
# Designing a Machine Assisted Translational Tool

| | | | |
|---|---|---|---|
| Country Component | Sri Lanka | Report no. | |
| Phase no. | | Report Ref no. | |

**Prepared By** : J.U.Liyanapathirana **Designation:** Research Assistant

**Project Leader:** Dr.Ruvan Weerasinghe **Signature:** _____

**Date:** 06-11-2008

# TABLE OF CONTENTS

# Designing a Machine Assisted Translational Tool

Jeevanthi Liyanapathirana
Language Technology Research Laboratory,
University of Colombo School of Computing,
35, Reid Avenue,
Colombo 07,
Sri Lanka.
juliyanapathirana@yahoo.com

## Abstract

It is evident that a large number of languages are being used all over the world, in different countries as well as between different races in the same country. Hence, it is equally evident that confusion in communication arises between different communities. In contrast, a huge amount of resources do exist in different languages, which need to be shared among knowledge seeking people. Hence rose the need of translating incomprehensible text to a comprehensible language, which also yielded the fact that it is not feasible to have human translators on the fly to meet the needed requirements.

This observation paved way to the idea of translating documents in a computerized manner, and has expanded its progress to a vast extent. Some ideas are built upon the notion of translating the documents completely without human aid. The other type is built for helping the human translator for his translation purposes. The latter idea is the base for building a translation memory, of which the building process is discussed in this report.

# 1. Introduction to Translation Memory

## 1.1 Difference between MT (Machine Translation) and TM (Translation memory)

During the idea of Machine Translation, what happens is that the whole document would be translated without the intervention of the human translator, leaving the task of evaluating the output to the human. Whereas, in the Translation Memory, the human translator would be receiving an aid to the translation process with computerized suggestions to the document to be translated.

## 1.2 Objective

The purpose of this project is to develop a translation memory which would facilitate the translation of documents from English to Sinhala. Along with the continuous development of the memory, the translation suggestion for a given input English file would be suggested segment wise, which allows the translator to retrieve and accept the suggestion and use it as well as improving the translation suggestions. The next step would be using the continuously updating translation memory to translate a given input to get a translated output.

## 1.3 Main Process of a Translation Memory

The main steps of a translation memory can be stated as follows:

1. The user provides an input file to be translated.
2. The contents of the file are extracted as the text content which needs to be translated. At the same time, the layout of the file format (skeleton of the file) would be stored separately for further use.
3. The extracted content would be segmented (as sentences or as paragraphs) using the language specific rules. These would be the input segments which would be considered one by one during the translation process.
4. Each segment would be presented to the user. Basically for each segment, the following steps take place:
    - The translation memory database is searched for a previous translation for the provided segment. Either an exact or an approximate translation would be searched, which, if present, would be suggested as translation suggestions to the user. The user can agree to the suggestion or modify the suggestion, which in addition would be stored in the TM database for future purposes.

    - If a matching translation is not found, the entry that the user makes is entered to the database as the translation for the given segment.

5. At the end of the segment wise analysis, the TM database is updated with new as well as modified English text segments as well as their parallel Sinhala Translation segment. These new entries would then be available for future use of the Translation Memory.

Optionally, the translation segments for the given input can be combined with the skeleton to generate the translated document in the original layout format.

# 2. Designing a Translation Memory

## 2.1 Components of a Translation Memory

The basic components needed to identify the process of translation memory are identified as:

### 2.1.1 Content Extractor

The text should be stored separately for further processing. The standard way to store the content is known as xliff standard, which is an accepted standard to store data in translation memory.

### 2.1.2 Layout Format Extractor

The layout of the document to be translated would be stored separately, which is known as the skeleton (.skl).

### 2.1.3 Segmenter

This splits the text to sentences according to language specified. The rules would be stored in SRX format, which is a standard for storing the rules for segmentation.

### 2.1.4 Translation Suggestor

This suggests approximate or exact suggestions to the input segments.

### 2.1.5 TM database

This stores the parallel English and Sinhala translation segments separately. This would be a file-like database which conforms to the .tmx standard. This standard is the known format to store translation memory data.

## 2.2 Possible Solutions existing up to now

A survey conducted regarding existing translation memory tools yielded to furthermore analysis of two translation memory tools:

### 2.2.1 Transolution

Transolution is a translation memory tool which is multiplatform. Developed in python, it provides the Unicode support, and there is no language limitation. It supports the xliff files to store the data and also the .tmx file format to store the translation database in standard format.

### 2.2.2 OmegaT

OmegaT can be identified as a translation memory tool suite. Developed in Java, it provides the translation memory functionalities in a standardized and modularized manner. The tool provides a vast range of document formats to be able to be used as translation memory entries.

## 2.3 Towards Choosing OmegaT

After an analysis, these points were identified as important facts regarding using these tools as aid to build the translation memory from English to Sinhala.
Transolution proved out to be functioning well, but however its functionalities proved out to be providing fewer amounts of functionalities. Both support the standardized document formats.

In addition, Transolution , being developed in python, provides proper rendering of Sinhala text, where as OmegaT, being developed in java, the swing interface has difficulties in rendering the Sinhala text properly.

However, OmegaT is far more advanced in its principles: proper modularization, Adherence to standards properly. Some of the vast amounts of functionalities provided in standard format are Content Filtering, Skeleton and Content Storage, Rule Storage using proper standards and the most important one is the proper matching of translation suggestions: exact or approximate matching. This last functionality can be manipulated in such a way that the user can choose the extent of percentage a suggested translation should match. The methods found in OmegaT for these tasks were found out to be more convincing than those found in Transolution for the same tasks.

Because of these advantages, OmegaT was chosen as the tool to be analyzed. However, two main issues arise:

- Sinhala Rendering does not work in OmegaT swing interface properly.
- It would be preferred to have the translation memory as a service, thereby paying more interest to a client-server type system.

### 2.4 Conclusion

Considering these two facts, the final conclusion regarding building the English-Sinhala Translation Memory was built:

- A web based Translation Memory solution would be built. JSP would be used as the development technology. It is expected to solve both problems mentioned above: Sinhala Language would be properly rendered and the outcome would be a service which can be accessible to anybody needed.
- The functionalities of OmegaT would be analyzed in depth and would be extended as needed in building the main components of building the Translation Memory Process.

## 3. Translation Memory Design

The main two parts of the project are:

### 3.1 Building the Translation Memory

In order to make an input English text translated, the system should have a translation memory database of considerable quality and content. The first step is using collected parallel text to build up a translation memory of satisfactory quality

### 3.2 Getting Suggested Translations for an input file while updating the Translation Memory

Once a TM database is built, input English text can be fed to get a translated Sinhala output in the original document format with the use of suggested translations from the TM database. The user is free to accept those translations or modify them, from which gradual updating and improvement of the TM database would take place. The second step is making the translation application while improving the TM database.

The design for these two main components can be seen below:

- **Building the Translation Memory**

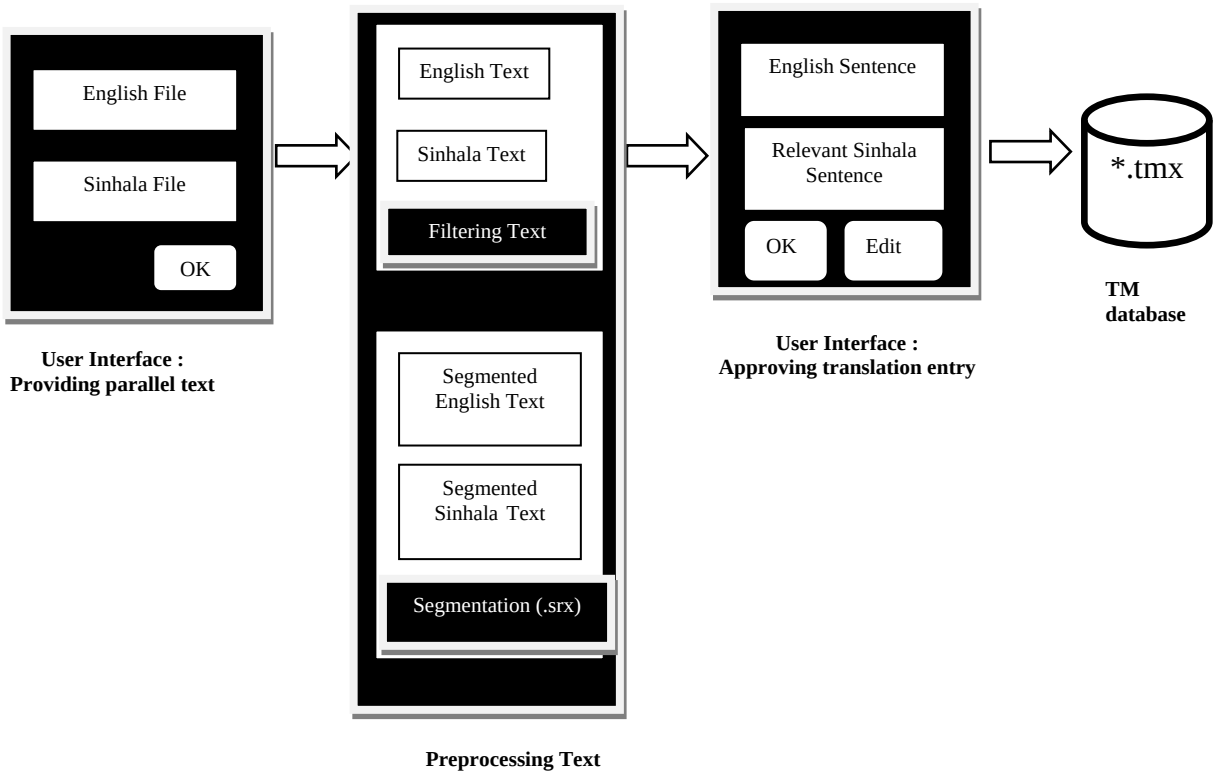


**Figure 01: Building a Translation Memory**

The procedure of building the translation memory shown in Figure 01 is as follows:

1. The user loads the parallel text to the interface.
2. Text Processing takes place in two steps:

    - Filtering the text: The document format would be identified and necessary steps would be taken to extract the text with the unnecessary content layout being ignored.

    - Segmenting : The text would be split using the language specific rules for sentence breaking. The rules would be in SRX standard.

Once segmented, the English sentences would be presented to the user with the segmented sentence assumed as the Sinhala Translation for the user.

3. Following approval or editing of the text, the translation would be stored in the database files which comply with the TMX standard.

Once the translation memory is built to a satisfactory level, the system can be presented to the user to improve it while using it, so as to providing a system serving to the user while iteratively developing the translation memory.

- **Getting suggested translations for an input file while updating the Translation Memory**

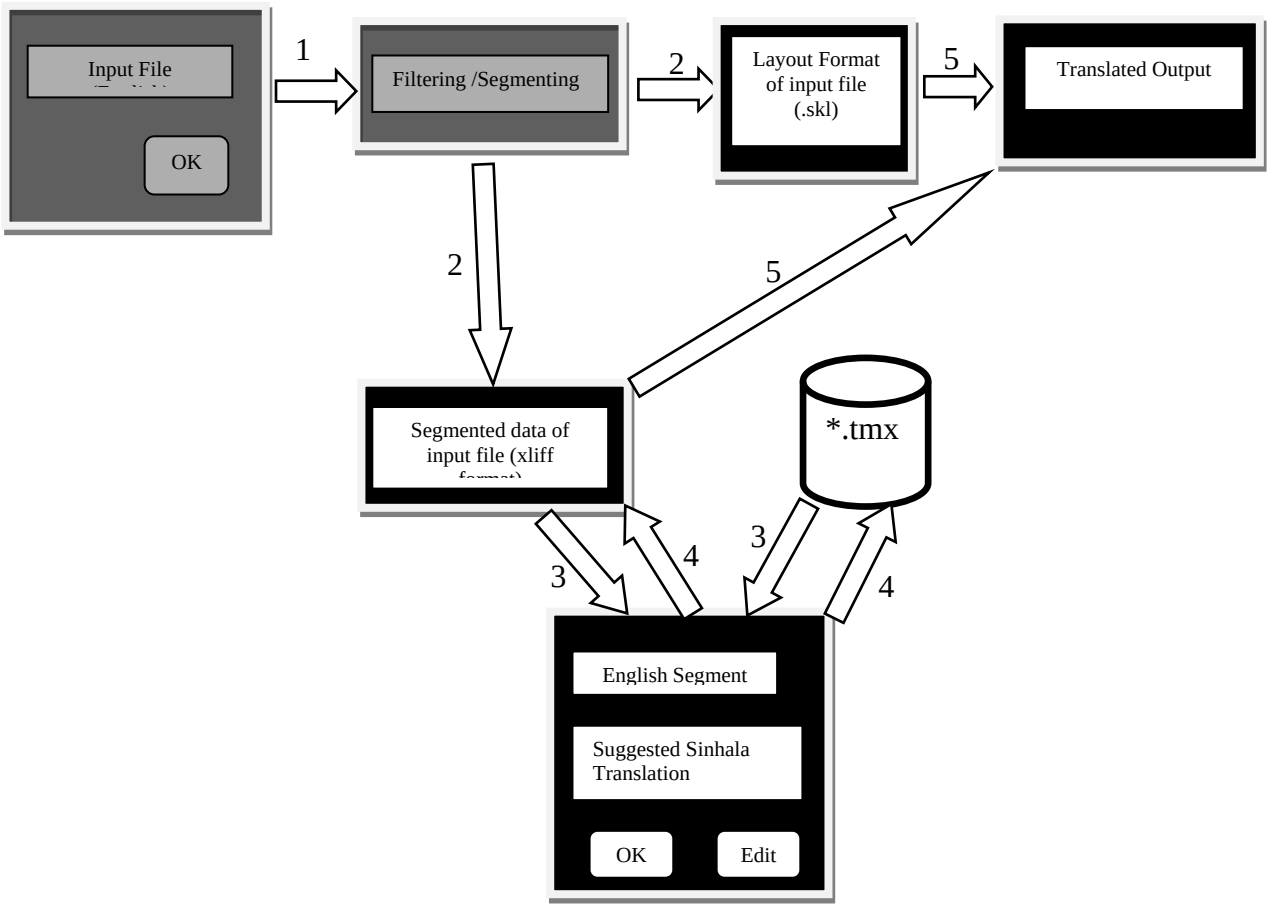


**Figure 02: Suggesting translations while updating the Translation Memory**

The design shown above in Figure 02 can be explained as follows:

1. The user provides the English input file (.html, .txt, etc.) to the system.
2. The processing of the input file would be divided into two modules:

Filtering and Segmenting: The file format of the input file would be analyzed to extract the text according to the file format. The text would be segmented along to the language specific rules.

The text segments would be stored in standard xliff documents and the layout format of the original input document (skeleton) would be stored in a separate .skl format.

3. Once the xliff document is generated, the text segments would be considered one by one with the existing translation memory database. An exact or approximate translation match would be searched from the TM database. If found any, the user would be demonstrated a suggestion about that translation.
4. The user then approves or edits the suggested translation, which enters to the TM database as a new entry or refined translation for an existing source segment (if the user has modified the existing meaning).The new translation enters to the xliff document also as the translation to the relevant input text segment.
5. To get the translated document in the original document format, the skeleton file and the newly provided translation entries would be integrated together to get an translated file of the proper format.(This can be stated as an optional extension to the translation memory building process )

# 4. Summary

To conclude with, this report can be stated as the attempt of designing a potential design to build up a machine assisted translational tool. This design was proposed after analyzing the major components needed to be available in a translation memory, the possible solutions which exist and the issues we have to consider about. This proposed design is expected to provide solutions for all the requirements we need when developing a machine assisted translational tool.