A

brief introduction to

Anubadok

The Bengali Machine Translator

অনুবাদক

স্বয়ংক্রিয়ভাবে ইংরেজী থেকে বাংলায় অনুবাদ করার জন্য একটি কম্পিউটার সফটওয়্যার সিস্টেম

by
Golam Mortuza Hossain

Work in progress: Version July 8, 2008

Please send your comments to gmhossain@gmail.com

"Anubadok: The Bengali Machine Translator" is a computer software system for performing machine translation of English texts into Bengali. *Anubadok* literally means *the one who translates* in Sanskrit. The word has the same meaning also in Bengali and is scripted as "অনুবাদক". The Anubadok system is written in a dynamic programming language known as *Perl*. Perl provides ease of use for text manipulations and it has quite mature support for processing of Unicode encoded text. These were the main reasons for choosing Perl as the preferred programming language. For internal processing of English texts, Anubadok uses the so-called Penn Treebank annotation system for part-of-speech tagging. Penn Treebank system annotates texts by a set of different tags for different parts of speech. Anubadok system writes out translated documents as Unicode encoded Bengali texts.

License of Anubadok System:

Basics

It is easiest to consider explicit examples for exploring the internal working of Anubadok system. For definiteness let us consider the following English sentence:

- *I am reading a book.*

This sentence has *"I"* as subject, *"am reading"* as verb, and *"a book"* as object. We should note that the subject (S) is followed by the verb (V) which is then followed by the object (O). For this reason, structure of basic English sentences is often mentioned as having S-V-O pattern. On the other hand, in Bengali the given sentence is translated as

- আমি একটি বই পড়ছি।

In this Bengali sentence, "আমি" (I) is the subject (S), "একটি বই" (a book) is the object (O) and "পড়ছি" (am reading) is the verb (V). We may note that in contrast to English sentences, basic Bengali sentences follow the so-called S-O-V pattern. Thus, a simple word-for-word translation method for translating English texts into Bengali doesn't work. Let us now use Anubadok to perform machine translation of the sentence. The simplest way to use Anubadok (without installing the machine translator package in the system) is to run the following command in a terminal inside Anubadok package directory

```
~$ echo "I am reading a book." | ./bin/anubadok
```

আমি একটি বই পড়ছি।

Here "echo", a command in unix-terminal, passes the sentence to "anubadok" which is one of the top level executables of Anubadok system. Anubadok then processes the English sentence and prints out the translated Bengali sentence as Unicode encoded texts.

Machine Translation

In performing machine translation of English text documents into Bengali, the Anubadok system uses four logically different steps. These steps are

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.

These steps themselves are composed of several logical sub-steps. In the following sections, these steps are descibed in more details.

Pre-processing of English documents

In today's computer system a document file consists of not only its *contents* but often it contains some internal informaton that are used by client programs either to understand the structure of the document or to understand how to display the document on screen. For example, consider the following element of an HTML document:

```
<font color=red>I am reading.</font>
```

In this HTML element, the actual contents are "*I am reading.*" where as "`<font color=red>`" is a starting tag along with an attribute *color* and its value *red*. This informaton instructs a client program to display the contents upto "`</font>`" using *red* color. Naturally to translate a document file, the translator needs to understand the internal structure of the given document. Otherwise, translated document could become incomprehensible to a client program. For example, if a translator also translates the words "font", "color" and "red" then a web browser will fail to display this element properly.

The Anubadok system can handle several kinds of documents as input including plain text files, any XML documents, HTML files with in-line javascript, CSS. Apart from these, Anubadok is also capable of translating Portable Object (PO) files. In pre-processing step Anubadok basically convert all kinds of documents into XML documents. However, this convertion preserves all structural informaton in special forms. This informaton is restored at the post-processing stage. For the above example element, pre-processing stage of Anubadok system protects all structural informaton and exposes only its contents *"I am reading."* to the machine translator.

Part-of-Speech (POS) tagging

Parts of speech tagging is one of the most crucial steps in a machine translation sequence. In this step *three* different morphological analysis of a given English documents are performed. In the first stage, the entire English document is *tokenized* which is used for parts of speech *tagging* in the next stage. The tagged document is then lemmatized by a *lemmatizer* in the last stage. Anubadok system itself does not perform parts of speech tagging of an English document. Rather it relies on external program for POS-tagging. Anubadok uses Penn Treebank annotation system for internal processing. So in principle, it can work with any Penn Treebank tagger.

Anubadok by default uses `GPoSTTL` which is an enhanced version of Eric Brill's rule-based Parts-of-Speech Tagger. This tagger has built-in Tokenizer and Lemmatizer. Anubadok can also work with Tree-Tagger which is a probabilistic parts of speech tagger developed by Helmut Schmid. TreeTagger can be

used with Anubadok system for non-commerical translation under a license agreement with the author of TreeTagger. However, given its non-free nature TreeTagger is not a preferred tagger for general purpose translation with Anubadok system.

To illustrate the working of a parts of speech tagger, let us perform morphological analysis of our example sentence using `GPoSTTL`. As earlier, the sentence can be passed to the tagger by using the `echo` command in a terminal as

```
~$ echo "I am reading a book." | gposttl
GPoSTTL (Ver. 0.9.1): Tagging...
I       PP      i
am      VBP     be
reading VVG     read
a       DT      a
book    NN      book
.       SENT    .
GPoSTTL: Done
```

In the tagger output above, one may note that the tagger has broken the given sentence into *six* parts which are called *tokens*. These six tokens are "I", "am", "reading", "a", "book" and ".". First token is tagged as "PP" which stands for *personal pronoun*. Here "PP" is an element of the tagset that is used in the Penn Treebank annotation system. Second token is tagged as "VBP" which stands for *verb (be), present tense*. Third token is tagged as "VVG" (*verb, continuous*). Forth and fifth tokens are tagged as "DT" (*determiner*) and "NN" (*noun, singular*). The last token is tagged as "SENT" which means *sentence boundary*. We may also note here that the tagger can determine the base form of the verb "reading" as "read". We will see in the next section that the base form of a verb, known as *lemma* plays an important role in synthesizing translated Bengali sentences.

Translation of tagged English documents.

This is the step where actual translations are performed by Anubadok. Given this step follows the parts-of-speech tagging step, it has access to the English documents which are both tagged and lemmatized. Access to tagged and lemmatized English documents is essential for performing machine translation in Anubadok system. The translation step is composed of several logical sub-steps. The few main sub-steps are described in the following sub-sections.

Sentence type determination

In this logical step, Anubadok determines the properties of a given sentence by looking at the Penn tags of the tokens in the sentence. In particular, it determines whether it is a Declarative, Imperative, Interrogative or Exclamatory sentence. By default, Anubadok treats a sentence as being Declarative unless it is determined otherwise.

Subject, Object and Verb determination

In the next logical step, Anubadok determines subject, object and verb of a given English sentence. The given example sentence "I am reading a book.", has *"I"* as subject, *"am reading"* as verb, and *"a book"* as

object.

Tense determination

In English, tense forms of a sentence are indicated by the presence of auxiliary *be* verbs (*am, is, are, was...*) and *have* verbs (*has, have, had*) along with the forms of main verbs. On the other hand in Bengali, tense forms are encoded in the modification of the main verbs as there are no auxiliary verbs in Bengali. Thus it is very crucial to determine the tense forms of a English sentence accurately. Tense information is then used to *generate* the final forms of Bengali main verbs from the root verbs (base form). 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.*

Subject and object translation

Having determined 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. For the example sentence "I" (personal pronoun) is the subject and consequently the person is determined to be *first person.* By default, Anubadok sets the person to be *third* person. To construct the subject in Bengali, Anubadok makes dictionary lookup for the word "I" in its English to Bengali dictionary database. Anubadok employs the same algorithm for translating object. In the situation where *person* of the sentence is not determined from the subject, Anubadok tries to guess the person from the object. To construct the object of the example sentence in Bengali, Anubadok needs to make two dictionary lookup for the words "a" and "book".

Verb translation

While translating verbs, Anubadok leverages the power of the fact that Bengali derives its inheritance from *Sanskrit.* In Bengali, final forms of verb can be *generated* by knowing the base form of the verb (root verb), the person, the tense forms and whether verb is *active* or *passive.* We will see here that generation of final verb forms in Bengali has striking similarity with *linear algebra* in abstract mathematics. More precisely, construction of final verb forms in Bengali can be viewed as a method of multiplication between a *scalar* quantity and a *matrix.* In this analogy, all final verb forms can be viewed as the elements of resultant matrix from the product between scalar root verb (base form) and an *universal* matrix containing all verb modifier suffixes as its elements. The suffix matrix is universal in a sense that all verb forms for all tense can be generated from this matrix. Their product rules can be defined using the *rules of joining words*, the juncture rules which are known as *Sondhi* (সন্ধি).

We now discuss how Anubadok system implements this abstract notion of verb form generation. Let us consider the situation, relevant for our example sentence which has *"am reading"* as verbs. It has two verbs: "am" with lemma "be" and "reading" with lemma "read". Here verb is acting *actively.* While constructing Bengali verb Anubadok ignores auxiliary verbs such as "am" here. To construct the final verb form in Bengali, Anubadok first performs a dictionary lookup for the lemma "read" whose base form in Bengali is "পড়". Then it looks up for the element in the suffix table corresponding to *present continuous* tense, *first* person and *active* verb. The corresponding suffix element in Bengali is "ছি". Other elements in the suffix

Table 1: Verb modifier suffix table for *first person* when verb is active

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

table for such situations are given in the Table 1. Having known the base form of the verb and relevant suffix, Anubadok joins them following the juncture rules. In particular, for this example final Bengali verb "পড়ছি" is generated as

$$\text{পড়} \odot \text{ছি} \ \rightarrow \text{পড়ছি} \quad .$$

For this example, the rule of joining is rather trivial. However, for many situations the joining rules can be quite involved. Generations of final verbs for all tense forms with subject being first person and verb being active, are illustrated below.

$$\begin{pmatrix} \text{পড়ি} & \text{পড়ছি} & \text{পড়েছি} & \text{পড়ছি} \\ \text{পড়েছিলাম} & \text{পড়ছিলাম} & \text{পড়েছিলাম} & \text{পড়ছিলাম} \\ \text{পড়ব} & \text{পড়ব} & \text{পড়ব} & \text{পড়ব} \end{pmatrix} = \text{পড়} \odot \begin{pmatrix} \text{ি} & \text{ছি} & \text{েছি} & \text{ছি} \\ \text{েছিলাম} & \text{ছিলাম} & \text{েছিলাম} & \text{ছিলাম} \\ \text{ব} & \text{ব} & \text{ব} & \text{ব} \end{pmatrix}$$

Above verb form generating equation can be abstractly written as

$$\text{Final verb forms matrix} = \text{Root verb form} \odot \text{Universal suffix matrix} \ ,$$

where a part of "Universal suffix matrix", relevant for *first person*, is illustrated in the Table 1. This aptly demonstrates the structural advantages for machine synthesis of Bengali sentences. These underlying structures owes to the fact that Bengali derives its origin from *Sanskrit*.

Construction of final Bengali sentence

Having constructed Bengali subject, object and verb separately, Anubadok joins them together to form the final Bengali sentence in the S-O-V order. At the end punctuation marks are also translated. For the example sentence, "আমি" (I), the subject (S), "একটি বই" (a book), the object (O) and "পড়ছি" (am reading), the verb (V) are joined together to form the final Bengali sentence "আমি একটি বই পড়ছি ।".

Post-processing of translated documents

The post-processing stage 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. In particular, for an XML document, tags and attributes those were protected during translation stages, are restored into the standard forms at the end of this stage.

Implementation in Perl

In this section, we consider a simple Perl program which can be used to translate an XML document using Anubadok. This program also 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>))));
```

The first line of the the Perl script contains `#!/usr/bin/perl` which specifies how to execute the programming instructions contained in the file. In particular, it specifies where to find the Perl interpreter in the machine. In this example program, Perl is located at `/usr/bin/perl`. However, it does not have to be in the same location for every machine. Next few lines of the program begin with the Perl directive `"use"`. It tells Perl interpreter to load the mentioned Perl module. To translate an XML document using Anubadok, one needs to load at least three Anubadok modules which are `Anubadok::XMLPP`, `Anubadok::PoSTagger` and `Anubadok::Translator`.

As mentioned earlier, translation of an XML document in Anubadok system is performed in four steps. First step is performed by calling the subroutine `XMLPP::xml_pre_processor()`. In the example Perl program here, this subroutine reads from *standard input* `STDIN`. The output of this subroutine is then passed as input of the next subroutine `PoSTagger::penn_treebank_tagger()`. This subroutine performs the parts of speech tagging of pre-processed documents. The tagged output is then passed to the subroutine `Translator::translate_in_bengali()`. Output of translation step is then post-processed by subroutine `XMLPP::xml_post_processor()`. Final translated output is then printed to *standard output* `STDOUT`.

References:

Anubadok website

http://anubadok.sourceforge.net/

GPoSTTL website

http://gposttl.sourceforge.net/

Brill Tagger

http://en.wikipedia.org/wiki/Brill_tagger

TreeTagger

http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

Penn Treebank Project

http://www.cis.upenn.edu/~treebank/

Ankur English to Bengali dictionary

http://www.bengalinux.org/english-to-bengali-dictionary/