ATR-MATRIX Approach*, Proceedings of MT Summit VII, 1999.