Bootstrapping of a Rule Based English – Bangla Machine Translation System using work done for a sister language

Supervisor: Prof. Dr. Mumit Khan
Co-supervisor: Matin Saad Abdullah

Conducted by:
E. M. Yeaseenur Rahman Tahin
09201024

School of Engineering and Computer Science
BRAC University

Submitted on
August 2010

DECLARATION

I, E. M. Yeaseenur Rahman Tahin, student of Computer Science and Engineering department, BRAC University represent my thesis work on “Bootstrapping of a Rule Based English – Bangla Machine Translation System using work done for a sister language” as requirement of completion of bachelor degree. This thesis research was performed under supervision of Dr. Mumit Khan, Professor, BRAC University, Dhaka, Bangladesh and co-supervised by Matin Saad Abdullah, Senior Lecturar, BRAC University, Dhaka, Bangladesh.

This is to declare that the thesis work was done by me and it has not been submitted before. Help that was taken from internet and books was mentioned at references.

Signature of Supervisor

_____

Prof. Dr. Mumit Khan

Signature of Author

_____

E. M. Yeaseenur Rahman Tahin

ACKNOWLEDGMENTS

I am grateful to my Almighty Lord for blessing me with the patience and knowledge and the opportunity to learn something new.

I am heartily thankful to my thesis supervisor Dr. Mumit Khan for his belief in me and pushing me to do better.

I am thankful to my Co-supervisor, Matin Saad Abdullah for his encouragement, guidance and support from the initial to the final level to enable me to develop my understanding and complete my thesis. He inspired me and gave solutions to problems I could not solve by myself.

I am thankful to all my teachers, especially to Annajiat Alim Rasel for giving me suggestions and advices.

I would also like to thank ‘Center for Research on Bangla Language Processing (CRBLP)’ for helping me by providing resources that I needed.

And last but not the least I thank my family and all my friends for supporting me in times of need.

Abstract

Bootstrapping of a Rule Based English – Bangla Machine Translation System using work done for a sister language

The idea of machine translation is not new and started from 17th century. The purpose of machine translation is to translate one human language to other human language. By saying human language it means- Bangla, English, Spanish, Hindi etc.

Human language is ambiguas. So, a perfect machine translation system is very difficult to develop and yet to be done. Companies like Yahoo, Google etc. has their own research team researching on this issue to make machine translation more perfect. The whole idea of machine translation is not to do the perfect translation but to generate understandable translated sentences with the same meaning as the source sentences.

English is the most researched language in this field and it is now possible to translate English to many other languages and vice versa. However, for Bangla no good translator is done yet. 'Anubadok' was a good attempt but it lacks proper computational grammar for Bangla. So, I decided to work on a rule based machine translation system for English to Bangla machine translation.

Recently 'Center for Research on Urdu Language Processing (CRULP)' open sourced their English to Urdu machine translation system. As Urdu is a sister language of Bangla there is a lot of similarities between these two languages in both sentence structure and grammar. So, in my thesis semester I studied their system and figured out the way to use their system for English to Bangla translation.

This report contains all steps for downloading and setting up the system, how that system works and how we can use this system for translating English to Bangla.

TABLE OF CONTENTS

List of Figures

List of Tables and Charts

Literature Survey

The idea of Machine Translation is being developed since 17th century. However, for Bangla research started few years back. I came to know that while I was studing for my thesis. I studied the common approaches of Machine Translation and the existing sytems for translating English to Bangla. Few sources that I used for my study is mentioned detail in this chapter. Rest are mentioned in the reference chapter of this report.

As my goal is to develop a Rule Based Machine Translation System for English to Bangla translation, my reading materials are also related to that topic.

"A brief introduction to 'Anubadok' (The Bengali Machine Translator)" by Golam Mortuza Hossain

'Anubadok' is the first English to Bangla Machine Translation System developed by Golam Mortuza Hossain. I found this paper very helpful to understand how 'Anubadok' works. It provides a detail and step by step demostration of 'Anubadok'. It gave me the basic idea of an English to Bangla Machine Translation System.

"*Semi-Automatic Lexical Functional Grammar Development*" by Umer Khalid, Nayyara Karamat, Shahid Iqbal and Sarmad Hussain
Center for Research in Urdu Language Processing

This gave me a basic understanding about the term Lexical Functional Grammar (LFG), how to form this type of grammar and how that is important for Machine Translation.

"AUTOMATIC LFG GENERATION" by Umer Khalid Qureshi (MS Thesis)
National University of Computer and Emerging Sciences

This thesis report gave me a detail and deep knowledge about Urdu computational grammar, Urdu sentence formation, Lexical Functional Grammar (LFG) and how it is parsed.

Syntax Analysis and Machine Translation of Bangla Sentences by Md. Musfique Anwar, Mohammad Zabed Anwar and Md. Al-Amin Bhuiyan
Dept. of Computer Science & Engineering, Jahangirnagar University, Bangladesh
From this I got a basic understanding about the Bangla grammar and sentence formation

There are other publications and websites from where I have taken information. Links of those resources are given in reference section.

Chapter 1: Introduction

1.1: Machine Translation

Machine translation (MT) is automated translation. It is the process by which computer software is used to translate a text from one natural language (such as English) to another (such as Bangla).

To process any translation, human or automated, the meaning of a text in the original (source) language must be fully restored in the target (foreign) language. While on the surface this seems straightforward, it is far more complex. Translation is not a word-for-word substitution. A translator must interpret and analyze all of the elements in the text and know how each word may influence another. This requires extensive expertise in grammar, syntax (sentence structure), semantics (meanings), etc., in the source and target languages, as well as familiarity with each local region.

Improved output quality can also be achieved by human intervention; for example, some systems are able to translate more accurately if the user has unambiguously identified which words in the text are names. With the assistance of these techniques, MT has proven useful as a tool to assist human translators. But the greater challenge lies in how machine translation can produce publishable quality translations.

1.2: Common approches of Machine Translation

Some common approaches of Machine Translation are –

- *Rule based* machine translation
- *Statistical* machine translation
- *Hybrid* machine translation

Rule based Machine Translation

A rule based machine translation system consists of -

- collection of rules called grammar rules
- lexicon
- software programs to process the rules

Rule based approach is the first strategy ever developed in the field of machine translation. This process requires extensive lexicons with morphological, syntactic, and semantic information, and large sets of rules. Rules are written with linguistic knowledge gathered from linguists. Rules play major role in various stages of translation: syntactic processing, semantic interpretation, and contextual processing of language.

In most cases, there are two steps: an initial investment that significantly increases the quality at a limited cost, and an ongoing investment to increase quality incrementally. Users can improve the out-of-the-box translation quality by adding their terminology into the translation process. They create user-defined dictionaries which override the system's default settings.

Statistical Machine Translation

Statistical machine translation tries to generate translations using statistical methods based on bilingual text corpora. Building statistical translation models is a quick process, but the technology relies heavily on existing multilingual corpora. Where such corpora are available, impressive results can be achieved translating texts of a similar kind, but such corpora are still very rare. A minimum of 2 million words for a specific domain and even more for general language are required. Additionally, statistical machine translation is CPU intensive and requires an extensive hardware configuration to run translation models for average performance levels.

The first statistical machine translation software was from IBM. Google used rule-based MT system for several years, but switched to a statistical translation method in October 2007.

Hybrid MT

Hybrid machine translation is a combination of both statistical and rule-based translation methodologies.

1.3: My motivation

There is immense knowledge out there in the Internet. But most of the people in our country can not take advantage of it because of the language barrier. Today we have many Bangla websites and blogs but those are not enough to satisfy our need for knowledge. So, I myself thought that an English–Bangla translator will be great to make much but not all information meaningful to all people in Bangladesh.

An English-Bangla translator might be useful in various ways-

- Automatic sub-titling (close captioning) in Bangla for YouTube videos
    - If Hindi is already available then why not Bangla?
- Official websites of government or online newspapers can use it
- Different international companies can use it to localize (I18N) their services

Chapter 2: Existing System Overview

As the idea of Machine Translation is very new for Bangla, very few people worked on it before. That is why 'Anubadok' is the only existing system of this kind.

2.1: "Anubadok"

"Anubadok" in the only existing system for English to Bangla Translation. It was developed by Golam Mortuza Hossain.

It can be visited from the following link –
http://bengalinux.sourceforge.net/cgi-bin/anubadok/index.pl

It can be downloaded from the following link –
http://anubadok.sourceforge.net/download.htm

"Anubadok" is implemented in Perl. Following program illustrates the usage of "Anubadok" modules in a Perl program.

```
#!/usr/bin/perl

use strict;
use encoding "utf8";

use Anubadok::XMLPP;
use Anubadok::PoSTagger;
use Anubadok::Translator;

print STDOUT
    XMLPP::xml_post_processor(
        Translator::translate_in_bengali(
            PoSTagger::penn_treebank_tagger(
                XMLPP::xml_pre_processor(<STDIN>))));
```

Figure 2.1: Perl implementation of 'Anubadok'

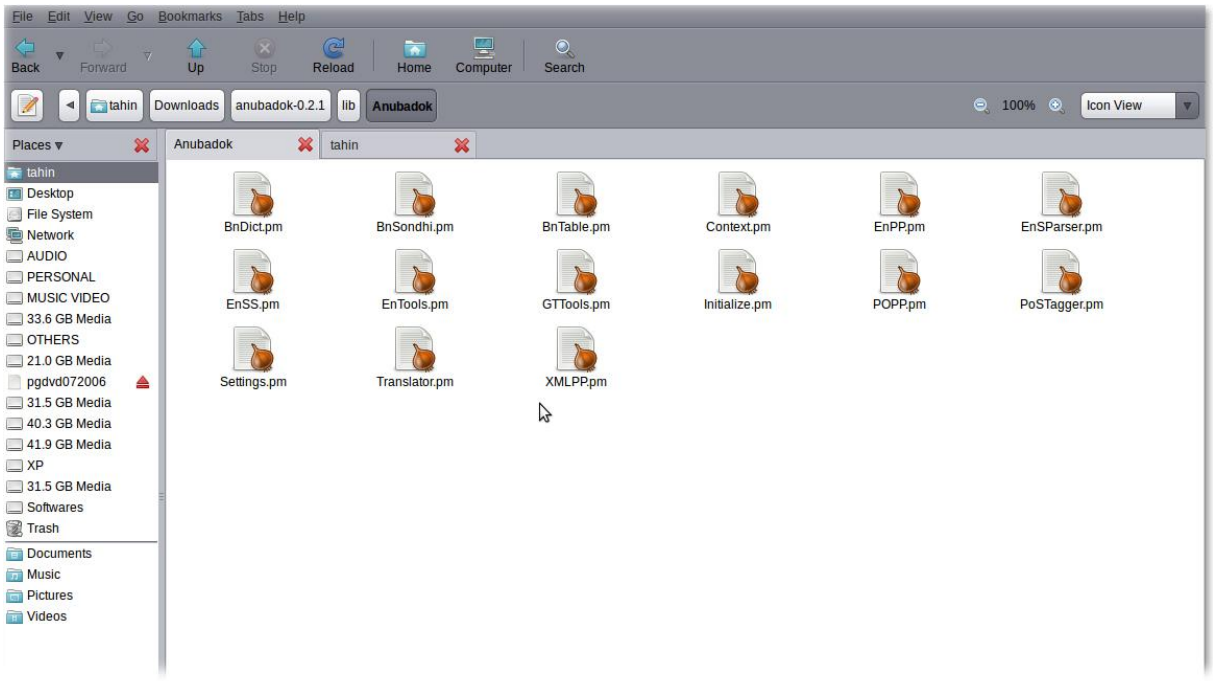


Figure 2.2: Screenshot of all “Anubadok” files

2.2: How “Anubadok” works

It can be described in four steps –

1. Pre-processing of English documents
2. Parts of speech (POS) tagging of pre-processed documents
3. English to Bengali translation of POS-tagged documents
4. Post-processing of translated documents

2.2.1: Pre-processing of English Document

In this step, the system will take input from any plain text, html page, xml or portable object file and extract the original content of it

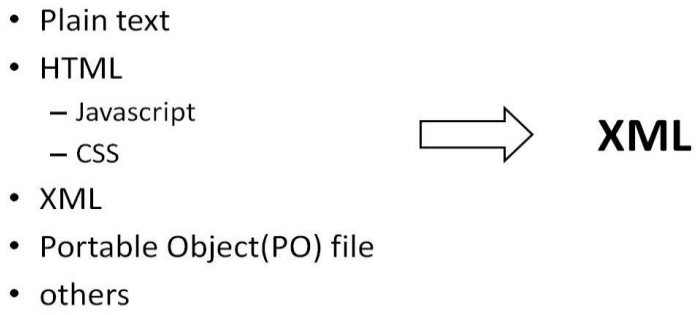


Figure 2.3: Pre-processing in ‘Anubadok’

For example, from the following html line it will extract only “I am reading a book”.

I am reading a book.

2.2.2: Parts of speech (POS) tagging of pre-processed documents

Parts of speech tagging of a document is done in three stages –

- Stage 1: The entire English document is tokenized
- Stage 2: Parts of speech tagging
- Stage 3: The tagged document is then lemmatized by a lemmatizer

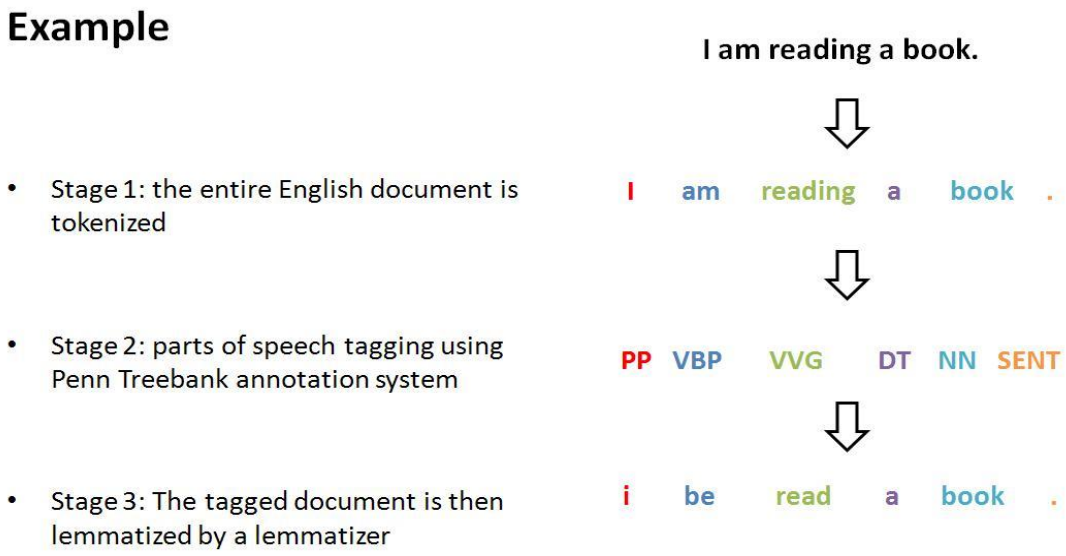


Figure 2.4: POS tagging example in 'Anubadok

2.2.3: English to Bengali translation of POS-tagged documents

Translation is done in six different steps –

1. Sentence type determination
2. Subject, Object and Verb determination
3. Tense determination
4. Subject and object translation
5. Verb translation
6. Construction of final Bengali sentence

2.2.3.1: Sentence type determination

Following sentence types are considered in this system –

– Declarative (default)
– Imperative
– Interrogative
– Exclamatory

2.2.3.2: Subject, Object and Verb determination

In the next logical step, “Anubadok” determines subject, object and verb of a given English sentence.

Example –

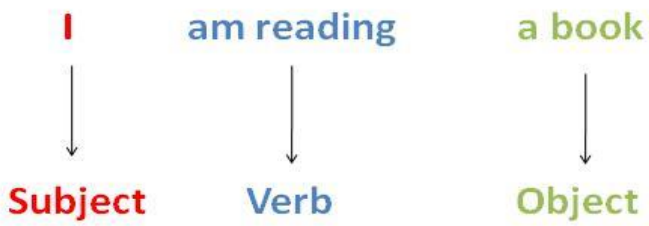
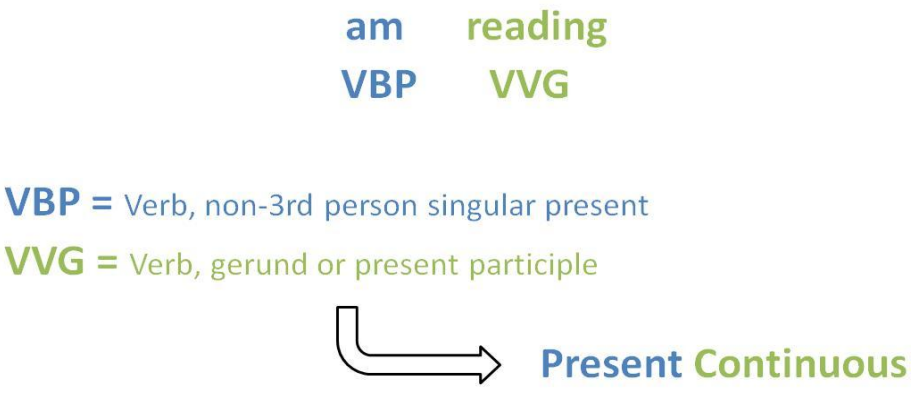


Figure 2.5: SVO determination in ‘Anubadok

2.2.3.3: Tense determination

“Anubadok” determines tense forms of an English sentence, from the Penn tags of the verb tokens. In the example sentence, the verb part contains two tags "VBP" and "VVG". The first tag VBP indicates present tense where as the second tag VVG implies the tense form to be continuous. Together they determine the tense of the example sentence as being present continuous.



2.2.3.4: Subject and Object translation

After determining subject, object and verb of a sentence, “Anubadok” proceeds to translate subject, object and verb separately. During subject translation, “Anubadok” determines the person of the sentence.

Example – Here "I" (personal pronoun) is the subject and it is determined to be *first person*.

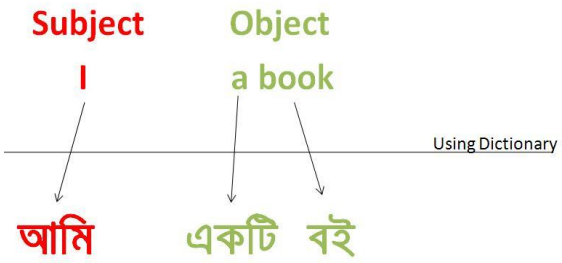


Figure 2.6: Translation example in ‘Anubadok

2.2.3.5: Verb translation

Actual verb depends on -

- Base form of the verb (root verb)
- Person
- Tense
- Whether verb is active or passive

“Anubadok” derives the actual verb by concatenating scalar root verb and the universal suffix matrix.

	Simple	Continuous	Perfect	Perfect Continuous
Present	ি	ছি	েছি	ছি
Past	েছিলাম	ছিলাম	েছিলাম	ছিলাম
Future	ব	ব	ব	ব

Actual verb matrix = Scalar root verb X Universal suffix matrix

Figure 2.7: Verb translation in 'Anubadok

Example –

পড়ছি = পড় X ছি

Actual verb = Root verb X Actual suffix

Example of a universal suffix matrix

: খা ()

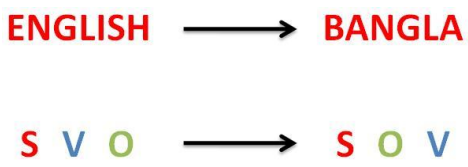
: পা ()

খা ()		1st	2nd			3rd	
Verbal noun (present participle/gerund)	ওয়া	N/A	Polite	Familiar	More Familiar	Polite	Familiar
			N/A	N/A	N/A	N/A	N/A
Infinitive (to do)	খেতে [!]/ বার জন্য	N/A	N/A	N/A	N/A	N/A	N/A
Present simple	N/A	ই	ন	ও	স	ন	য়
Present continuous	N/A	ছি	ছেন	ছ	ছিস	ছেন	ছে
Future simple	N/A	ব	বেন	বে	বি	বেন	বে
Simple past	N/A	খেলাম [!]	খেলেন [!]	খেলে [!]	খেলি [!]	খেলেন [!]	খেল [!]
Habitual past (Imperfect)	N/A	খেতাম [!]	খেতেন [!]	খেতে [!]	খেতি [!]	খেতেন [!]	খেত [!]
Conditional past (... would eat)	N/A	ব	বেন	বে	বি	বেন	বে
Continuous past	N/A	ছিলাম	ছিলেন	ছিলে	ছিলি	ছিলেন	ছিল
perfect	N/A	খেয়েছি [!]	খেয়েছেন [!]	খেয়েছ [!]	খেয়েছিস [!]	খেয়েছেন [!]	খেয়েছে [!]
pluperfect (past perfect)	N/A	খেয়েছিলাম [!]	খেয়েছিলেন [!]	খেয়েছিলে [!]	খেয়েছিলি [!]	খেয়েছিলেন [!]	খেয়েছিল [!]
Past participle	খেয়ে [!]	N/A	N/A	N/A	N/A	N/A	N/A
Conditional participle	খেলে [!]	N/A	N/A	N/A	N/A	N/A	N/A
Present imperative	N/A	N/A	ন	ও	[-]	N/A	N/A
Future imperative	N/A	N/A	খেয়েন [!][?]	খেও [!]	স	N/A	N/A
Future Continuous	N/A	খেতে থাকব [!]	খেতে থাকবেন [!]	খেতে থাকবে [!]	খেতে থাকবি [!]	খেতে থাকবেন [!]	খেতে থাকবে [!]
(... might have eaten)	N/A	খেয়ে থাকব [!]	খেয়ে থাকবেন [!]	খেয়ে থাকবে [!]	খেয়ে থাকবি [!]	খেয়ে থাকবেন [!]	খেয়ে থাকবে [!]

Table 2.1: Universal suffix matrix in 'Anubadok

2.2.3.6: Construction of final Bangla sentence

"Anubadok" joins the subject, object and verb together to form the final Bengali sentence in the S-O-V order.



Example -

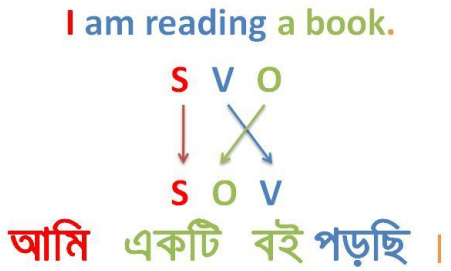


Figure 2.8: Constructing sentence in 'Anubadok

2.2.3.7: Post-processing of translated documents

This is the last stage before "Anubadok" writes out the translated final output. In this stage, it reverts the changes that were made in pre-processing stage to preserve certain formatting information of the documents.

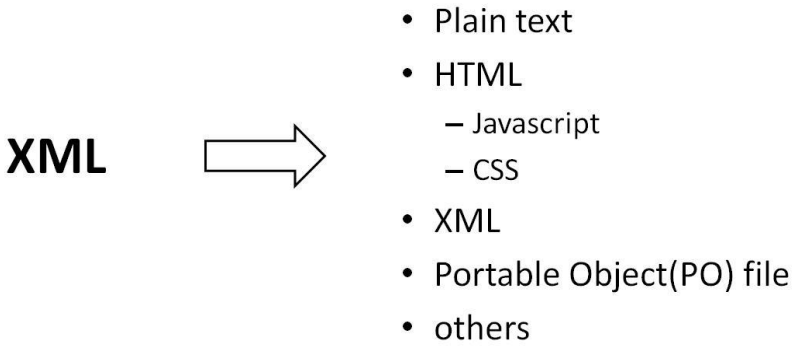


Figure 2.9: post-processing in 'Anubadok

2.3: Limitations of this system

- Limited word database
- Words are not properly tagged (singular/plural are not tagged for all nouns)
- Has many limitations (like, excluding adjectives that can be used as nouns)
- Limitations of rules:

  He came to me - সে আমার কাছে এসেছিল

  “Anubadok” translates -

  He came to me – সে আমাকেতে আসেছিল

Status Table of “Anubadok”-0.2.0

Status Table (Version: Anubadok-0.2.0)

	Declar.	Imper.	Interro.	Exclam.
Simple	W	W	W	M
Compound	M	M	M	M
Complex	N	N	N	N
Compound – Complex	N	N	N	N

W: Well implemented
M: Moderately implemented
N: Not/Not-well implemented

Table 2.2: Status table for ‘Anubadok’

Chapter 3: English to Urdu MT system developed by CRULP

3.1: About the system

This is a rule based machine translation system developed using Lexical Functional Grammar (LFG) by *Center for Research on Urdu Language Processing (CRULP)*.

3.2: Lexical Functional Grammar (LFG)

Primary structure of LFG has two different parts –

- Grammatical structure (**c-structure**)
- Grammatical functions (**f-structure**)

Grammatical structure (c-structure)

In grammatical structure (c-structure), a sentence of any language is described using phrase structure rules.

For instance: I eat rice

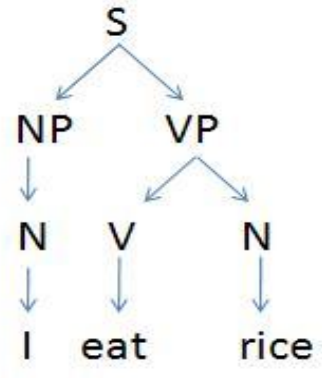


Figure 3.1: C structure

Grammatical functions (f-structure)

Feature structure is a set of attribute-value pair for a given sentence. Few important attributes are –

- Number
- Tense
- Subject
- Predicate
- Object etc.

Steps to make LFG

A sentence is described to its LFG in two steps –

Step 1: Extracted parse tree from a given sentence is converted into its corresponding Context free grammar (CFG).

Step 2: That context free grammar with addition to the feature description (f_description) of that sentence creates the actual Lexical functional grammar (LFG). Feature description (f_description) set of a sentence is derived using some *Meta rules*.

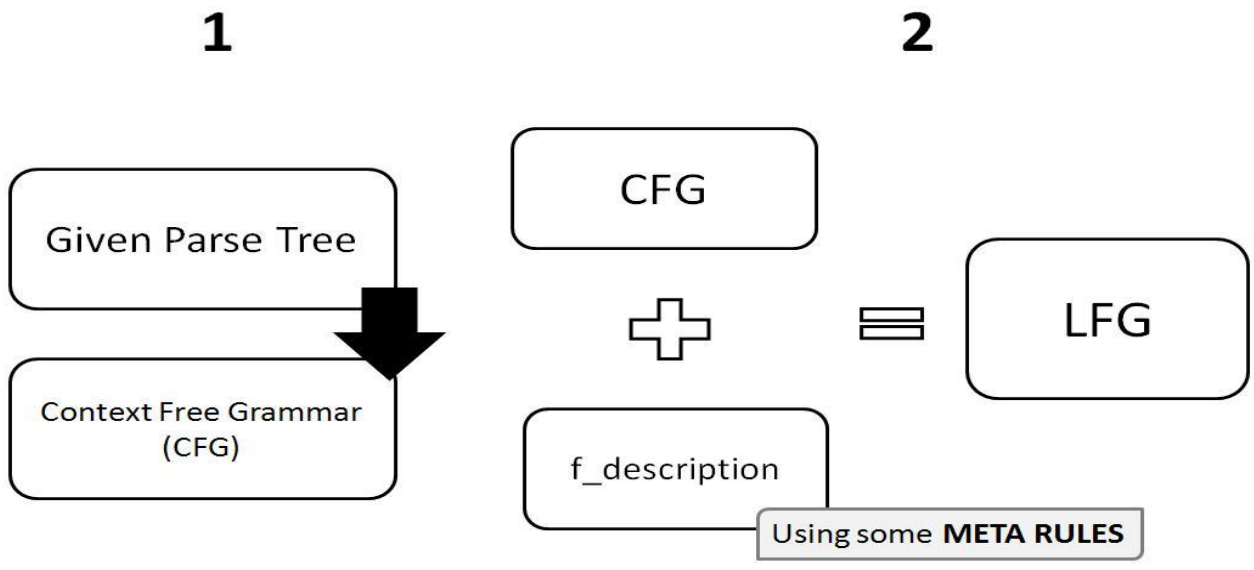


Figure 3.2: Constructing LFG

Meta rules

These are some general rules to find out the feature description of a sentence. These rules are called Meta rules. Regex are used to define these rules.

Format of Meta rules –

Lhs -> Rhs @ Anno1 (@ Anno2)

Here, Lhs and Rhs are regular expressions which are to be matched with the grammar rule to apply the @Anno f-description on it.

Example of a Meta rule –

VP:vp > * MD:m1 * [VP-A:v1|VP:v1] * @[vp:^===m1,vp:^===v1]

3.3: How to run the system

English - Urdu MT source code can be downloaded from CRULP website.

Download link is -
http://www.crulp.org/software/langproc/E2UMachineTranslationSystem.htm

The source code was written in C++. CRULP made only the source codes available, not the other project files. So, a project need to be created to run this system.

Step by step instruction to create a **Dev-C++** project and run is given below -

Step 1: Create a new Dev-C++ project.

Step 2: Add all files (both .cpp and .h) in **Dev-C++** into that project and compile it.

Step 3: Debugging

This first time compilation will show lots of errors and warnings. These errors can be resolved in 4 different steps. Those steps are described as following -

a. **Converting to 'int' from 'double'**

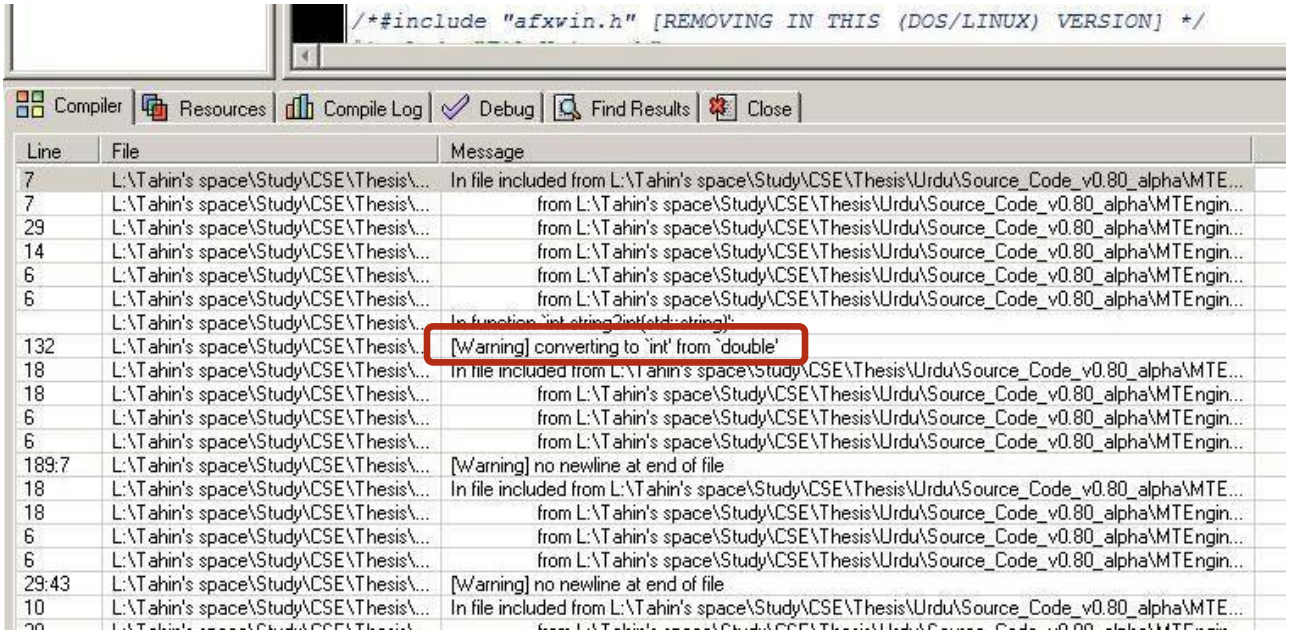


Figure 3.3.1: Debugging: converting ‘int’ to ‘double’

It causes because it is trying to put a *double* value into an ***int*** type variable. Dev-C++ caught it as a warning.

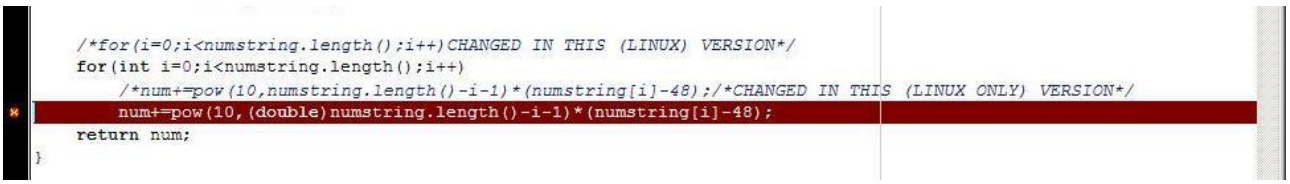


Figure 3.3.2: Debugging: converting ‘int’ to ‘double’

Type cast that *double* value into *int* data type as the following screenshot.

```
    /*for(i=0;i<numstring.length();i++)CHANGED IN THIS (LINUX) VERSION*/
    for(int i=0;i<numstring.length();i++)
        /*num+=pow(10,numstring.length()-i-1)*(numstring[i]-48);/*CHANGED IN THIS (LINUX ONLY) VERSION*/
        num+=(int)pow(10,(double)numstring.length()-i-1)*(numstring[i]-48);
    return num;
}
```

Figure 3.3.3: Debugging: converting 'int' to 'double'

b. const_iterator problem

```
        std::vector<T>::const_iterator it=now.begin();
        for(int cnt1=0;cnt1<size;++cnt1)
        {
            if(cnt1==cnt)
            {
```

Figure 3.4.1: Debugging: const_iterator problem

Add **'*typename*'** before that line of code, otherwise compiler will not understand that, ***const_iterator*** is a type.

Modified line of code should look like the following.

```
//          std::vector<T>::const_iterator it=now.begin();
            typename vector<T>::const_iterator it=now.begin();
            //http://bytes.com/topic/c/answers/488898-troubles-when-using-stl-iterator
            for(int cnt1=0;cnt1<size;++cnt1)
            {
                if(cnt1==cnt)
```

Figure 3.4.2: Debugging: const_iterator problem

c. Memory exceed problem

Running this program in a computer with 1gb RAM might cause a memory exceed problem. Switching to a machine with 2gb RAM should work fine for this program.

d. Resource unavailable

Downloaded source code directory does not contain the resources needed for this program. In other directory there in a folder named *'Data'.* Placing that *'Data'* folder into that source code directory will solve this problem.

After these steps should run and work perfectly with proper output.

3.4: How this system works

This is a Rule Based Machine Translation System which will take some English sentences as input and give the translated Urdu sentences as output.

It does the whole process in 2 steps –

1. Breaking down an English sentence
2. Constructing a translated Urdu sentence

3.4.1: Breaking down an English sentence

This step involves few sub steps –

a. Takes an English sentence as input
b. Parses the Context Free Grammar (CFG) of that sentence and extracts the morphemes
c. Tag each morpheme according to its' Part of Speech (POS).

Example:

For an English sentence *'I am eating mango'* -

```
1. I am eating mango.
2. pro:pro    EMPTY:aux    eat_v:v    mango_n:n
3. CFG parsing completed
4. ----------------------------------------
```

3.4.2: Constructing a translated Urdu sentence

In this step –

a. Using a mapping file, system maps each English morpheme to it's corresponding Urdu morpheme. Thus it finds out the Urdu morphemes for that given input.

b. Using a lexicon or dictionary file and other features (number, gender, person, tense etc.) of that input sentence, system picks out the right word to use in the translation.

c. At last using those Urdu words and the Urdu grammars specified in a grammar file (UGRules.grl), system constructs a complete Urdu sentence. Here appropriate rules picked using different meta rules.

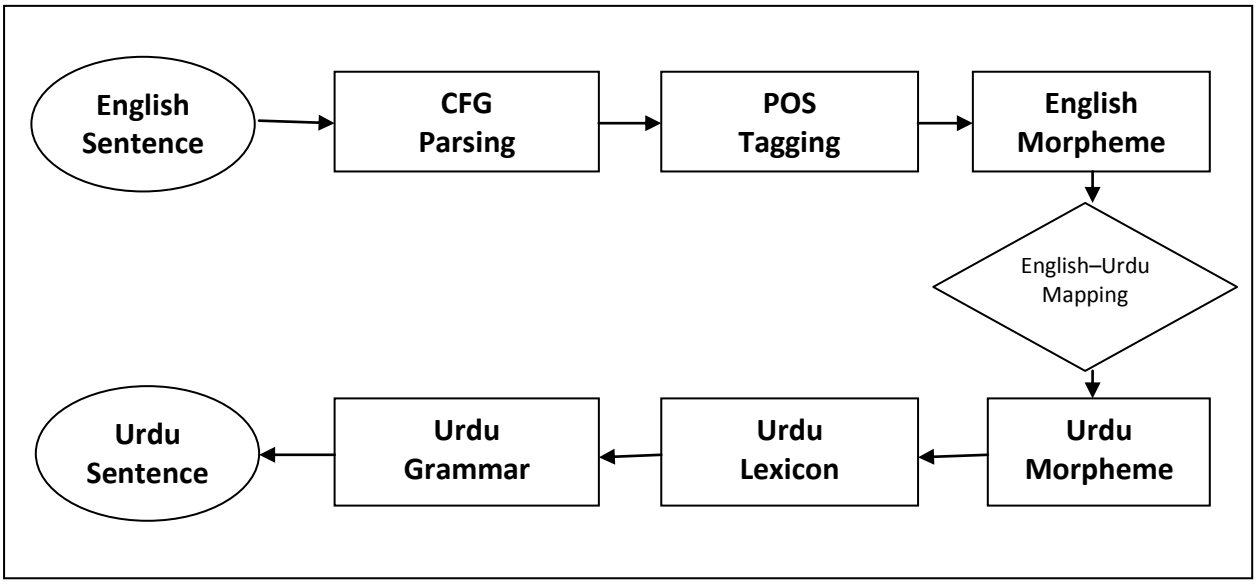


Figure 3.5: Process diagram

Chapter 4: Using that system for English to Bangla translation

In chapter 3 we have seen that English to Urdu translation is done in 2 steps. Where the first step is common if we want to use that for English to Bangla translation. This system can be modified for Bangla by tweaking few files.

For my thesis, I have choosen few sentences for which I changed the lexicon file, mapper file and the grammar file and got Bangla translated output.

Changes that I made are given below.

In “MapLexRaw.mlr” file –

```
mango_n
[
      --> PRED = 'আম;
      //n
]


eat_v
[

      (OBJ.PRED =c 'word_n') --> #VERBAL_NOUN('وایس',ley);

      (OBJ.PRED =c {'heart_n'}) --> #SAME_SUBCAT('جگر');

      (OBJ.PRED =c {'hat_n'}) --> PRED = 'منڈوا', PRED.GF =
PRED.GF, OBJ.PRED = 'موںچھ', OBJ.NOUN = {POS};

//FUTURE: NOT CARTERED IN URDU GRAMMAR, ASP_AUX
//    (OBL.PFORM =c {'into'} && OBL.OBJ.SEM_TYPE =c {ABSTRACT})
--> PRED = 'جا چاٹ';
//    (OBL.PFORM =c {'into'} && OBL.OBJ.SEM_TYPE =c {UNANIM_CONC})
--> PRED = 'جا چاٹ';

      (DEFAULT) --> #SAME_SUBCAT('খা');
]
```

Figure 4.1: Change in “MapLexRaw.mlr”

In “Ulexicon.lex” file –

```
// Pronoun
আমি : pro, ^PRED= 'pro', ^ NUM = SG, ^ GEND= {M,F}, ^ PRONTYPE= PERS, ^ PERS= 1, ^ FORM =
NOM, ^ CASE=NOM, ^ ANIM = POS, ^ RESPECT =NORESPECT.
আমি : pro, ^PRED= 'pro', ^ NUM = SG, ^ GEND= {M,F}, ^ PRONTYPE= PERS, ^ PERS= 1, ^ FORM = OBL,
^ CASE=ERG, ^ ANIM = POS, ^ RESPECT =NORESPECT.


খাচ্ছি : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 1, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PRES.
খাচ্ছ : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 2, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PRES.
খাচ্ছে : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 3, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PRES.
খাচ্ছিলাম : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 1, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PAST.
খাচ্ছিলে : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 2, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PAST.
খাচ্ছিল : v,^PRED ='খা<SUBJ,OBJ>',^_MORPH_FORM = BARE, ^ NUM_MID =
{SG,PL}, ^ NUM_END = {SG,PL},^ GEND ={M, F}, ^ PERS = 3, ^
RESPECT = {NORESPECT,FAMILIAR,USUAL,EXTRA},^SUBJ_CASE=ERG, ^
TENSE = PAST.
```

Figure 4.2: Change in “Ulexicon.lex” file

In “UGRules.grl” file –

If we compare the Urdu sentences with corresponding Bangla sentences we will see some dissimilarities.

	tense_aux (am, is are)	asp_aux (+ing)	verb	obj	case_marker	subj
আমি আম খাই	ہوں		کھاتا	آم		میں
তুমি আম খাও	ہو		کھاتے	آم		تم
সে আম খায়	ہے		کھاتا	آم		وہ
তারা আম খায়	ہیں		کھاتے	آم		وہ
আমি আম খেয়েছিলাম			کھایا	آم	نے	میں
তুমি আম খেয়েছিলে			کھایا	آم	نے	تم
সে আম খেয়েছিল			کھایا	آم	نے	اس
তারা আম খেয়েছিল			کھایا	آم	نے	انھوں
আমি আম খাব	گا		کھاؤں	آم		میں
তুমি আম খাবে	گے		کھاؤ	آم		تم
সে আম খাবে	گا		کھائے	آم		وہ
তারা আম খাবে	گے		کھائیں	آم		وہ
আমি আম খাচ্ছি	ہوں	رہا	کھا	آم		میں
তুমি আম খাচ্ছ	ہو	رہے	کھا	آم		تم
সে আম খাচ্ছে	ہے	رہا	کھا	آم		وہ
তারা আম খাচ্ছে	ہیں	رہے	کھا	آم		وہ
আমি আম খাচ্ছিলাম	تھا	رہا	کھا	آم		میں
তুমি আম খাচ্ছিলে	تھے	رہے	کھا	آم		تم
সে আম খাচ্ছিল	تھا	رہا	کھا	آم		وہ
তারা আম খাচ্ছিল	تھے	رہے	کھا	آم		وہ

Table 4.1: Analysis of Urdu and Bangla sentences

We can see that Urdu has some case markers, tense auxiliary verbs and aspectual auxiliary verbs depending on the tense of the sentence. But these auxiliary verbs are absent in Bangla. So, I had to change Urdu grammar a bit so that it does not look for these auxiliary verbs and choses the main verb according to tense and other features of the sentence

```
VPnonperf -> ComplexP:^ = !,
             ! _MORPH_FORM  =c {HABITUAL,SUBJUNCTIVE},
             ^ NUM = ! NUM_MID;

             TenseAuxP:
             ^ _MORPH_FORM = ! _MORPH ALLOWED_FORM,
             ^ TNS_ASP TENSE = ! TENSE,
             ^ TNS_ASP PROBABLE = ! TNS_ASP PROBABLE,
             ^ NUM = ! NUM,
             ^ RESPECT = ! RESPECT,
             ^ PERS = ! PERS,
             ^ GEND = ! GEND;.
```

Original Urdu Grammar

```
VPnonperf -> ComplexP:^ = !,
             ! _MORPH_FORM  =c {HABITUAL,SUBJUNCTIVE},
             ^ NUM = ! NUM_MID,
             ^ RESPECT = ! RESPECT,
             ^ PERS = ! PERS,
             ^ GEND = ! GEND,
             ^ TNS_ASP TENSE = ! TENSE;.
```

Changed Grammar for Bangla

Figure 4.3: Change in “UGRules.grl” file

After these changes, the system started to give proper Bangla translation for a small set of input sentences.

Chapter 5: Examples

Example Input:	Example Output:	Comment
I eat mango You eat mango He eats mango They eat mango	আমি আম খাই তুমি আম খাও সে আম খায় তারা আম খায়	
I shall eat mango You will eat mango He will eat mango They will eat mango	আমি আম খাব তুমি আম খাবে সে আম খাবে তারা আম খাবে	
I am eating mango You are eating mango He is eating mango They are eating mango	আমি আম খাচ্ছি তুমি আম খাচ্ছ সে আম খাচ্ছে তারা আম খাচ্ছে	
I was eating mango You were eating mango He was eating mango They were eating mango	আমি আম খাচ্ছিলাম তুমি আম খাচ্ছিলে সে আম খাচ্ছিল তারা আম খাচ্ছিল	
I ate mango You ate mango He ate mango They ate mango	আমি নে আম খেয়েছিলাম তুমি নে আম খেয়েছিলাম সে নে আম খেয়েছিলাম তারা নে আম খেয়েছিলাম	Here نے rekram esac a si Grammar for past tense needs more tweak to make this disappear.
Rumon eats mango	রুমোন আম খায়	Here ‘Rumon’ is not a from the lexicon database. So, system is just transliterating this word.

Table 5.1: Input and output examples

Chapter 6: Future Works

- **Develop a complete English - Bangla lexicon mapping**
  This system lacks a complete English to Bangla lexicon mapping database. We need to develop one in future.

- **Develop a complete Bangla dictionary with all possible forms of words and their corresponding features**
  We need a full featured Bangla dictionary as well. We already have a dictionary which we can edit to include feature information to it.

- **Develop a full fledged computational grammar for Bangla**
  A full fledged computational grammar is something that we need very much. It is not wasy to develop one. We need expert linguists and enough time for that.

Chapter 7: Conclusion

This is as far as I did in my thesis semester. Within a very short time I managed to get these outputs which shows a very good opportunity for this system in future. If we can rewrite the lexicon files according to Bangla and can develop our own computational grammar by doing minor change of their one than it would be a huge time saving work done for Bangla. Getting assistance from a linguists would be more helpful.

References

Publications

1. *"A brief introduction to 'Anubadok' (The Bengali Machine Translator)"* by Golam Mortuza Hossain
2. "*Semi-Automatic Lexical Functional Grammar Development*" by Umer Khalid, Nayyara Karamat, Shahid Iqbal and Sarmad Hussain
   Center for Research in Urdu Language Processing
3. *"AUTOMATIC LFG GENERATION"* by Umer Khalid Qureshi (MS Thesis)
   National University of Computer and Emerging Sciences
4. *Syntax Analysis and Machine Translation of Bangla Sentences* by Md. Musfique Anwar, Mohammad Zabed Anwar and Md. Al-Amin Bhuiyan
   Dept. of Computer Science & Engineering, Jahangirnagar University, Bangladesh

Internet

5. http://www.systran.co.uk/systran/corporate-profile/translation-technology/what-is-machine-translation
6. http://en.wikipedia.org/wiki/Machine_translation
7. http://language.worldofcomputing.net/machine-translation/rule-based-machine-translation.html
8. http://en.wikipedia.org/wiki/Machine_translation#Statistical
9. http://www.crulp.org/
10. Wikipedia
    - http://en.wikipedia.org/wiki/Lexical_Functional_Grammar
    - http://en.wikipedia.org/wiki/Phrase_structure_rules
    - http://en.wikipedia.org/wiki/Feature_structure
11. http://"Anubadok".sourceforge.net/
12. http://wiki.apertium.org/wiki/Bengali_and_English