

Project Report

Translation Recall : A Translation Memory-based Method using Statistics in a Bilingual Corpus

Group Members

Jakarin Chanpapatpol 5322780156
Thee Sritabtim 5322780032

Prof. Dr. Thanaruk Theeramunkong

School of Information, Computer and Communication Technology,
Sirindhorn International Institute of Technology,
Thammasat University

Semester 2, Academic Year 2013

10th March 2014

Table of Contents

| | |
|--|----|
| Abstract | 1 |
| 1 Introduction..... | 2 |
| 2 Background..... | 2 |
| 3 Objectives | 3 |
| 4 Outputs and Expected Benefits | 4 |
| 4.1 Outputs | 4 |
| 4.1.1 Translation Memory Manager | 4 |
| 4.1.2 Performance result | 4 |
| 4.2 Benefits | 4 |
| 4.2.1 Short-Term benefit..... | 4 |
| 4.2.2 Long-Term benefit..... | 4 |
| 5 Literature Review | 4 |
| 5.1 XML Frame Construction | 4 |
| 5.2 Word Marking for Translation Assistance in CAT | 5 |
| 6 Methodology..... | 5 |
| 6.1 A memory- and statistical-based approach | 5 |
| 6.2 Tools..... | 6 |
| 6.2.1 GIZA++ | 6 |
| 6.2.2 Lucene..... | 6 |
| 6.2.3 MongoDB | 6 |
| 6.3 System Overview | 8 |
| 6.4 Preparing Translation Memory | 9 |
| 6.4.1 Segmentation | 9 |
| 6.4.2 Word Aligning | 9 |
| 6.4.3 Create a new translation memory | 10 |
| 6.5 Suggestion Construction | 10 |
| 6.6 Translation Process | 10 |
| 7 Project Schedule | 7 |
| 8 Technical Description | 8 |
| 8.1 System Overview | 8 |
| 8.2 Requirement and Database Design | 11 |
| 8.2.1 Requirement..... | 11 |
| 8.2.2 Database design | 11 |
| 8.2.3 User Interface and Implementation Details | 12 |
| 8.3 Evaluation | 13 |

| | | |
|-----|----------------------------------|----|
| 8.4 | Conclusion and Future work | 14 |
| 9 | References..... | 15 |
| | Appendix A: File directory | 16 |

Statement of Contribution

By submitting this document, all students in the group agree that their contribution in the project so far, including the preparation of this document, is as follows:

Jakarin Chanpapatpol 5322780156 50%

Thee Sritabtim 5322780032 50%

Abstract

Towards the formation of the ASEAN Economic Community, there is a need to develop translation of documents among languages in those AEC countries. While it is hard to create a machine translation engine that can perfectly translates a source language (Thai) to a target language (English), as a more manual method the translation memory-based method just provides a translation of an input text by listing the translations of sentences that are similar to that input text. In this work, we propose a method to combine the fully automatic machine translation approach and the translation memory-based approach. Our proposed computer-assisted translation system retrieves the translation pairs which are similar the current input and then flexibly modifies some parts in the translations using a dictionary or translations in a corpus. The system we have constructed is comprised of various tools that each has its own specialty function including GIZA++, Lucene, and MongoDB in which each tool's functions in the system are provided. We also show the overall components, process, and the techniques we use in our system.

Keywords: translation memory, computer-assisted translation, statistical machine translation, GIZA++, word alignment

1 Introduction

In present day, there are two main methods of translation that involve computer into the process; Machine Translation (MT) and Translation Memory (TM). While machine translations focus on automatic translation by computer, translation memory places translation work on human translators instead.

Translation memory is a database which stores translation units (pairs of source text segment and the corresponding translated text segment) which can later be retrieved whenever a human translator wants to translate a text segment similar to the one stored in the database. Translation memory is most efficient when the source document contains repetitions of text segments with similar patterns and structures. Typically, TMs are used in conjunction with computer-assisted translation (CAT) tools, [1] where CAT tools can provide access to translation memories through a user-friendly graphical user interface.

Using translation memory can definitely reduce translation time and workload when the translations involve multiple text segments similar to translation units stored in TM.[2] However, a human translator still has to post-edit the translation since there will be some portion of translated segments that is still not completely translated.

With the help from statistical machine translation (SMT) techniques, word alignments between input text segment and its translation portion retrieved from TM can be identified statistically which we can use them to identify the “missing parts”, the portion of input text segment that is needed to be manually translated by human translators.[3] And by using automatic dictionary lookup for the missing parts, the system can suggest the most likely adjustment the translator should perform which would provide convenience for them during post-editing.

In our project, we develop a computer-assisted translation software (software that assist in users’ translation process) called Translation Recall. It assists users by using translation memory to store previously translated text segments, which can later be retrieved using information retrieval techniques. Users will be able to import external translation memory files, which will enable users to translate more efficient for a particular domain. The translation memory will store sentence structures, in order to make the retrieval process more rapid. When the user inputs a text segment to be translated, the system will pre-process the input by performing word segmentation (in case the input language is Thai). The result of this step will be used as a query to the translation memory, where a list of translations is returned. The returned list will be ranked according to the similarity of the input to the source text segment. If necessary, the system will align the input with the translation text segments and suggest a most probable translation. The user reviews the output and, if necessary, edits the suggestion, after which the result will be inserted into the translation memory.

In this report, the background behind our project is provided in Section 2. In Section 3, the objective of this project is pointed out. The outputs and expected benefits are provided in Section 4. For Section 5, literature review of the papers we studied is provided. In Section 6, the methodology that we use in development the system is provided. The project schedule about what we have done is shown in Section 7. The technical description is discussed in Section 8. Finally, we conclude the work we have done and possible future work in Section 9.

2 Background

With the establishment of ASEAN Economic Community (AEC), the need for translation has become a basic requirement for information exchange among AEC members. We aim to develop a system that would provide the ease in translating information exchanged among them.

Although several machine translation techniques have been significantly developed with the most popular one being statistical machine translation that predicts the most likely translation for a given input text statistically, the translation from them is still not applicable with the documents that need precise translation such as official government documents. High accuracy is usually required for professional document translations; this is where human translators are needed. In order to provide error-free translation, the process needs to be operated or supervised by professional human translators. [4]

Translation memory has been developed with this concept in mind which rely more on human translated data, where the source text along with its reliable translation can be stored and retrieved for later usage. The limitation of TM is that it can only return the translation of what it store. If the human translator needs to translate something similar to the sentences stored in TM, some post-editing has to be performed to the translation or, in case of no match; the translator might have to construct the translation on their own. The situation is illustrated in Fig.1 where the first sentence in documents finds its similar match from translation memory as depicted in underlined text which we know that part of its translation could be used as reference for. However, the second sentences could not find any of matches from the translation memory, this gives user no choice but to start construct its translation from scratch. To overcome this limitation, statistical machine translation techniques can be used on the retrieved translations from the translation memory to provide a suggestion for the post-editing step [3] as well as integrating the dictionary function into the system. With the proposed solution, when the system finds the second sentence, instead of suggesting nothing to user, it could look up the dictionary and try to provide the user bit of translation across the sentence.

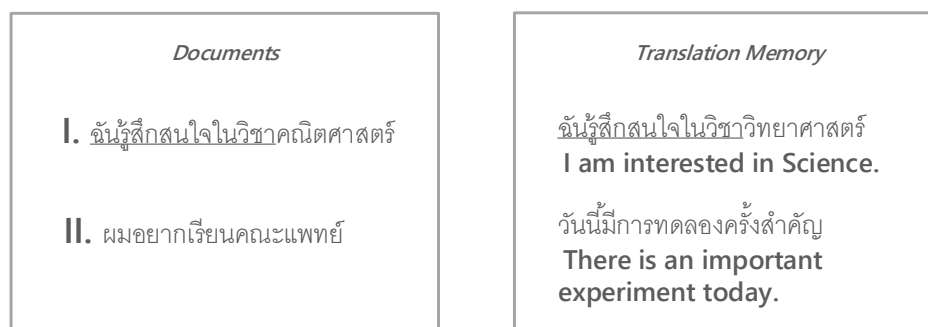


Figure 1

3 Objectives

The aim of this project is to introduce statistical techniques to enhance the Translation-Memory system by not only listing the translations of similar sentences to translators but also suggesting translations by modifying some parts of translations using a dictionary or a corpus in order to make the translation close to the target. This technique is used in machine translation. In other words, we propose a translation method that combines the advantage of machine translation and translation memory. Our four objectives are as follows.

1. To study and understand the working concepts of human translators and implement the system according to their needs.
2. To study and understand the concepts of implementation of Translation Memory System.
3. To develop the statistical natural language processing techniques that would improve the performance of the Translation Memory System.
4. To apply the Thai Language Processing Technique to the system to improve the Thai language translation work.

4 Outputs and Expected Benefits

4.1 Outputs

4.1.1 Translation Memory Manager

A computer assisted translation tool. The system will be implemented as web-based application. Also, the user interface of the program will be designed in order to make the usage become easier.

4.1.2 Performance result

The comparison in term of aiding human translators in translation work using ROUGE (Recall-Oriented Understudy for Gisting Evaluation). The comparison will be made between the translation done by human translator alone and the output of the system.

4.2 Benefits

4.2.1 Short-Term benefit

The people that gain the most benefit would be human translators as the system including the user interface will be implemented based on the needs of them, thus, aiding them as much as possible in the translation works.

4.2.2 Long-Term benefit

The program will also be made available as an open-source program so that any developer that interest in Translation Memory System could study from it or develop the even better program.

5 Literature Review

In the literature one can find several systems or methods which are related to our work. Some papers introduce a general theory of integrating translation memory systems (TM) with statistical machine translation (SMT) engine, other papers discuss practical approaches of software integrations, while some other papers focuses on word alignment techniques, which are central to our approach of integrating TM with SMT. We will discuss three relevant works which focus on different aspects related to our work.

5.1 XML Frame Construction

Koehn and Senellart (2010) suggested a concept of combining a TM system with SMT techniques. In “Convergence of translation memory and statistical machine translation”, the authors proposed two methods of combining TM with SMT approach. We will discuss only one method, which is highly related to our work and forms the basic principles of our final system.

XML Frame Construction is a method of constructing an XML file to be forwarded to the Moses SMT decoder. An XML Frame is an XML file where the portions which do not need to be translated are tagged such that the Moses SMT decoder will only translate the untagged portions. This method uses GIZA++ to train the alignment model which is necessary for XML frame construction. The process begins by retrieving the best fuzzy-matched source language segments from the TM. Then using word alignment obtained from word-based Levenshtein edit-distance of the input and source language segment in the TM along with source-target language alignment obtained from GIZA++, an XML frame is constructed according to 5 different cases of alignments the authors have described. The authors have experimented with different ways

of replacing the words in the XML Frame. The method obtains the BLEU scores between 58.5-79.9 for fuzzy matches greater than or equal to 70%

5.2 Word Marking for Translation Assistance in CAT

Esplà et al. (2011) proposed a method to improve CAT systems based on TM. By marking the words in the suggested translation as “keep” or “edit”, the system would recommend the user to only edit some portion of the translation suggestions. In contrast to other work we will discuss, this work does not aim to replace any words in the translation output, but focuses on suggesting the user to edit the necessary words.

This paper describes the technique of selecting the suitable word to keep or replace by applying similar principles as the XML frame construction suggested P.Koehn. Using word alignment between the source segment s' (input) and the source language segment s_i along with word alignment between the source language segment s_i and the target language segment t_i , the system will be able to decide which words should be left unedited and therefore marked as “keep” and which word should be modified and therefore marked as “edit”. The word alignment between s' and s_i is obtained from word-based Levenshtein edit-distance, while the alignment between s_i and t_i is obtained from alignments produced by the open source tool GIZA++ using SMT techniques (IBM Models). The authors have experimented with different criteria for indicating words as “keep” or “edit” as well as different modes of combining GIZA++ alignments. The experiment is conducted by performing Spanish to English translation, with the result of more than 94% accuracy (correctly guessed words to replace) for fuzzy-match scores greater than or equal to 60%.

Although this paper propose a method of marking the word in the target translation, with a high accuracy, the translation output of this approach will still have to be edited by the user. On the other hand, if the system also provides suggested replacements for the words marked as “edit” the user would have to spend less time editing the translation and therefore increase productivity. An important remark is that the techniques suggested in this work will be implemented in our final system; therefore we can regard this as an effective method for integrating in a CAT system.

6 Methodology

6.1 A Memory- and Statistical-based Approach

Recently, researchers have been experimented with several approaches to combine the concept of Translation Memory with Statistical Machine Translation techniques (Koehn, 2010). In Convergence of Translation Memory and Statistical Machine Translation[3], Koehn P. and Senellart J. described methods which combines both approaches and showed that combinations of the two approaches outperform both approaches for high fuzzy match ranges.

For our work we are interested in the method called the XML method, where XML markup is used specify which source words still need to be translated. Esplà et al. also proposed a method to improve computer-aided translation (CAT) systems by marking which words the user should change or keep unedited [1]. This method uses word alignments in combination with Translation Memory to decide and assists users in editing translation work.

The main idea of both approaches is to compare the input segment with a segment in the Translation Memory and detect the mismatched parts, where the system can translate the missing parts or mark the word as “to be translated”.

In our work we implement a method similar to the XML method, but instead of producing XML markup and forward it to an SMT decoder, we construct a final translation suggestions in JSON markup and forward to the web user interface to let user select and edit the suggestions.

6.2 Tools

In our system, we integrate various tools that provide the desire functionality together which are discussed in the followings.

6.2.1 GIZA++

It is a statistical machine translation toolkit that is used to train IBM Models 1-5 and an HMM word alignment model.[5] Also, it is an open source according to its GPLv2[6] license agreement. We use this tool for identify word alignments between source and target sentences of the translation pairs stored in TM which could be used in the suggestion construction process discussed in section 2E).

6.2.2 Lucene

It is a text search engine library written in Java[7]. Lucene is an open source project which user is subjected to follow its Apache License version 2.0[8] agreement. One of Lucene searching algorithm that we use involves around “vector space model”[9] where it search result and the ranking depend on the similarity in word-level between the query and the data stored in Lucene. We use Lucene for two purposes which are:

1. Provide search and recall translation pairs from TM:
2. Use one of its libraries for Thai words segmentation.

6.2.3 MongoDB

MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database[10]. While it is written in C++, it also provides the driver so that its database and functionality could be accessed from Java-based program. Although it is an open-source, it contains various license depending on its parts and application including GNU AGPLv3, Commercial License from MongoDB, Inc., Apache License v2.0, and many more[11]. Our use for MongoDB is for two purposes which are:

1. Store terminology and its respective translations from GIZA++’s output
2. Store information and data about translating documents as well as other miscellaneous information.

7 Project Schedule

| Task | Start | Duration (days) | End | Status |
|--|-----------|--------------------|-----------|----------|
| Choose Project Topic | 10 Jun 13 | 8 | 18 Jun 13 | Complete |
| Study techniques and processes | 18 Jun 13 | 14 | 2 Jul 13 | Complete |
| Prepare first practice presentation | 10 Jul 13 | 14 | 24 Jul 13 | Complete |
| Prepare second practice presentation | 26 Aug 13 | 14 | 9 Sep 13 | Complete |
| Prepare final presentation | 30 Sep 13 | 14 | 14 Oct 13 | Complete |
| Write the proposal | 30 Sep 13 | 14 | 14 Oct 13 | Complete |
| Implement a demo based on PHP+MySQL | 15 Aug 13 | 15 | 30 Aug 13 | Complete |
| Implement a demo based on Lucene | 1 Sep 13 | 15 | 16 Sep 13 | Complete |
| Test the demos and compare results | 17 Sep 13 | 7 | 24 Sep 13 | Complete |
| Design the user interface | 1 Nov 13 | 7 | 7 Nov 13 | Complete |
| Design the database | 8 Nov 13 | 7 | 14 Nov 13 | Complete |
| Implement the translation retrieval system | 15 Nov 13 | 30 | 15 Dec 13 | Complete |
| Implement user interface | 16 Dec 13 | 31 | 15 Jan 14 | Complete |
| Implement suggestion function | 16 Dec 13 | 31 | 15 Jan 14 | Complete |
| Revise and update the final report | 15 Feb 14 | 21 | 9 Mar 14 | - |
| Prepare and present the project result | 10 Mar 14 | 7 | 17 Mar 14 | - |

8 Technical Description

8.1 System Overview

Our application is to assist in translation work which, unlike normal machine translation program, aims to provide the user the suggestion of translation only which the suggestion will be made using the currently translating document and the already translated document that stored in a database.

8.1.1 System Architecture

The main target user of this application would be human translators whose work requires a lot of translating a large amount of documents. With our application in assistance, the translation work would lessen as the system will provide the suggestion which requires the user to edit some part of it which is better than constructing the whole new translation. Also, the suggestion would be made based on the most relevant documents of a set of already translated documents to ensure the best suggestion for the currently translating document. System Overview

Our system consists of five main parts as shown in fig. 2 which are: web interface, main system, MongoDB, Lucene, and GIZA++.

The web interface provides user-friendly interface where users can manage their translation work projects. The translation work will occur here where users import their translating document into the system and start their translation work through the interface provided.

The main system acts as the intermediate between web interface which interacts with users and the other components whereas it gets the input sentences from web interface and then try to retrieve translation pairs and their word alignment metadata from index created by Lucene. The metadata got from previous step will then be used to query the actual alignment information from MongoDB's database which then the main system will use them along with their respective translation pairs to construct translation suggestion for the input sentence. After that, the suggestions are sent back to display on web interface so that users can use them in translation work.

In addition to function stated in previous paragraph, the main system also act as automatic system that would communicate with MongoDB, Lucene, and GIZA++ whenever an translation project is complete, i.e. users decide to do that through web interface's provided function, or the new TM is added into the system. When one of these conditions met, the main system would signal the GIZA++ to calculate the word alignment from the newly completed project's or newly added TM's translation pairs. After that the main system will read the output of GIZA++, determine, and store them in their respective database, Lucene and MongoDB.

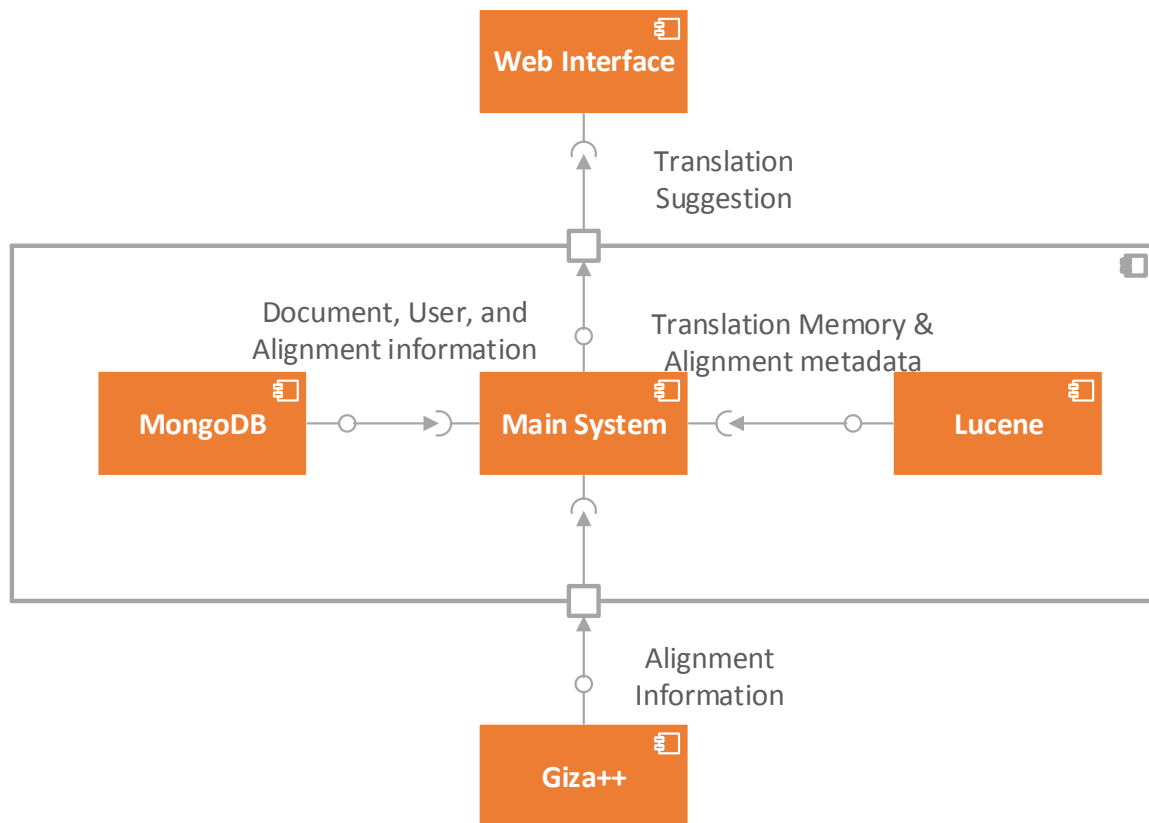


Figure 2 System Overview

8.1.2 Preparing Translation Memory

Our system, like any usual translation memory imbued system, needs some data in TM first before it could be used to its full capability. In this section, we would provide the overview of how the TM is prepared for the program first hand.

From fig. 3, we could divide the whole process into three sub processes which are:

8.1.2.1 Segmentation

In this sub process, the sentences in translation pair corpus will go through the word segmentation process which actually to denote the word boundary in the sentences which could prove useful for the word aligning process. We perform the process by using Thai Analyzer from Lucene's library. Also, this is crucial for languages without clear word boundary like Thai as well.

8.1.2.2 Word Aligning

The sub-process takes the output from previous step and send it to GIZA++ as an input which will result in many output files. However, there are two files that we are interested which the first one contains information about the word alignment between the original language text, or source, and its translated language text, or target, and the second one contains dictionary of all words in source and their possible alignment from target with their probability. The information about word alignment will be stored in Alignment Storage of MongoDB's database while the information about dictionary will be filtered and stored in Terminology Database which is provided by MongoDB also.

8.1.2.3 Create a new translation memory

In this last sub process, the system will combine the input from two sources which are the Translation pair corpus itself and Alignment reference ID from Alignment Storage. The input will then be stored in the Translation Memory Storage provided by Lucene with each translation pair will be stored with the reference ID of its corresponding alignment into the same record or document, in term of Lucene.

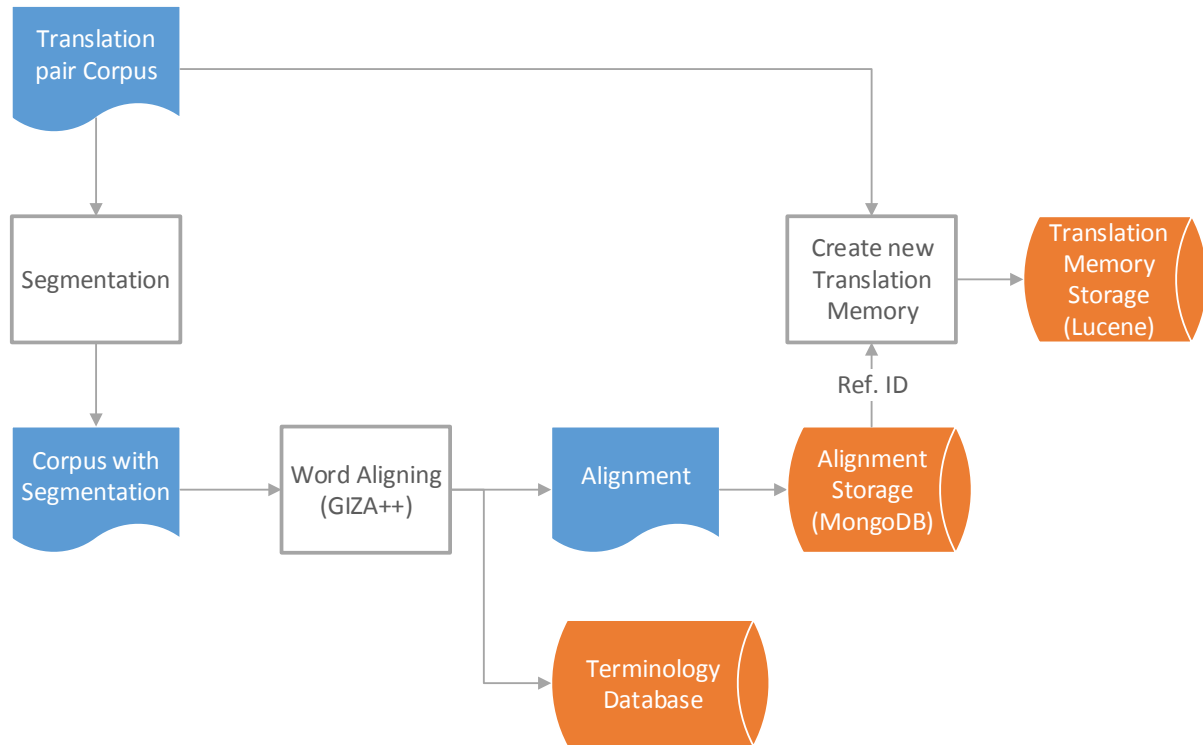


Figure 3

8.1.3 Suggestion Construction

For every source language input segment s' the system will construct a list of suggestion segments based on the segments retrieved from the translation memory which are most similar to s' . Each suggestion segment is constructed based on the XML method[3] (Koehn P., 2010), with additional term replacement step.

According to the XML Method, there are 5 possible ways alignments can occur; Normal, Insertion, Deletion, Unaligned and Multiple Alignments. In our implementation a word in an input source segment can only be aligned to at most one word in a translation memory source segment, therefore the case of Multiple Alignments will not occur.

After an intermediate suggestion comparable to the XML Frame is obtained, the system will try to replace each un-translated word from the missing parts with a list of terms, retrieved from the terminology database and ranked according to the translation probability $P(\text{source} | \text{target})$ [Giza++ documentation reference], if no match can be found in the terminology database, the un-translated word will be left unchanged.

8.1.4 Translation Process

When the user submits an input document into the system, the document will be segmented into translation units (segments), after which the user can select one of the segments to translate (1).

When a segment is selected the system will display a list of suggestion segments to the user (2), where the user can choose and modify one of the suggestions to be the translation of the input segment (3).

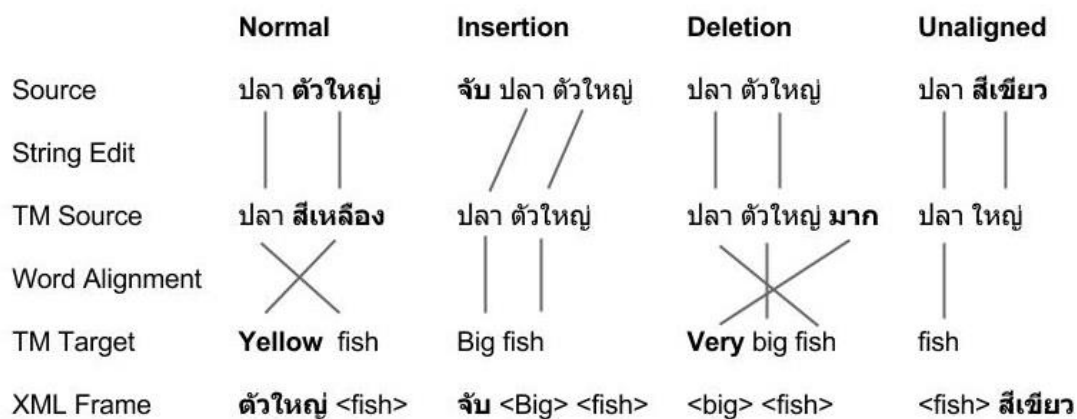


Figure 4

8.2 Requirement and Database Design

8.2.1 Requirement

We use four primary systems for our application main function which are:

1. Lucene-Java: For storing and retrieving the translated documents that would be used for providing suggestion.
2. GIZA++: For calculating the information from the translated documents before storing them into Lucene's index file. The information will be used for suggestion-related function.
3. MongoDB-Java: For storing and retrieving the data about users and their translating project.
4. JavaScript: For processing the data retrieved from both Lucene and MongoDB providing pre-editing on the suggestion, and also provide the user-friendly interactive user interface.

8.2.2 Database design

1. Lucene: The same category translated documents will be stored in the same index file, or in our technical term translation memory. Each of them will contain fields:
 - 1.1. "source": Store the segment/sentence of the source language (Thai).
 - 1.2. "target": Store the segment/sentence of the target language (English) which is the translation counterpart of the previous field.
 - 1.3. "reference": Store the reference identifier for MongoDB "TPs" collection which contains extra information for each translation pair.
2. MongoDB: The database contains four collections which are:
 - 2.1. "Users": Store the information about user account.
 - 2.2. "Documents": Store the information about translation project associated to each user in the collection Users.
 - 2.3. "Segments": Store the information about the currently translating segments/sentences which is associated with each projects in the collection Projects.
 - 2.4. "TMs": Store the information about the Lucene index or the Translation Memory available.

- 2.5. “TPs”: Store extra information of each record in Lucene database including alignment information obtained from GIZA++.

8.2.3 User Interface and Implementation Details

8.2.3.1 Users and projects management

New user can be registered into the application which will store the information into the collection “Users” and then they can start using the application by choosing to create a new project. The new project will require the user to provide the information such as project name, source and target language, the original document that will be translated, and translation memory which will be stored into collection “Projects”. After which the application will start to process the original document into JSON format of each individual sentence and store it into the collection “Segments”.

Once a project is created, it will be specific for the user who creates it. The user can then start translation work or manage information about it, which consists of editing and deleting.

8.2.3.2 Translation Assisting

1. Document translation part

The application will retrieve the JSON format that contains the information about the sentences of the original document from the collection “Segments” that is corresponding to the current project and then process it into block objects of translation segments which will provide function for user to translate them individually.

Each block’s left end will be marked with color which is green, yellow, or red. Each color refers to the current status of the translation block it is marking with green as “translated”, yellow as “pending/translating” and red as “un-translated”.

2. Suggestion part

At the right of translation segment block is another interface that would provide the suggestion of the translation for the sentence of block that is currently selected by the user.

The process is done through AJAX to another part of the application that will retrieve the suggestion from Lucene’s index or translation memory that is used for the project.

The retrieval process is done using the sentence that is being translated as a query and the results will be ranked with the Levenshtein edit-distance to find the most likely suggestion.

The results will then be processed again using the alignment information gets from GIZA++ that is currently stored in Lucene’s index to make a replacement of the translating sentence with each result. The process is possible because the alignment done by GIZA++ gives a clue about the words and their translation counterparts that is the part of original sentences and their respective translations. With this information, the application could guess the word to be replaced in the translating sentence. Combining with the information from Levenshtein edit-distance that is done in the ranking process, the application would know which parts of the result could be used and which parts could be removed.

Once the replacement is done, it is common that some parts of the results might still be those of the translating sentence. These parts are words that are not in the results retrieved from Lucene’s index which is why they are not replaced.

However, the application will also provide the function to suggesting those words with two methods: 1) Using information from available dictionary source like Lexitron Look API; 2) Make use of extra information from GIZA++ which also gives information about alignment possibility of all words and their possible translations from the translated document and its

original document. The application will look up this information to find whether those words have the information or not. Once found the application will take those that have reasonable possibility and use for suggesting use to replace those words with them.

3. Progress saving and continuing

While translating, user can save their working progress so that they can continue on their work later. The progress which is the current status of each translation segment block including the translation done will be updated on the collection “Segments”.

4. Project Finalize

Once the project is done, the user has the choice of finalize the project which means that the project is finish and there are no more translation to be done. This function is meant for application performance as it is designed to constantly update the Lucene’s index each time a project is finish which could be time-consuming as well as resource consuming process. Thus finalizing the project is to confirm that the Lucene’s index could be update without the less likelihood of possible need revert back once it is done.

5. Translation Export

The user can export the translated project into an actual document file such as .docx .txt

8.3 Evaluation

Since our system focuses on assisting professional translators and we focus on translation productivity, we want to measure the amount of edit operations the user needs to make for each sentence. For this purpose we use Translation Error Rate (TER) [13] to compare our system translation suggestion output (using translation memory with statistical methods) with the result from Lucene’s retrieval (translation memory approach). The formula for TER is as follows

$$TER = \frac{\# \text{ of edits}}{\text{average \# of reference words}}$$

We use a travel domain Thai-English bilingual corpus to generate evaluation datasets. From the corpus, which contains about 16,000 translation pairs, we select 10 distinct sets of 150 translations pairs. For each set, we construct a translation memory by using a disjoint set from the corpus.

Table 1 Comparisons of system suggestions

| | Plain Translation Memory | System Suggestion | Difference |
|--------------------|--------------------------|-------------------|------------|
| Dataset #1 | 0.880804 | 0.542805 | -0.338 |
| Dataset #2 | 0.969917 | 0.584535 | -0.38538 |
| Dataset #3 | 1.038016 | 0.610861 | -0.42715 |
| Dataset #4 | 0.890902 | 0.573678 | -0.31722 |
| Dataset #5 | 0.980283 | 0.532774 | -0.44751 |
| Dataset #6 | 0.93382 | 0.564906 | -0.36891 |
| Dataset #7 | 0.916416 | 0.564037 | -0.35238 |
| Dataset #8 | 0.927913 | 0.591024 | -0.33689 |
| Dataset #9 | 0.803942 | 0.561156 | -0.24279 |
| Dataset #10 | 0.803942 | 0.529317 | -0.27463 |
| Average | 0.914596 | 0.565509 | -0.34909 |

From table 1, we can see that the translations suggested by our system reduced the TER by about 0.34 which means that, using translation memory with statistical methods construct translation suggestions reduces the number the user have to edit comparing to when using only translation memory.

To evaluate whether using string edit distance to rank the suggestions improves the precision of the translations, we use the same evaluation sets to compare the target ranks (the rank of the sentence with the lowest TER) by comparing the suggestions with and without re-ranking using edit-distance prior to translation construction.

Table 2 Number of target sentences found in each rank (without using edit-distance)

| Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---------|-------------|-------------|-----------|-------------|------------|------------|------------|------------|------------|------------|-------|
| #1 | 62 | 33 | 13 | 12 | 13 | 3 | 4 | 3 | 3 | 3 | 149 |
| #2 | 58 | 25 | 21 | 11 | 9 | 11 | 5 | 1 | 9 | 0 | 150 |
| #3 | 68 | 28 | 21 | 11 | 8 | 3 | 5 | 3 | 2 | 1 | 150 |
| #4 | 68 | 24 | 11 | 20 | 8 | 4 | 3 | 6 | 3 | 2 | 149 |
| #5 | 64 | 28 | 22 | 13 | 6 | 2 | 3 | 5 | 3 | 3 | 149 |
| #6 | 76 | 27 | 15 | 12 | 7 | 5 | 3 | 2 | 2 | 1 | 150 |
| #7 | 62 | 28 | 19 | 21 | 6 | 7 | 2 | 2 | 1 | 2 | 150 |
| #8 | 66 | 31 | 20 | 7 | 7 | 5 | 4 | 2 | 6 | 2 | 150 |
| #9 | 68 | 30 | 14 | 13 | 6 | 8 | 1 | 4 | 4 | 2 | 150 |
| #10 | 73 | 21 | 14 | 13 | 8 | 5 | 5 | 6 | 0 | 5 | 150 |
| Average | 66.5 | 27.5 | 17 | 13.3 | 7.8 | 5.3 | 3.5 | 3.4 | 3.3 | 2.1 | |

Table 3 Number of target sentences found in each rank (using edit-distance)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---------|-------------|------------|-------------|-------------|-------------|------------|------------|------------|------------|------------|-------|
| #1 | 47 | 19 | 19 | 18 | 7 | 10 | 7 | 10 | 7 | 5 | 149 |
| #2 | 47 | 23 | 18 | 10 | 12 | 5 | 8 | 11 | 9 | 7 | 150 |
| #3 | 50 | 20 | 23 | 15 | 5 | 8 | 7 | 11 | 10 | 1 | 150 |
| #4 | 46 | 31 | 19 | 8 | 11 | 13 | 5 | 4 | 5 | 7 | 149 |
| #5 | 50 | 19 | 18 | 14 | 13 | 10 | 8 | 7 | 4 | 6 | 149 |
| #6 | 51 | 19 | 19 | 14 | 10 | 15 | 4 | 8 | 1 | 9 | 150 |
| #7 | 43 | 29 | 9 | 17 | 17 | 10 | 8 | 9 | 2 | 6 | 150 |
| #8 | 52 | 26 | 15 | 13 | 10 | 13 | 8 | 3 | 3 | 7 | 150 |
| #9 | 43 | 25 | 15 | 15 | 13 | 7 | 9 | 9 | 7 | 7 | 150 |
| #10 | 65 | 20 | 20 | 8 | 10 | 8 | 1 | 7 | 8 | 3 | 150 |
| | | 23. | | | | | | | | | |
| Average | 49.4 | 1 | 17.5 | 13.2 | 10.8 | 9.9 | 6.5 | 7.9 | 5.6 | 5.8 | |

8.4 Conclusion and Future work

In this work, we aim to provide an open-source translation tool for professional translators, which can be accessed through any platform. We proposed web-based translation toolset as a solution, which makes use of Translation Memory and Statistical Machine Translation techniques.

We try to build the system as a platform which can be developed and improved through an open-source community. Our focus is on improving translation suggestions by experimenting with other machine translation techniques.

Also, we would like to provide an easy way to extend the range of languages, especially for languages in the AEC.

9 References

- [1] Esplà Gomis, Miquel, Felipe Sánchez Martínez, and Mikel L. Forcada Zubizarreta. "Using word alignments to assist computer-aided translation users by marking which target-side words to change or keep unedited."
- [2] Higinbotham, Dan. "Partial sentence translation memory program."
- [3] Koehn, Philipp, and Jean Senellart. "Convergence of translation memory and statistical machine translation." *Proceedings of AMTA Workshop on MT Research and the Translation Industry*. 2010.
- [4] Barrachina, Sergio, et al. "Statistical approaches to computer-assisted translation." *Computational Linguistics* 35.1 (2009): 3-28.
- [5] Franz Josef Och, Hermann Ney. "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics*, volume 29, number 1, pp. 19-51 March 2003.
- [6] GNU General Public License, version 2, <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>
- [7] Lucene, <https://lucene.apache.org/>
- [8] Apache License, version 2.0, <http://www.apache.org/licenses/LICENSE-2.0>
- [9] G. Salton, A. Wong, and C. S. Yang (1975), "A Vector Space Model for Automatic Indexing," *Communications of the ACM*, vol. 18, nr. 11, pages 613–620
- [10] MongoDB, <http://www.mongodb.org/>
- [11] MongoDB Licensing, <http://www.mongodb.org/about/licensing/>
- [12] LEXiTRON,
http://lexitron.nectec.or.th/2009_1/index_en.php?q=common_manager/download
- [13] Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. "A study of translation edit rate with targeted human annotation." In *Proceedings of association for machine translation in the Americas*, pages 223-231. 2006.

Appendix A: File directory

This is how the directory of the program looks like:

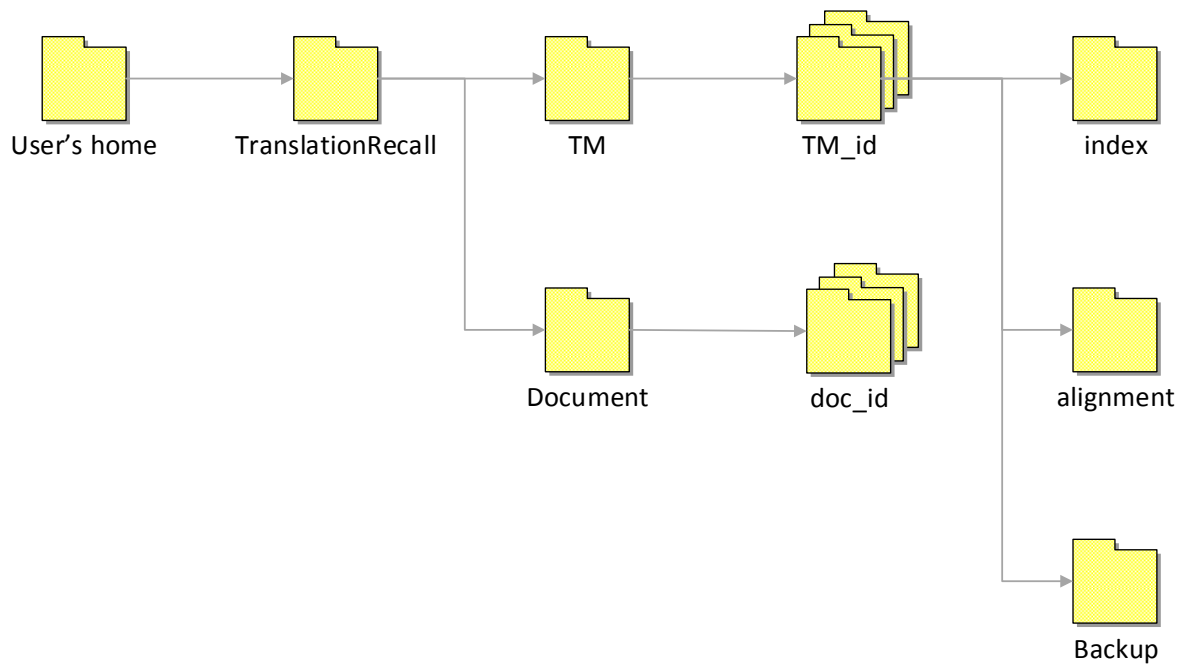


Figure 5

Description:

- User's home – The home folder of the current user.
- TranslationRecall – Main folder of the program which contains data used in the program.
- TM – Contains many Translation Memory folders with unique name.
- TM_id – Each folder contains Translation Memory and the information about alignment.
- Index – Contains Translation Memory files.
- Alignment – Contains alignment information of the Translation Memory.
- Backup – Contains backup data of previous version of Translation Memory.
- Document – Contains many unique named folder that contains data about the translating document.
- Doc_id – Each folder contains data of translating document.

Appendix B: Installation guide

The guide is divided into two parts which are:

1) System Environment Preparation

2) Program Installation

101) System Environment Preparation

Requirement:

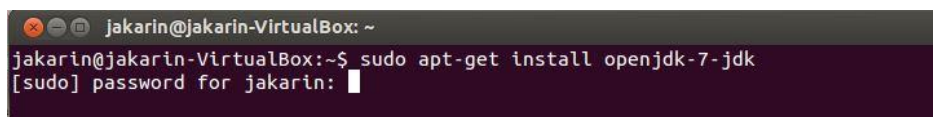
- Operating System: Ubuntu 12.04 or above
- Administrator user right on the operating system
- Internet connection

a) Open-jdk version Installation

- i) Open “Terminal” on Ubuntu and type the following command (# indicates the start of command):

```
# sudo apt-get install openjdk-7-jdk
```

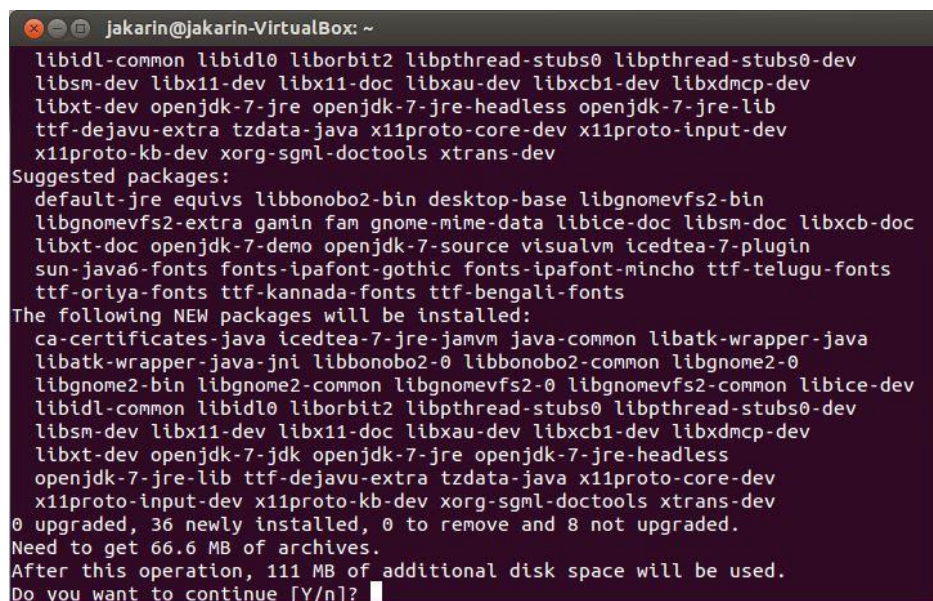
- ii) Terminal will prompt for the password of the user, type in the password as requested.



```
jakarin@jakarin-VirtualBox: ~
jakarin@jakarin-VirtualBox:~$ sudo apt-get install openjdk-7-jdk
[sudo] password for jakarin: 
```

Figure 6

- iii) Terminal will prompt for the confirmation of installing open-jdk, type in “y” and press Enter to proceed on.



```
jakarin@jakarin-VirtualBox: ~
libidl-common libidl0 liborbit2 libpthread-stubs0 libpthread-stubs0-dev
libsm-dev libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev
libxt-dev openjdk-7-jre openjdk-7-jre-headless openjdk-7-jre-lib
ttf-dejavu-extra tzdata-java x11proto-core-dev x11proto-input-dev
x11proto-kb-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
default-jre equivs libbonobo2-bin desktop-base libgnomevfs2-bin
libgnomevfs2-extra gamin fam gnome-mime-data libice-doc libsm-doc libxcb-doc
libxt-doc openjdk-7-demo openjdk-7-source visualvm icedtea-7-plugin
sun-java6-fonts fonts-ipafont-gothic fonts-ipafont-mincho ttf-telugu-fonts
ttf-oriya-fonts ttf-kannada-fonts ttf-bengali-fonts
The following NEW packages will be installed:
ca-certificates-java icedtea-7-jre-jamvm java-common libatk-wrapper-java
libatk-wrapper-java-jni libbonobo2-0 libbonobo2-common libgnome2-0
libgnome2-bin libgnome2-common libgnomevfs2-0 libgnomevfs2-common libice-dev
libidl-common libidl0 liborbit2 libpthread-stubs0 libpthread-stubs0-dev
libsm-dev libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev
libxt-dev openjdk-7-jdk openjdk-7-jre openjdk-7-jre-headless
openjdk-7-jre-lib ttf-dejavu-extra tzdata-java x11proto-core-dev
x11proto-input-dev x11proto-kb-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 36 newly installed, 0 to remove and 8 not upgraded.
Need to get 66.6 MB of archives.
After this operation, 111 MB of additional disk space will be used.
Do you want to continue [Y/n]? 
```

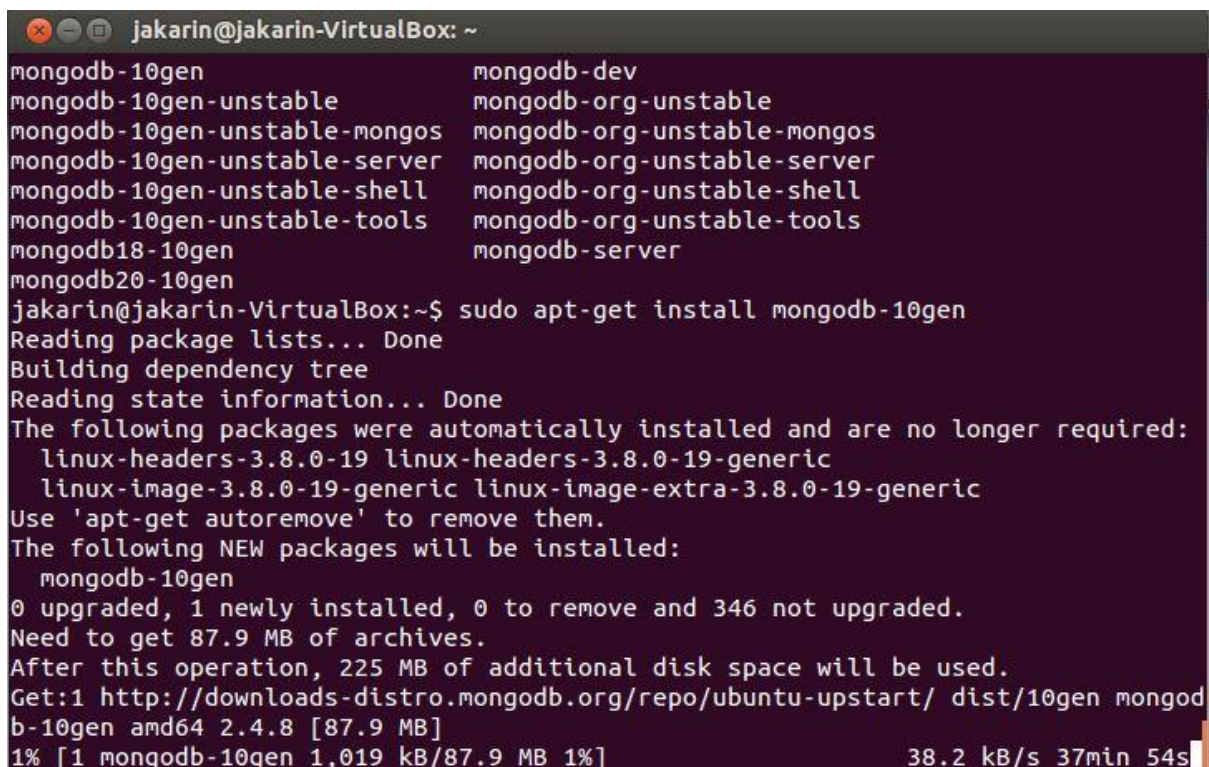
Figure 7

- iv) Wait until the installing successful is prompted by Terminal.

b) MongoDB Installation

- i) Open “Terminal” on Ubuntu and type in the following commands (# indicates the start of command):

```
# sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
# echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' |
sudo tee /etc/apt/sources.list.d/mongodb.list
# sudo apt-get update
# sudo apt-get install mongodb-10gen
```



```
jakarin@jakarin-VirtualBox: ~
mongodb-10gen
mongodb-10gen-unstable
mongodb-10gen-unstable-mongos
mongodb-10gen-unstable-server
mongodb-10gen-unstable-shell
mongodb-10gen-unstable-tools
mongodb18-10gen
mongodb20-10gen
mongodb-dev
mongodb-org-unstable
mongodb-org-unstable-mongos
mongodb-org-unstable-server
mongodb-org-unstable-shell
mongodb-org-unstable-tools
mongodb-server
jakarin@jakarin-VirtualBox:~$ sudo apt-get install mongodb-10gen
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-3.8.0-19 linux-headers-3.8.0-19-generic
  linux-image-3.8.0-19-generic linux-image-extra-3.8.0-19-generic
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
  mongodb-10gen
0 upgraded, 1 newly installed, 0 to remove and 346 not upgraded.
Need to get 87.9 MB of archives.
After this operation, 225 MB of additional disk space will be used.
Get:1 http://downloads-distro.mongodb.org/repo/ubuntu-upstart/ dist/10gen mongod
b-10gen amd64 2.4.8 [87.9 MB]
1% [1 mongodb-10gen 1,019 kB/87.9 MB 1%] 38.2 kB/s 37min 54s
```

Figure 8

- ii) Once all commands are run by Terminal, type in the following commands to start the Process of MongoDB:

```
# sudo service mongodb start
```

c) Glassfish server installation

- i) Download Glassfish installation file from
“<https://glassfish.java.net/download.html>”

- ii) Open Terminal and type in the following commands, with the filename of the installation file in this case is “glassfish-4.0-unix.sh”:

```
# sudo chmod +x glassfish-4.0-unix.sh  
# sudo sh glassfish-4.0-unix.sh
```

- iii) A new window will pop up similar to the fig.9 below. Click the Next button to proceed.

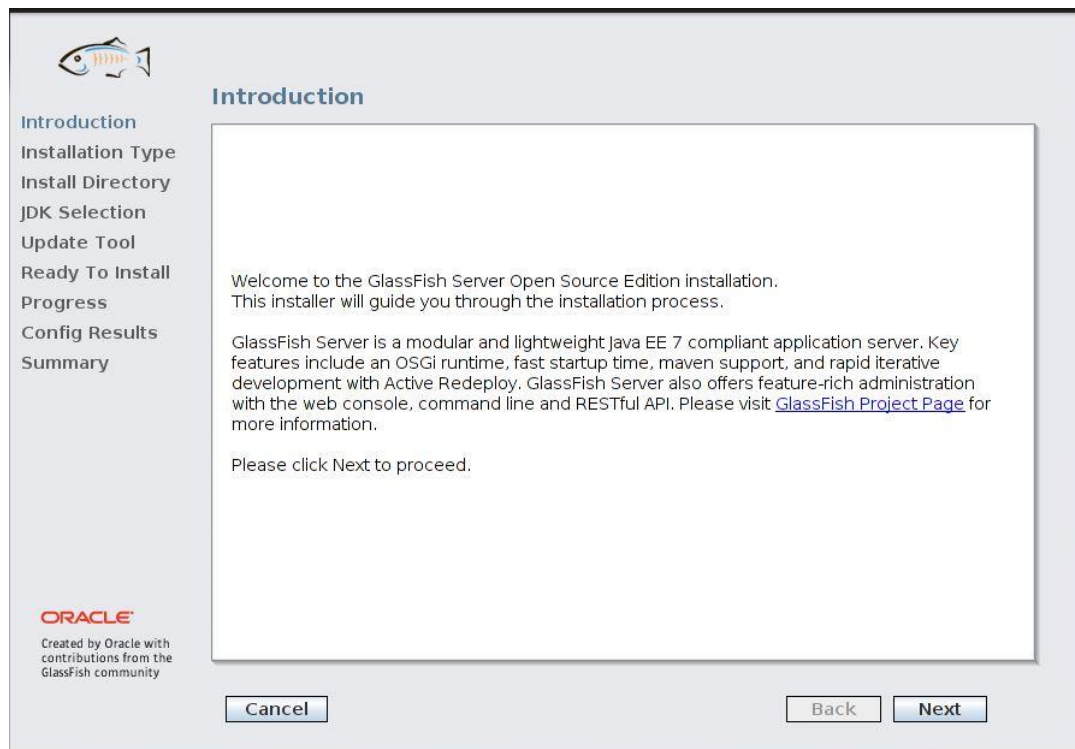


Figure 9

- iv) On the next page, choose “Typical Installation” and click Next button.

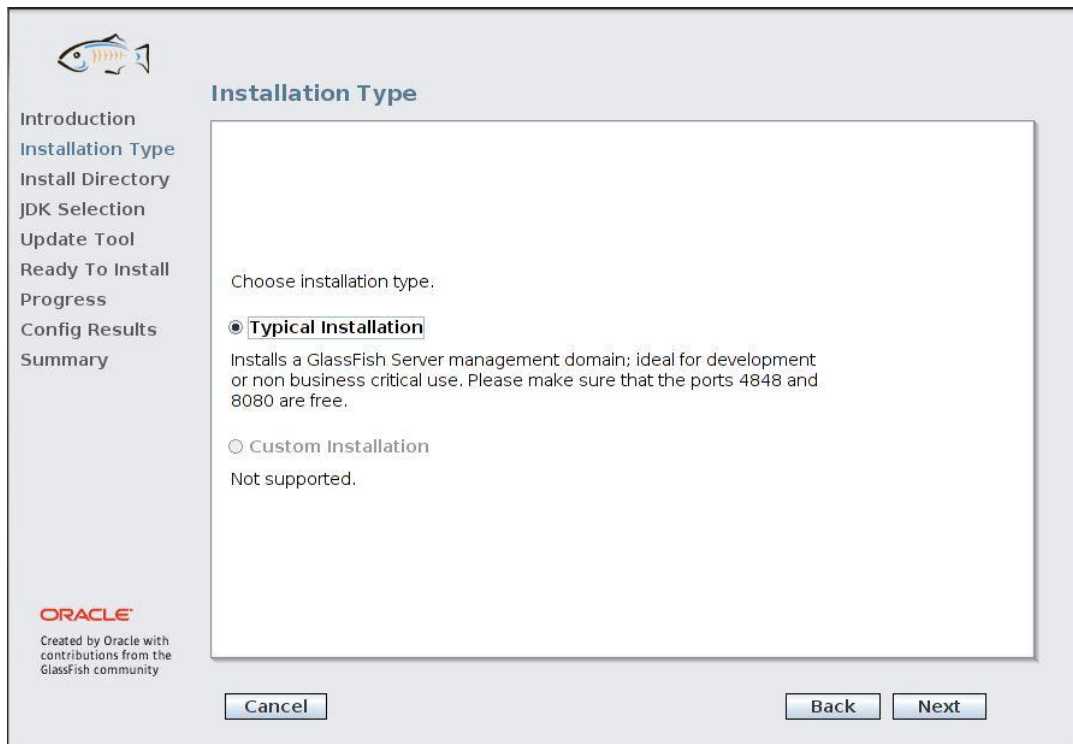


Figure 10

- v) Next, the window will prompt for choosing installation directory, this can be left as default. Click Next button to proceed.

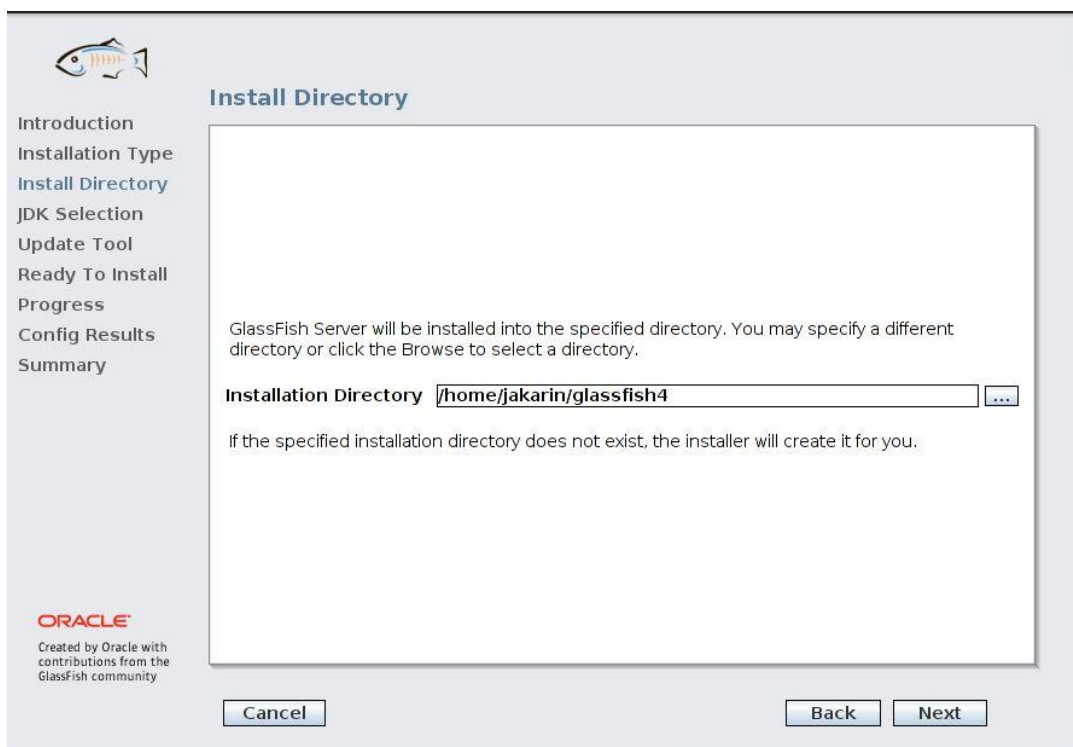
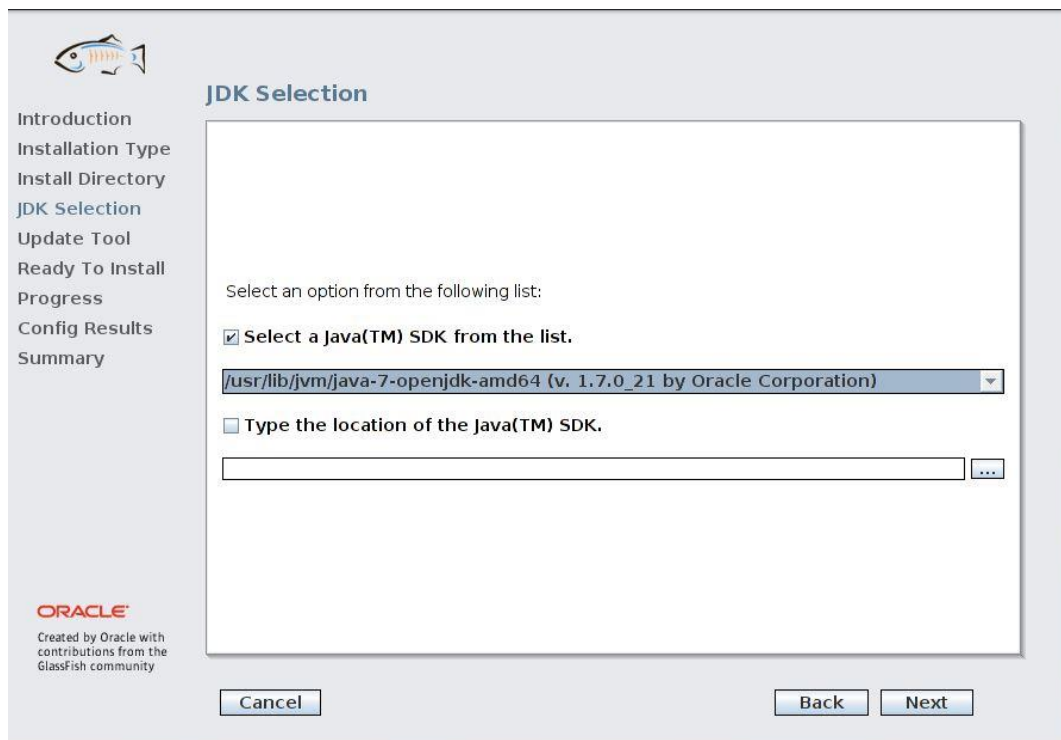


Figure 11

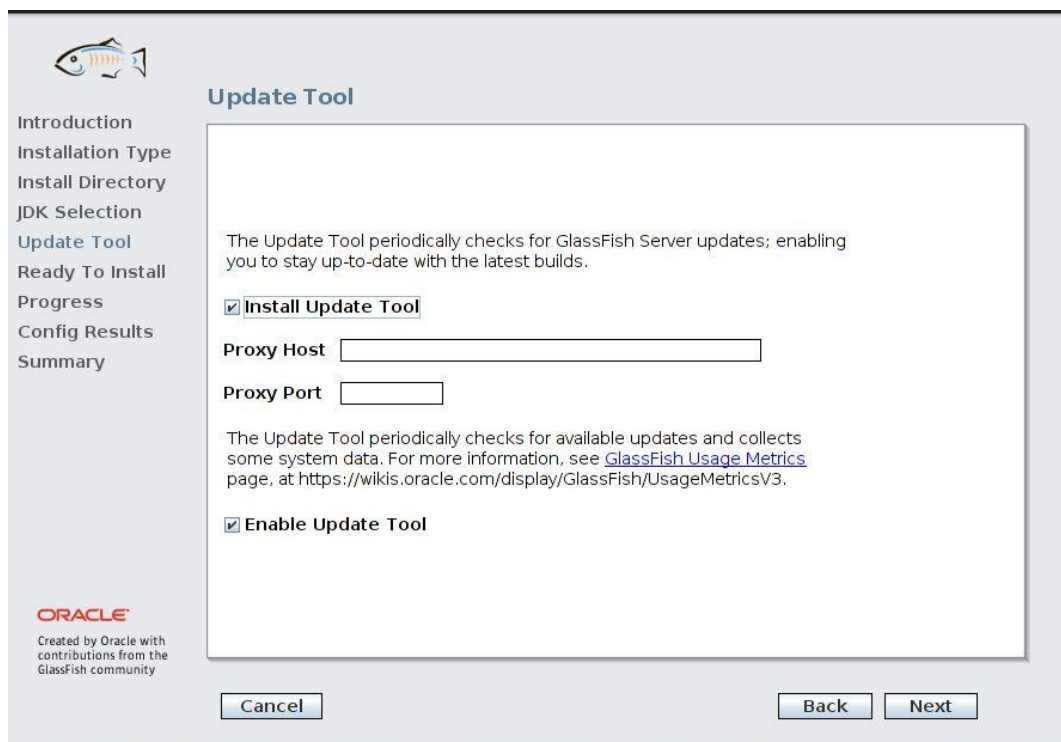
- vi) On the next page, choose “jdk” (Java Development Kit) that is installed and click Next to proceed.



The image shows the 'JDK Selection' window in the GlassFish installation wizard. On the left is a sidebar with a list of steps: Introduction, Installation Type, Install Directory, **JDK Selection**, Update Tool, Ready To Install, Progress, Config Results, and Summary. The main area is titled 'JDK Selection' and contains the text 'Select an option from the following list:'. There are two options: a checked checkbox for 'Select a Java(TM) SDK from the list.' and an unchecked checkbox for 'Type the location of the Java(TM) SDK.'. The first option has a dropdown menu showing '/usr/lib/jvm/java-7-openjdk-amd64 (v. 1.7.0_21 by Oracle Corporation)'. The second option has a text input field and a browse button (...). At the bottom are 'Cancel', 'Back', and 'Next' buttons. The Oracle logo and a note about contributions from the GlassFish community are in the bottom left.

Figure 12

- vii) On this page, installation of update tool will be confirmed, this could be decided to do or not, then click Next to proceed.



The image shows the 'Update Tool' window in the GlassFish installation wizard. The sidebar on the left is the same as in Figure 12, with 'Update Tool' now highlighted. The main area is titled 'Update Tool' and contains the text: 'The Update Tool periodically checks for GlassFish Server updates; enabling you to stay up-to-date with the latest builds.' There are two checked checkboxes: 'Install Update Tool' and 'Enable Update Tool'. Below the first checkbox are input fields for 'Proxy Host' and 'Proxy Port'. Below the second checkbox is a link to 'GlassFish Usage Metrics' with the text 'page, at https://wikis.oracle.com/display/GlassFish/UsageMetricsV3.' At the bottom are 'Cancel', 'Back', and 'Next' buttons. The Oracle logo and a note about contributions from the GlassFish community are in the bottom left.

Figure 13

- viii) Then the summary of installation to be done will be confirmed, click Next button to proceed on installation process.

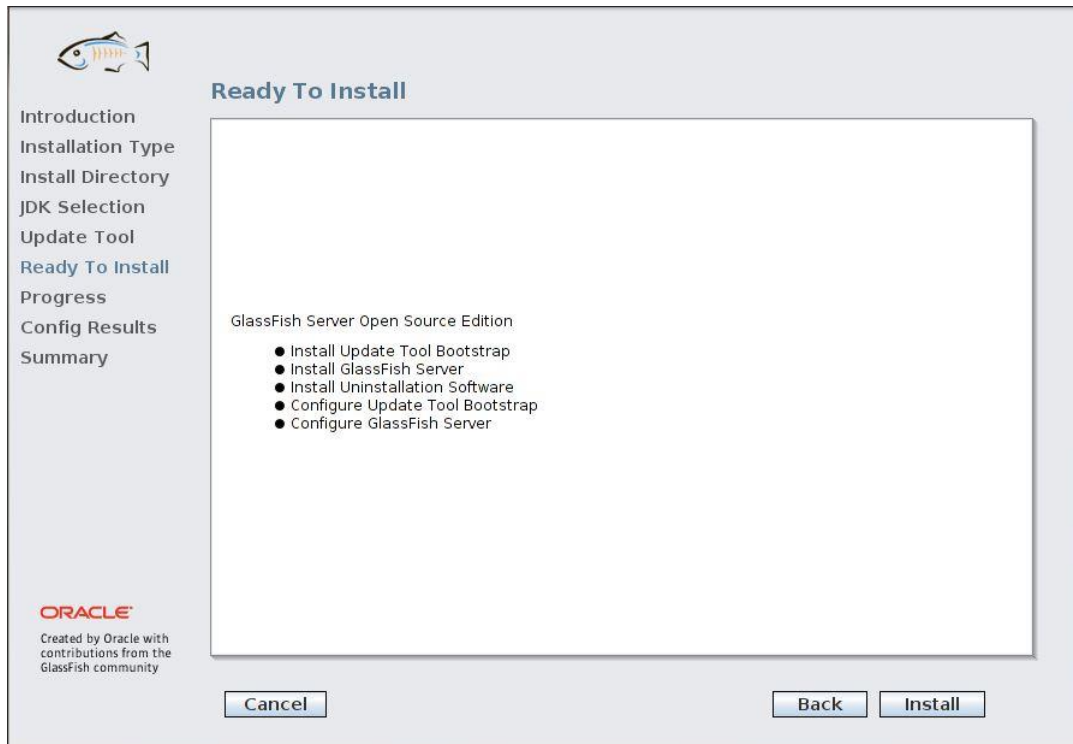


Figure 14

- ix) After the process is completed, the page similar to what fig.15 depicts will be displayed. Click Next to proceed.

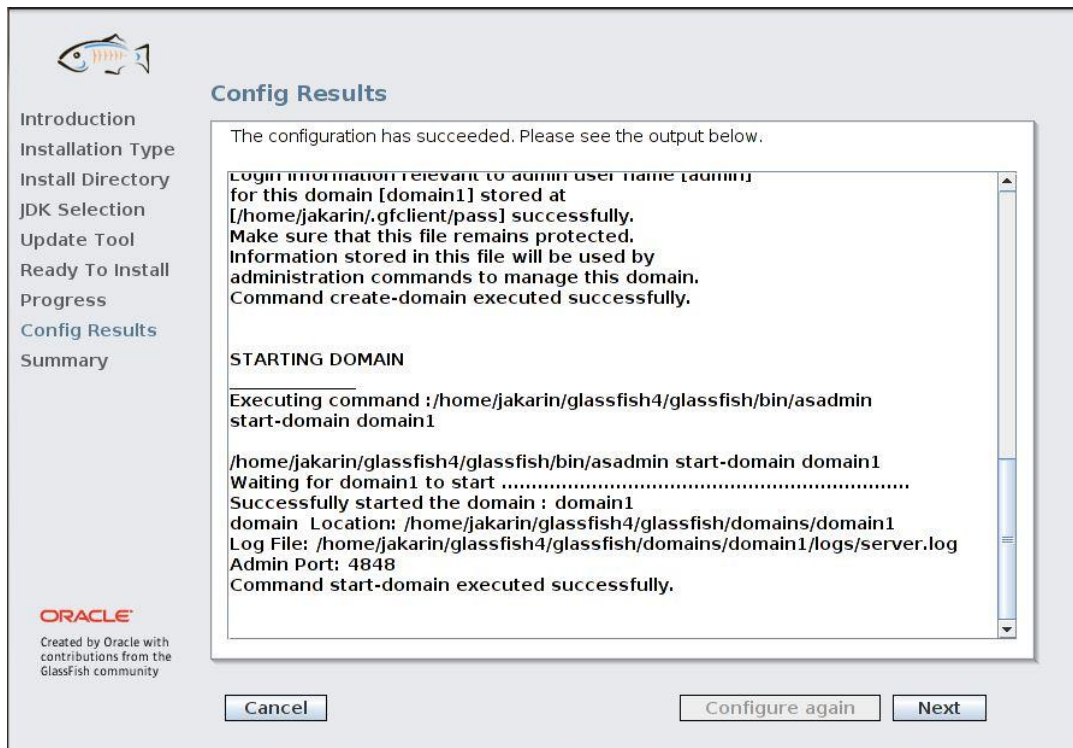


Figure 15

- x) The final page will show to summary of installation, click Exit button to end the installation.

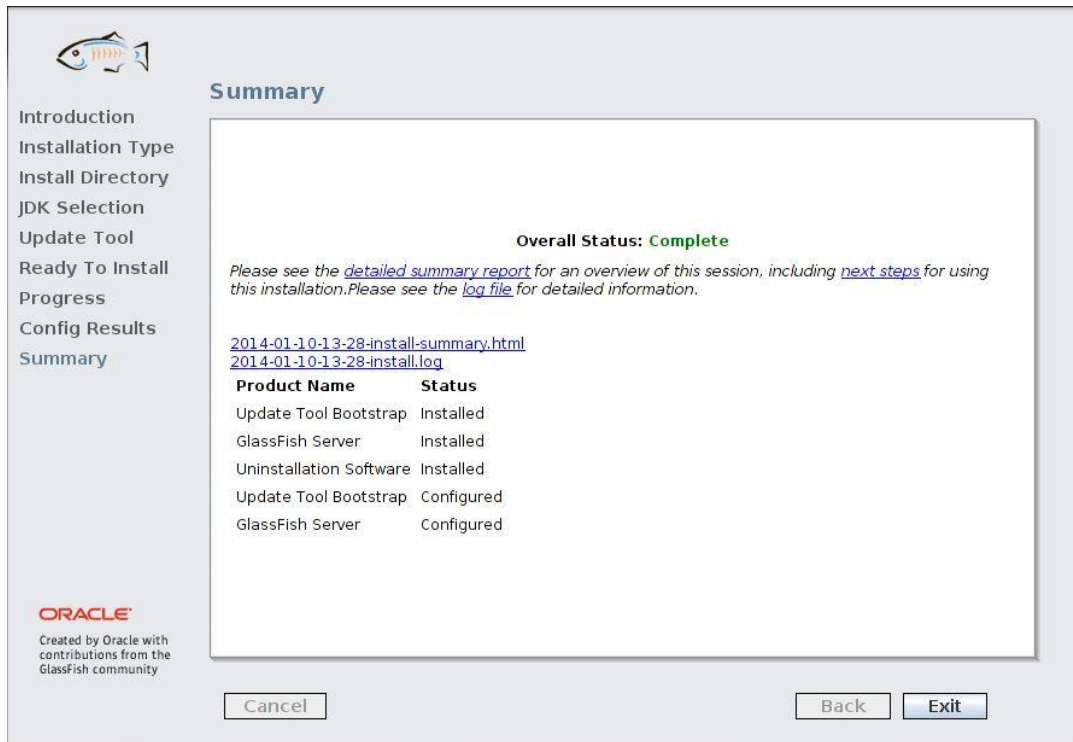


Figure 16

- xi) Open a web browser and type in “localhost:4848” into address bar and press Enter. If the web browser displays similar to what shown in fig.17, then the installation of glassfish is completed normally.

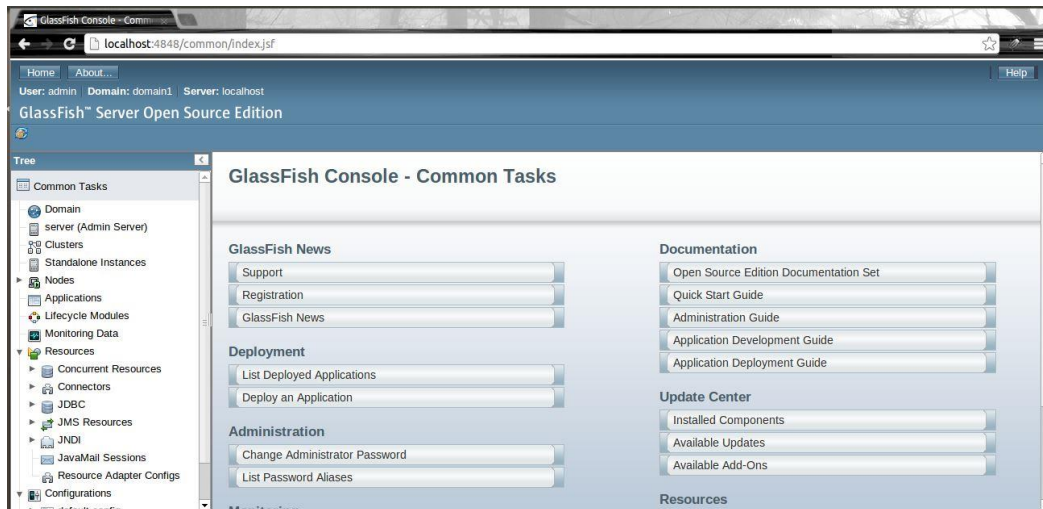


Figure 17

2) Program Deployment

- i) Open a web browser and type in “localhost:4848” into address bar and press Enter to open the glassfish’s management page. (If the page cannot be loaded even if the glassfish is installed, open Terminal and type in the following command)

```
# glassfish4/bin/asadmin start-domain
```

- ii) Click on “Deploy an Application” as shown in the red rectangle in fig.19.

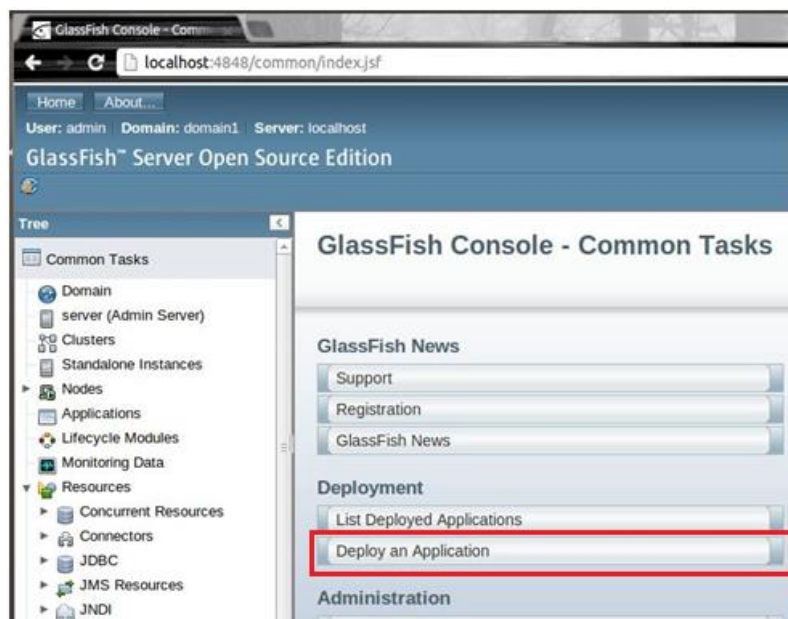


Figure 18

- iii) On “Location”, choose “Packaged File to Be Uploaded to the Server”, then click on Choose File button and select file “TranslationRecall.war” that is downloaded from “https://ict.siit.tu.ac.th/viewvc/projects/2013/senior/tt1/src/translationrecall/dist”.

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

Location: ☒ Packaged File to Be Uploaded to the Server

TranslationRecall.war

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Figure 19

- iv) On “Context Root”, change the value into “/” while, on “Virtual Servers”, select “server” from the list, then click OK button to proceed on deploy the application.
- v) After the deployment is complete, the page will be similar to what fig.21 depicts.

Applications

Applications can be enterprise or web applications, or various kinds of modules. Re: targets that the application or module is enabled on.

| Deployed Applications (1) | | |
|-------------------------------------|--------------------------|--|
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="button" value="Deploy..."/> <input type="button" value="Undeploy"/> <input type="button" value="Enable"/> <input type="button" value="Disable"/> Filter: <input type="text"/> |
| Select | Name | Deployment Order |
| <input type="checkbox"/> | TranslationRecall | 100 |

Figure 20

- vi) Go to “localhost:8080”, the start page as shown in fig.22 would show up.

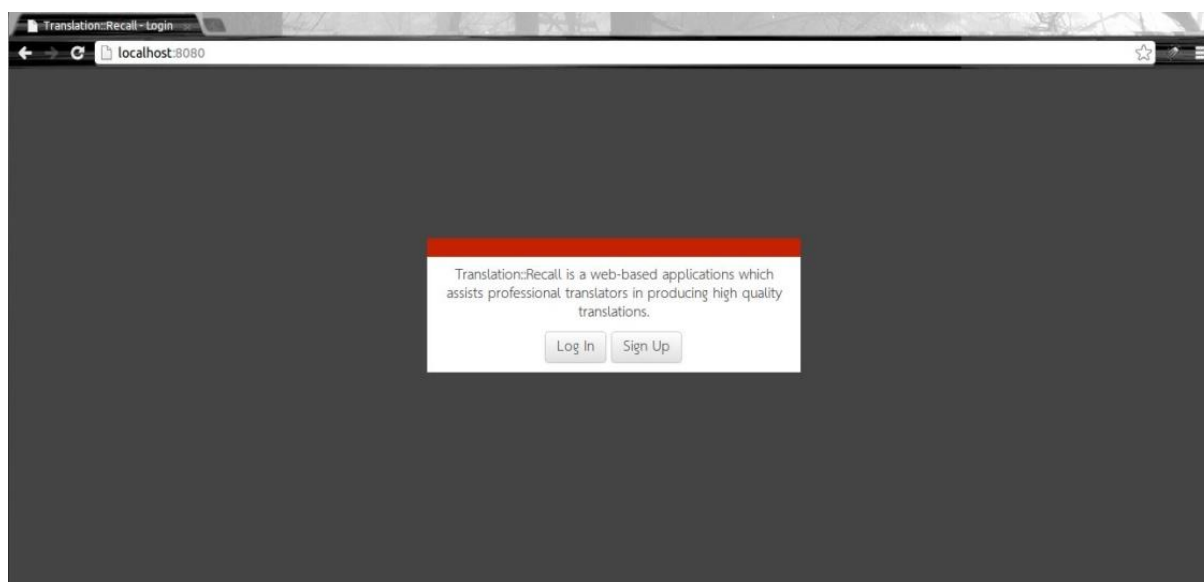


Figure 21

Appendix C: User manual

1) Start Page

a) Account Registration/System log in

- i) In order to use TranslationRecall, the user must register by clicking on “Sign Up” and log in into the system.

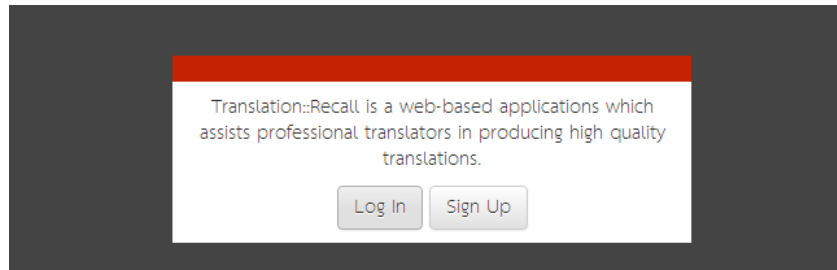


Figure 22 Start page is being displayed

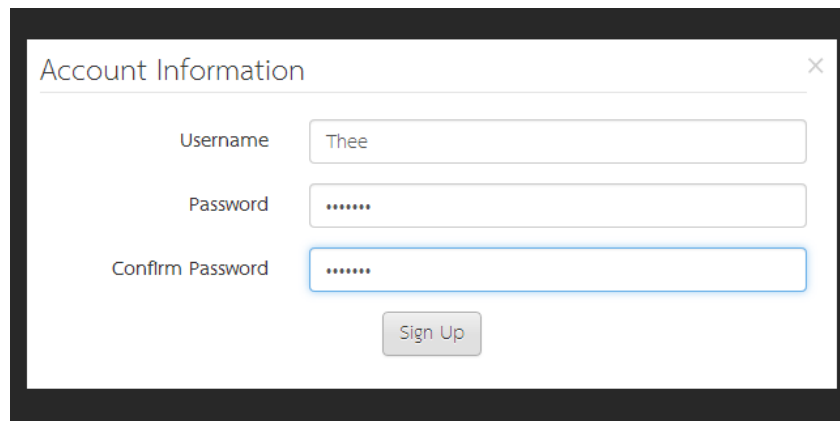


Figure 23 New user account registration page is being displayed

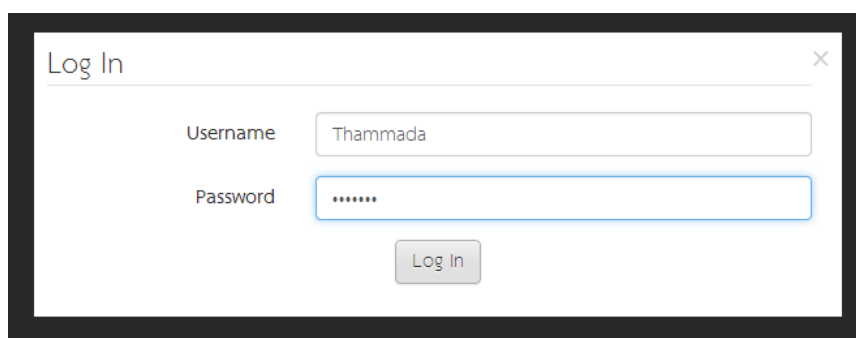


Figure 24 Login page is being displayed

- ii) If the login process is successful, the user will be redirected to the project list page

2) User Project List page

On the user project list page, the user will see all the translation document.

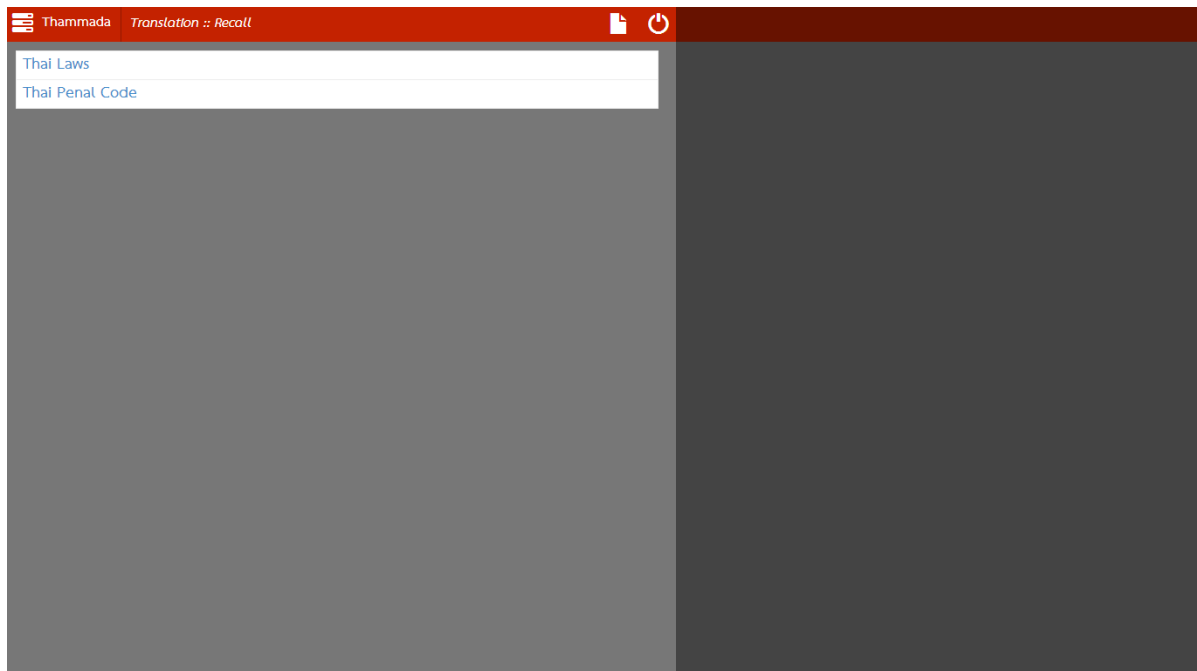


Figure 25 User Project List page is being displayed

a) Create new project

- i) The user can create a new project by clicking on the “new project” icon

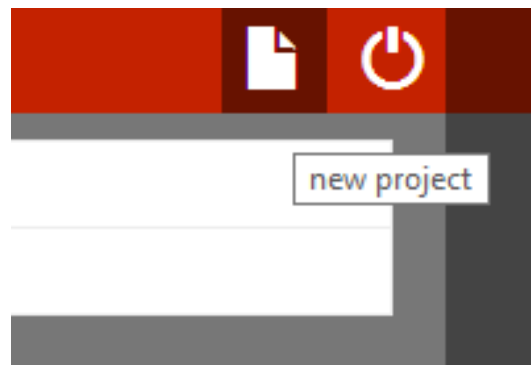
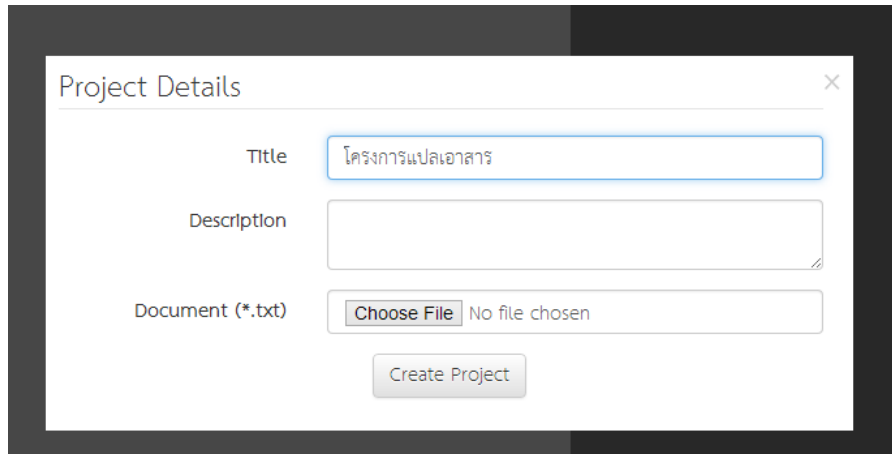


Figure 26 New Project Icon

- ii) A pop-up dialog will be displayed, the user can enter details about the new project and select the document to be translated. The document to be imported must be in plain text format (file extension *.txt) where every sentence is segmented by a new line character. When the document is successfully imported, the user will be redirected to the page “Document Editor”



Project Details

Title: โครงการแปลเอกสาร

Description:

Document (*.txt): Choose File No file chosen

Create Project

Figure 27 New Project Creating Form

b) System log out

The user can log out of the system by clicking on the “Log out” icon

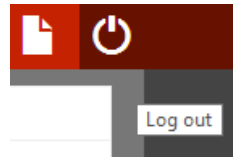


Figure 28 Log out Icon

c) Open project/ Edit Project’s detail/ Delete project

When the user points the cursor at the project name, a menu will appear for the following operations: “Open”, “Edit” and “Delete” project



Figure 29 Open, Edit , Delete Menu Icon for each available project, from left-to-right, Open, Edit, Delete

3) Document Editor page

The document editor contains two main sections; the left pane contains the current document and translation progress. The right pane shows translation suggestions.

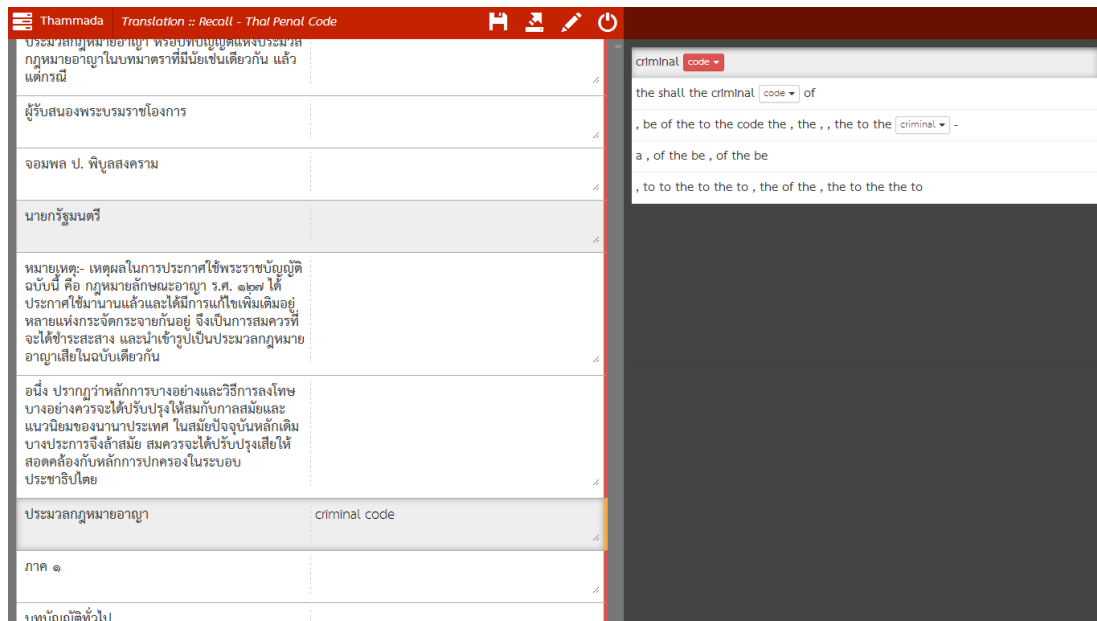


Figure 30

a) Start Translation

- i) The user selects a sentence to be translated by clicking on the source language sentence.
- ii) The system will suggest appropriate translation suggestions. The user can traverse through the source sentence by clicking on one of the source sentences or using the shortcut **Ctrl+↑** / **Ctrl+↓**
- iii) In case the system cannot find an exact word translation from the stored memory, a list of word will be suggested in a dropdown menu
- iv) The user can traverse through the translation suggestions by using the shortcut **Ctrl+Shift+↑** / **Ctrl+Shift+↓**
- v) On each sentence in the document, a small stripe signifies the status of each sentence:
 - Red : untranslated
 - Yellow : currently translating
 - Green : translated

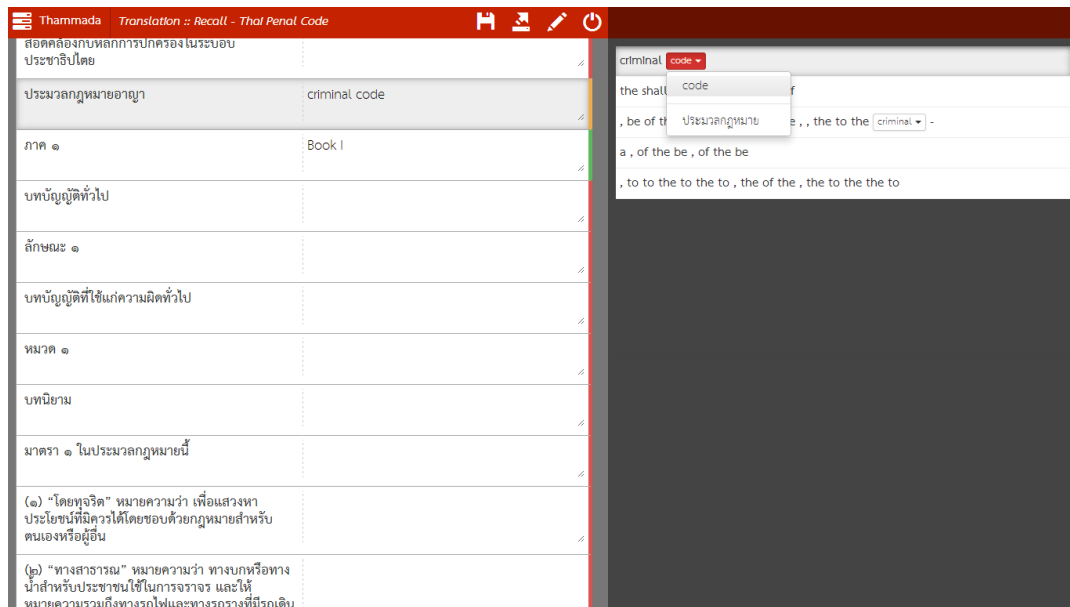


Figure 31 : Example of Translation Process

b) Save translation progress/ Export translated document/ Edit project's detail

On the top tool bar, the user will find icons for saving, exporting and editing document information.



Figure 32 Project Menu Icon, from left-to-right, Save, Export, Edit Project Menu

c) Dictionary lookup with LEXiTRON-Look

The user can lookup a word using LEXiTRON Look by drag-select the desired word

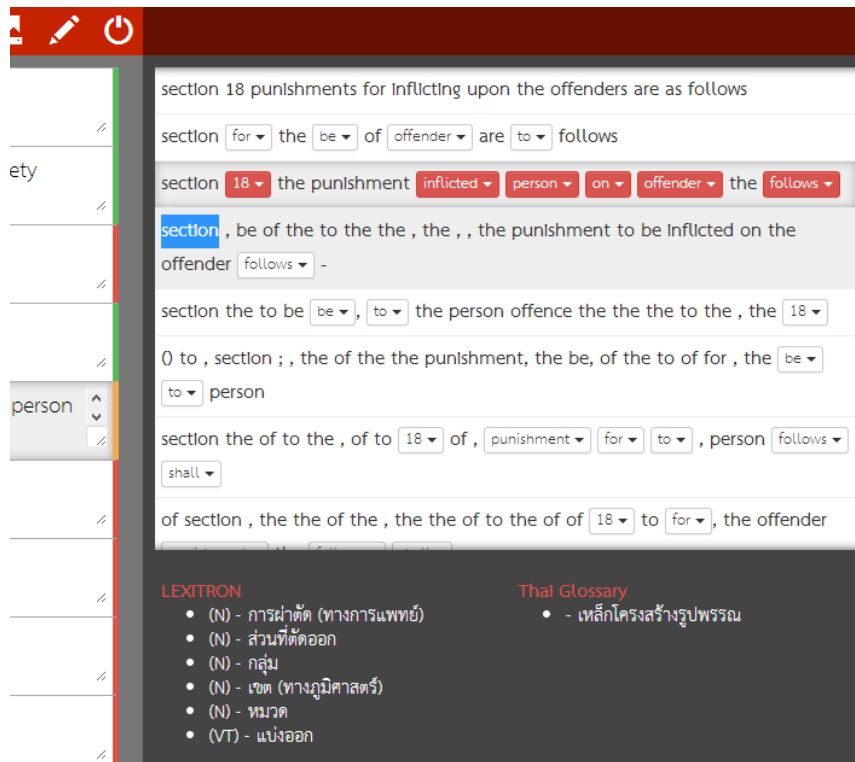


Figure 33 Example of LEXiTRON-look result