Plugin for Instantaneous Web Page Rejuvenation and Translation

Shashi Pal Singh[1(✉)], Ajai Kumar[1], Hemant Darbari[1], and Nikita Maheshwari[2]

[1] AAI, Center for Development of Advanced Computing, Pune, India
shashipalsingh@gmail.com, {ajai,Darbari}@cdac.in
[2] Banasthali Vidyapith, Banasthali, India
nikita.mundra143@gmail.com

Abstract. This paper outlines the Plugin for various browsers for instant rejuvenation and translation of web pages into Indian languages. The websites with Hindi content are less than 0.1% of the total websites, similarly less than .01% for other Indian languages. While English content are more than 55% [13]. It is high time for realization to provide the information to the local uses into their local languages so, that they can take the advantage of various resource available on websites which would result in enhancement in communication and knowledge. The Plugin tool can be plugged in various browsers. On single clicks, the whole page gets translated and rebuild into the original format without losing any information and graphics, which makes it easy, convenient, and readable for the users. The methodologies used in our system are Extraction, Rebuilding and Translation Memory. Further, we will discuss the workflow among the processes and then concluded with experimental results that are obtained with this tool.

Keywords: Extraction · English to Hindi · Plugin · Rebuilding · Translation · Translation memory (TM)

1 Introduction

Translation is in existence since human wants to communicate with each other in their own language. At initial stage, the translator was a human being itself who knew both source and target language. After sometime, people start thinking to handle this problem with the help of computer based translators. Although, it was not as much efficient as a human being but work is still in progress to do so. The approaches like Statistical Machine Translation (SMT), Rule Based, Example Based, Dictionary Based etc. have been introduced for translation but each has its own advantages and disadvantages. Here, we are going to introduce a good and efficient approach for translation which is Translation Memory (TM). Translation Memory has evolved as an important area in the translation industry. It is widely used among translators but not much of the work has been done for Indian languages in comparison to other foreign languages. So, the main focus is to translate the English text into its corresponding Hindi text or Indian Languages.

A. Unal et al. (Eds.): SmartCom 2016, CCIS 628, pp. 77–87, 2016.
DOI: 10.1007/978-981-10-3433-6_10

There are many translators that exist on web which can translate a web page. For example- Systran, Prompt, GTS Website Translator, Lucy KWIK Translator, Word lingo, Bing, and Google etc. But each translator has one or the other problem in it.

It is important to note, while translating a webpage, that the text content is not only the thing which is needed to be translated into target language. A webpage contains a lot more to be maintained like graphic, images, videos and other dynamic activities. If a translated page will look like same as it was originally then it will be more comfortable to read from the perspective of user. So, our concern is not only the translation but also the correct rebuilding of the webpage.

The complete process of extraction, rebuilding and translation is initiated by the extension. There is no specific reason to choose a particular web browser. The extension can be developed for any web browser. In this paper, we will discuss about web browser based extension, Extraction process, Rebuilding and Translation of the webpages. Further, the process flow among all these modules will be discussed.

2 Literature Review

2.1 Extension Plugin

The Extension is a Plugin, which is placed near the address bar and looks like a button. This extension can be made for any web browser for example- Chrome, Mozilla Firefox, Internet explorer, Safari etc. When this plugin button is clicked, the complete web page will be translated. There are some files which are related to make web browser extension plugin. Some of them are optional and some of them are compulsory. The files related to it are explained below [3, 4].

2.2 Manifest.json

The Manifest.json file is a very important file for making browser extension. This file tells web browser the important information about the extension like, the name of your extension, what kind of permission is needed, the icon image of the extension and the other files related to it etc. Every extension has a JSON-formatted manifest file. The name should be manifest.json otherwise it would not be recognized. There are many fields in this file. All these fields are not compulsory to be mentioned. It totally depends on the type of extension or on the application demand which you are developing.

2.3 Permission Block

Permission helps to limit the damage and protect by malware. Each permission can be either one of the list of known string example- http://www.google.com [3] or can be a match pattern example- “http://*/*”, “https://*/*” that gives access to one or most hosts. For giving permission to each and every host you can mention “<all_urls >” in place of any known string or pattern. There are many field of permission. Some of them are given below-

```
"permissions": [
"tabs", "<all_urls>","http://*/*","https://*/*","background","notifications",”downloads”,”history”,”location”,”active_tabs”  ],
```

2.4 Browser Action Block

There are two types of actions when we make a browser extension –

1. Page Action
2. Browser Action

Page Action is not preferred when you want to make your extension to work for each and every page that your browser visits. For this browser action will be used. For Example- The RSS icon in the following screenshot represents a page action that allow you to subscribe the RSS feed for the current page [3, 4] (Fig. 1).

Fig. 1. Page action

In the following figure, the T-shaped icon, right of the address bar, is the icon for a browser action (Fig. 2).

Fig. 2. Browser action

```
{  "name": " ",
  ...
 "browser_action": {
  "default_icon": {                  // optional
 },
        "default_title": " ",      // optional; shown in tooltip
        "default_popup": " "        // optional
      },       ...      }
```

2.5 Content Script

The content script [3, 4] is a JavaScript file that runs in the context of webpages. This means that the content script can interact with web pages that are currently open in the browser. JQuery is not necessarily required, but it makes the things easier.

```
"content_scripts": [  {
   "matches": [ "<all_urls>"    ],
   "js": [" ", " "],
  “css”: [“ ”]    }    ]
```

The ‘matches’ field tells Browser to inject Content.js file in every page open in the browser. If you want to inject this script to only some pages, we can use match patterns like-

[“https://mail.google.com/*”] and [http://*/*] will match any http URL, but no other scheme like https sites.

A content script can access the current page, and is limited in the APIs it can access. For example, it can’t listen the clicks on the browser action. So, a different type of script is needed to add to our extension that is ‘background script’, which has access to every Browser API but cannot access the current page.

So, the URL of the current page can be pull by content script, but this URL is needed to hand over, to the background script, to do something useful with it. In order to communicate, ‘message passing’ is done between background.js and content.js, which allows scripts to send and listen for messages. In this way content scripts and background scripts interact with each other.

2.6 Background Script

As we mentioned about the background script [3, 4] above in content script block, it is clear that it is an important part of extension and manifest file. The Browser API functions can only be used by background.js to listen the click on the browser action, so we’ll have to add some more message passing since background.js can open the tab on browser, but can’t grab the URL given in address bar. So for this, message passing is done between content.js and background.js. The Content.js file will grab the URL and pass it to Background.js for the further process.

```
{   "name": "My extension",
 …
"background": {
   "scripts": [" "]
   "page": " "
 },  ...}}
```

2.7 Translation Memory

Translation memory (TM) systems were available in the market in late 1990’s commercially but the researches in this field have been in continuation since 1970’s. There is a database that consists of existing translations which can be reused as a suggestion when translation is done.

Translation Memory (TM) technology belongs to CAT systems. It is used to providing a good precision translation. Basically, it is a database application that keeps record of previously translated units and reuse the existing if it is being repeated in future translations instead of translating the sentence from the scratch. It was believed that for

translating monotonous type of texts, the translation memory could be better utilized. But other features incorporated make it useful for non-monotonous texts also.

> "A multilingual text archive containing (segmented, aligned, parsed and classified) multilingual texts, allowing storage and retrieval of aligned multilingual text segments against various search conditions."

There are some benefits of Translation Memory which are-Consistency, Speed, Portability, Cost, Content Management. There are some limitations as well-Error can be propagated if misused and a memory is only as good as the maintenance it gets.

2.8 Jsoup Parser

Jsoup [10] is a Java HTML parser. It is a library for working with real-world HTML. It gives us a very convenient API which can extract and manipulate the data, using the best of DOM, CSS, and jQuery-like methods. Jsoup implements HTML5 specification, and parses HTML to the same DOM as modern browsers do.

2.9 Webpage Extraction

The source code of the web page can be downloaded via different-different methods. First method is via URL library of java and second method is via Parser. Here, we are taking Jsoup parser for parsing the HTML but we can't use it for downloading the source code of the webpage because of the drawback of Jsoup. Jsoup can't download the complete source code of the webpage. Sometimes, it leaves some tags or attributes of the tags at the time of downloading.

3 Proposed System

3.1 Brief Overview

The interface of our tool is an extension and a web browser. When, the user, who wants to translate the webpage, clicks on the extension button, and the source code of the webpage will be extracted. Each and Every tag that is present in the source code is parsed with the help of parser. The path of all the files (where these files are actually stored) present on that webpage is checked and modified according to the actual location of the file on the website's server. After that the complete webpage is traversed tag by tag and the text content present on the webpage is fetched from it. Then, these text strings are sent to the TM System for translation. Translation Memory is basically a database also called as TM base or Translation Memory database. TM base contains a bilingual pair of source and target language. Here the source language is English and Target Language is Hindi. There is also a concept of Term Base which is maintained to provide the translation of non-translatable English words (Fig. 3).

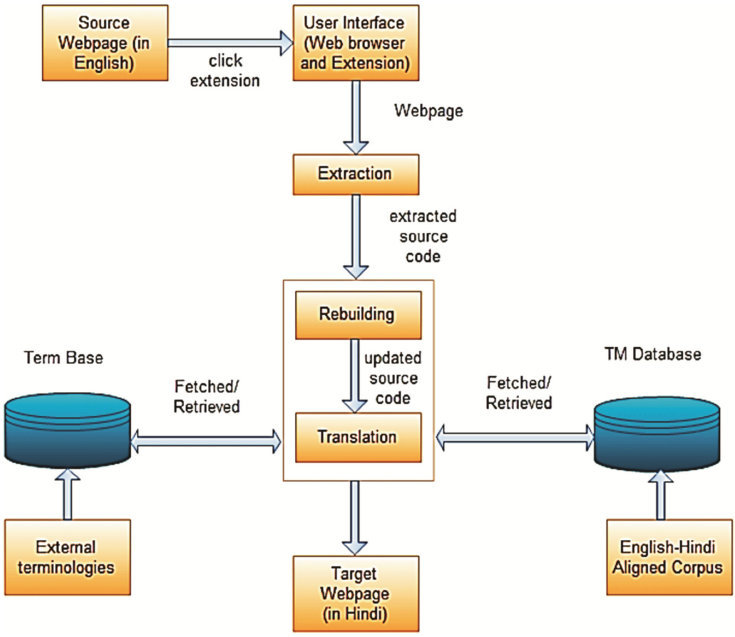

Fig. 3. System architecture

3.2 Extension

A web browser extension is a button near by the address bar of the web browser. It can be inside the address bar or outside the address bar depending on its action i.e. page action or browser action respectively. Page action is used for a particular webpage and browser action is used for many pages simultaneously open in the web browser. Here, we are opting 'browser action' to make this extension working for every webpage simultaneously open in web browser.

There are three basic files which are needed to develop an extension. These files are-Manifest.json, Background.js, and Content.js. Background.js file is needed because the click on the extension can only be listened by this file as it allows the web browser API's for example, chrome.* API in case of using Chrome browser while content.js can pull the URL of the webpage.

3.3 Extraction

When the user clicks on the extension button, the source code of the webpage is extracted by connecting with the URL of that webpage. There are some website which runs on https protocol which mandates that the communication can only be done via information provided in the certificate of that website then we need to download that certificate and import that certificate in the trustore, Default trustore

java uses can be found in \Java\jdk1.6.0_29\jre\lib\security\cacerts, then if we retry to connect to the URL connection would be accepted.

3.4 Rebuilding

After the extraction phase, the source code of the webpage is parsed with the help of parser. Here, we are taking Jsoup parser (Version 1.8.3) [13] because Jsoup is better than other parser in web scraping. It can parse and clean the Html code like other parser but it can update the Html code as well.

The tags, which contains the path of any file (images, script, stylesheet, audio, video etc.), can be image tag, script tag, source tag, link tag, object tag, input tag, td tag, body tag, table tag, embed tag, meta tag, iframe tag, frame tag etc. These tags are required to be parsed. The path of the file, given in the source code, is updated, according to the absolute path of that file on the website's server, on which it actually exists. Further, the script tag, iframe tag and frame tag may contain the files which can itself have images, videos or any of the tag that are mentioned above. These are also needed to be handled in the same way as depicted above. The webpage can also contain some background images, imported CSS or scripting code which can contain images and other files. The parser can't parse these patterns. So, we need to make separate regular expression for each condition to detect these files and update their path. The path of the files is required to be updated because the tool is running on our server and the files are not present on our server.

Algorithm for Rebuilding:-

Input: - Click on the extension and **Output**: - Updated source code

Steps: -

1. Click on the extension.
2. Source code of the webpage will be downloaded.
3. Parse all the tags which have path of any file (ex- Images, scripts, stylesheets, video, audio).
4. Extract the source path of that file mentioned in one of the Tag's attribute.
5. Check whether we need to change the path or not.
6. If yes, then check the correct location of the file exist on the server, change the path according to that and put it back to the same position from where it was retrieved.
7. Make the regular expression for those Patterns which can't be parsed with the parser.
8. If the pattern exists then go to step 3 to 6.

The rebuilding of the webpage is a very important phase in webpage translation. It helps to maintain the webpage as it was before translation. A webpage may contain images, audio, video, stylesheets which gives it proper look and other dynamic activities. A user will feel comfortable in reading the webpage when it is in proper and managed format. So, only text based translation is not only the thing to be focused. Rebuilding of the webpage has to be focused too.

3.5 Translation

The complete source code of the web page is parsed on the basis of its opening and closing tags and the string between these tags are fetched and sent to the TM System to check its corresponding Hindi translation.

Algorithm for getting the text of the webpage-

Input: - Updated source code and **Output**: - Source Language String for translation.
Steps: -

1. Parse the source code on the basis of opening and closing tag.
2. Check whether the text exists between the last closing tag and next opening tag.
3. If yes, check whether the last tag was script tag or style tag.
4. If it was script tag or style tag then don't send that text string for translation.
5. Else send it for translation.

Algorithm for translation-

Input: - Source Language String and **Output**: - Translated String in Target Language
Steps: -

1. The text string is preprocessed. In which the text filtering and segmentation is done.
2. In segmentation process, if any delimiter is present in the string, which is sent for the translation, then split the string, on the basis of delimiters like ".", "?", "-", ":", ";", "!",
3. The hash code of each string is generated.
4. The generated hash code is matched with hash code stored in TM database.
5. If matched, return its corresponding Hindi
6. Otherwise, if number of tokens in the segment is <=7 then compute 2-grams, else if number of tokens in the segment is > 7 <=10 then compute 3-grams, else compute 4-grams. Each N-gram obtained is treated as a search token to fetch TM segments which contains that N-gram token so as to get useful results only by fetching hardly 100 to 1000 segments and reduce searching time [6].
7. Now that source segment that contains 50% unique tokens of query segment is only considered for matching thus reducing search space.
8. Now among them, names (using Named Entity Recognition), gender cases and other placeables like numbers, dates etc. are handled [8].
9. If considering B1 as the N-gram of the query segment and B2 as the N-gram of the TM source segment, apply Levenshtein algorithm.
10. If the value returned by the algorithm is either 0 or 1 then consider that N-gram in score computation else discard it. Increases the M (a variable) by 1 for each N-gram added.

Mathematically, the Levenshtein distance [11] between two strings a (Source string), b (Target String) is given by $Lev_{a,\,b}$ (i, j), which is the distance between the first "i" characters of "a" and the first "j" characters of "b".

If (a_i ! = b_j) then Cost function = 1
If (a_i == b_j) then Cost Function = 0
If min (i, j) = 0,

$$\mathrm{Lev}_{a,b}(i,\ j) = \max(i,\ j) \tag{1}$$

Otherwise,

$$\mathrm{Lev}_{a,b}(i,\ j) = \min\left\{\mathrm{Lev}_{a,b}(i-1,\ j) + 1,\ \mathrm{Lev}_{a,b}(i,\ j-1) + 1,\ \mathrm{Lev}_{a,b}(i-1,\ j-1) + \mathrm{cost}\right\} \tag{2}$$

The advantage of Translation Memory is that it gives faster translation results comparative to other approaches. So the time taken in webpage translation is comparatively less and user doesn't need to wait more.

4 Result

There are so many tools for webpage translation like- Systran, Prompt, GTS Website Translator, Lucy KWIK Translator, Word lingo, Bing, and Google etc. But each translator has one or the other problem in it. Some can't translate in Indian Languages, some others cannot translate web pages based on https:// protocol and some don't rebuild the webpage correctly etc. So, this extension tool is made to overcome with some of these problems.

Total 155 webpages were tested by this extension tool. The result is only based on "How good rebuilding of a webpage can be done by this tool" The results are not calculated on the basis of quality of the translation. The webpages based on framework such as Joomla, WordPress etc. was taken. The webpages, either developed in HTML or HTML5 format, are considered too and it can also translate the https:// protocol based web page etc.

The following graph will show the results of this tool -

Testing is done using this tool and the results are obtained manually with human intervene for checking the result of rebuilding. Five parameters are decided to rank the rebuilding of the webpage. These parameters are- Excellent, Very Good, Average, Below Average and Poor. The webpage is put under the 'Excellent' category if there is no fault in the rebuilding of the webpage. If there are one or two images/video/audio missing then it is put in 'Very Good' category. If, more than two image/video/audio are missing or the content is not properly managed then it is put under 'Average' category. If almost nothing is present on the webpage after rebuilding or if the output is coming totally haphazard then it is put under 'Below Average' category and if the tool is unable

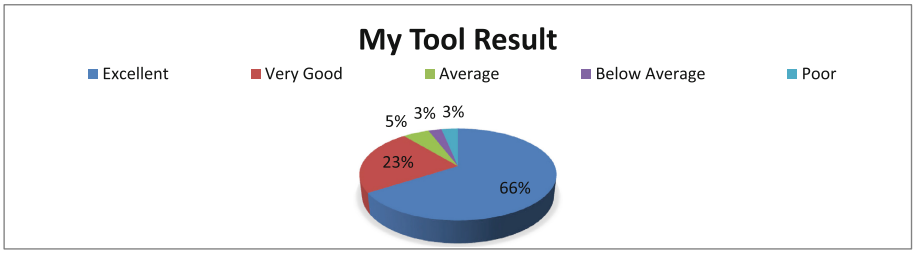

Fig. 4. TM translator tool result

to rebuild the webpage or giving nothing after rebuilding then it is put in 'Poor' category (Fig. 4).

On comparing the processing speed of other translation tools and this tool, we will see that this tool is little slow than other translation tools and sometimes, it also takes time to load the webpage which have heavy graphics.

5 Conclusion and Future Scope

Not much work has been done in the area of rebuilding and translation of a webpage for Indian Languages. So, we have developed an extension for performing this task which will extract the source code of the webpage, rebuild it and translate it. Jsoup parser is helping to parse the tags present in the source code and the advantage of using this parser has already been discussed. Total 155 webpages are tested by this extension tool and other translators. The final results of our tool are comparatively better. 66% of webpages are in 'Excellent' category when they are rebuilt with this tool, 23% are in 'Very Good' category, 5% are in 'Average' category, 2% are in 'Below Average' category and 2% are in 'Poor' category.

In Future, the enhancement can be done by finding out other different tags with their various attributes. And if they are necessary for helping in good rebuilding of webpage then they must be handled in the rebuilding process. The other thing which can be improved is the processing speed of rebuilding process which is slow in comparison of other translator. Also, the functionality of this tool can be extended to perform translation in other languages as well.

References

1. Srivastava, N., Singh, P., Chauhan, S., Singh, S.P., Kumar, A., Darbari, H.: Hindi-English translation memory systems. Int. J. Emerg. Trends Technol. Comput. Sci. (IJETTCS) (2014). AAI, Center for development of Advanced Computing, Pune, India
2. Joshi, N., Mathur, I.: Design of English-Hindi Translation Memory for Efficient Translation, Department of Computer Science, Banasthali Vidyapith University (2012)
3. https://developer.chrome.com/extensions/getstarted.html turorial by Google Chrome. Accessed 23 Jan 2015
4. Berke-Williams, G.: A developer in San Francisco. https://robots.thoughtbot.com/how-to-make-a-chrome-extension. Accessed 23 Jan 2015
5. Somers*, H., Diaz**, G.F.: (UMIST, Manchester)* (Universidad de Sevilla)**, Translation Memory vs. Example-based MT – What's the difference? (2004)
6. Wołkowicz, J., Kulka, Z., Warsaw, V.K.: n-Gram-Based Approach to Composer Recognition, University of Technology Institute of Radioelectronic Nowowiejska 15/19, 00-665 Warszawa, Poland Dalhousie University Faculty of Computer Science, Canada (2008)
7. McTait*, K., Olohan**, M., Trujillo*, A.: A Building Blocks Approach to Translation Memory Centre for Computational Linguistics*, Centre for Translation Studies** Department of Language Engineering UMIST Manchester M60 1QD (1999)
8. Saha, S.K., Ghosh, P., Sarkar, S., Mitra.P.: Named Entity Recognition in Hindi using Maximum Entropy and Transliteration, Indian Institute of Technology, Kharagpur (2008)

9. Arthern, P.J.: Machine Translation and Computerized Terminology Systems a Translator's Viewpoint, Head of English Translation Division, Council of the European Communities, Brussels (1979)
10. http://jsoup.org/ tutorial by jsoup HTML parser © 2009 – 2015 Jonathan Hedley. Accessed 15 Feb 2016
11. Haldar, R., Mukhopadhyay, D.: Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach, Web Intelligence & Distributed Computing Research Lab Green Tower, C-9/1, Golf Green, Calcutta 700095, India (2011)
12. EAGLES Evaluation of Natural Language Processing System, Initial Survey on the Availability of Translation Memory Tools. Featurization: Design and function of translation memory. www.issco.unige.ch/research.projects/ewg95/node152.html
13. https://en.wikipedia.org/wiki/Languages_used_on_the_Internet. Accessed 23 Jan 2015
14. https://github.com/jhy/jsoup/. Accessed 23 Jan 2015