

Software Design

Specification

**Terminology Creation**

*Thao Thai, Bikram Pokhrel, and Mary Hogan*

*Revision 1.1*

**Table of Contents**

1. Project Description.…………………………………………………………………….………3

2. Components…………………………………………………………………………………….4

2.1 Algorithm..……………………………………………………………………………...…4

2.2 Database...…………….…………………………………………………………………...6

2.3 Website…………….……….……………….…………………………………………….6

3. Third Party Technologies...……………………………………………………………………..7

4. Timeline………………………………………………………………………………………...7

**Section 1 – Project Description**

**1.1 Project**

Terminology Creation

**1.2 Description**

There is a lack of translation for technical terminologies in the World Wide Web. As a solution to this problem, we come up with Terminology Creation. Terminology Creation is a web user-interface translator that takes an English technical terminology from the user and provides the corresponding translation of it in a given language. Users will also be able to input their own translation of technical terms in the languages that they are fluent in. In addition, a user can edit a translation to a more fitting foreign term if they see fit. Terminology Creation connects the user not only through web user interface but also the developer.

**1.3 Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Comment** | **Author(s)** |
| 10/6/2016 | Requirement Document | T. Thai, B. Pokhrel, M. Hogan |
| 2/12/2016 | Design Document | T. Thai, B. Pokhrel, M. Hogan |

**Section 2 – Components**

**2.1 Algorithm**

Pointwise Mutual Information (PMI) uses the co-occurrence of the source language (English) and the target language (foreign) to measure the frequency of both terms appearing together in a corpus. The overview of the algorithm is shown in the UML Activity Diagram below.



***Figure 1.0 UML Activity Diagram***

The algorithm best running time is O(n), if all the sentences in the files have the length of 1. Its worst running time is O(n3).

From previous evaluation, PMI alone yields an average accuracy of 3.68% across 31 different languages. Our goal for this term was to find a way to improve the accuracy of PMI algorithm. The new algorithm we implement is cPMI, or simply PMI with the concepts corpus-level significant co-occurrence. According the research of Om Damani, corpus-level significant co-occurrence determines whether the ratio of observed bigram occurrences to their expected occurrences across the corpus is a pure chance phenomenon. cPMI incorporates , where is a chosen parameter varying between 0 and 1 as shown in Table 1. By taking the probability of observing a given deviation between and its expected value in account, cPMI addresses one of the main weakness of PMI of working only with probabilities and completely ignoring the absolute amount of evidence (Damani, 2013). Therefore, we expect cPMI to yield a higher average for co-occurrences of the English term and its corresponding translation.

|  |  |
| --- | --- |
| **PMI** | **cPMI** |
| PMI: | cPMI: |

***Table 1: Algebric definitions of PMI and cPMI***

In addition to incorporating cPMI, we took into consideration of co-occurences among translated terms themselves. For example, the Vietnamese translation of “environment” is “môi trường,” but cPMI will only yield either “môi” or “trường” or both separately. Therefore, we decided to combine the two targeted terms together into a single translation to accommodate the multiple terms definition of a single English word.

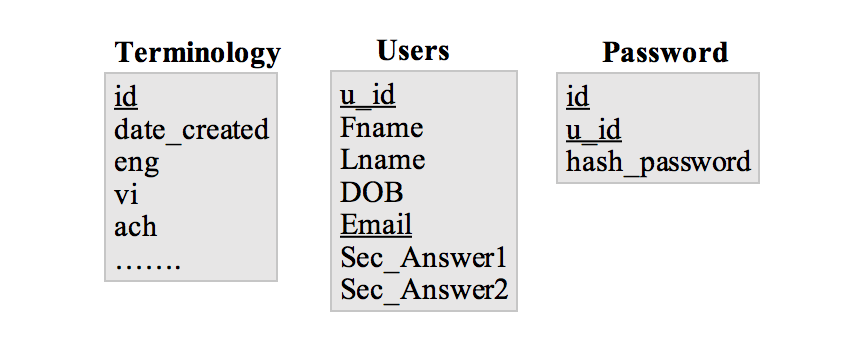
By looking at Figure 2.0, we can see that cPMI’s average accuracy, with the addition of targeted terms co-occurrence, is 3.75% and PMI’s average accuracy is 3.68%. The difference between the average is 0.07%, which is an improvement. However, the number may not significant enough.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Languages** | **PMI** | **cPMI + co-occurences** | **Direct Translation** | **Eval[1]** | **Eval[2]** |
| ach | 64 | 66 | 1409 | 4.54% | 4.68% |
| an | 101 | 98 | 1487 | 6.79% | 6.59% |
| anp | 0 | 3 | 78 | 0.00% | 3.85% |
| arn | 19 | 18 | 481 | 3.95% | 3.74% |
| bo | 31 | 29 | 453 | 6.84% | 6.40% |
| brx | 69 | 63 | 1163 | 5.93% | 5.42% |
| cak | 33 | 32 | 1450 | 2.28% | 2.21% |
| kok | 63 | 47 | 1196 | 5.27% | 3.93% |
| lij | 78 | 73 | 1149 | 6.79% | 6.35% |
| lo | 20 | 22 | 535 | 3.74% | 4.11% |
| meh | 9 | 9 | 256 | 3.52% | 3.52% |
| vi | 30 | 39 | 1208 | 2.48% | 3.23% |
| hi\_IN | 65 | 63 | 1358 | 4.79% | 4.64% |
| hto | 19 | 20 | 677 | 2.81% | 2.95% |
| mni | 55 | 37 | 1000 | 5.50% | 3.70% |
| my | 55 | 37 | 1270 | 4.33% | 2.91% |
| or | 62 | 50 | 1235 | 5.02% | 4.05% |
| mxp | 6 | 6 | 167 | 3.59% | 3.59% |
| nd | 25 | 23 | 389 | 6.43% | 5.91% |
| neb | 2 | 3 | 189 | 1.06% | 1.59% |
| tl | 30 | 28 | 744 | 4.03% | 3.76% |
| trs | 6 | 6 | 286 | 2.10% | 2.10% |
| vi | 33 | 56 | 1208 | 2.73% | 4.64% |
| wo | 30 | 32 | 1094 | 2.74% | 2.93% |
| my | 55 | 38 | 1270 | 4.33% | 2.99% |
| mni | 55 | 47 | 1000 | 5.50% | 4.70% |
| nch | 0 | 1 | 112 | 0.00% | 0.89% |
| ncj | 0 | 4 | 145 | 0.00% | 2.76% |
| neb | 2 | 3 | 189 | 1.06% | 1.59% |
| or | 62 | 58 | 1235 | 5.02% | 4.70% |
| hus | 2 | 4 | 204 | 0.98% | 1.96% |
|  |  |  | **Average** | **3.68%** | **3.75%** |

***Figure 2.0 Probability of PMI and cPMI with co-occurences on the targeted terms***

**2.2 Database**

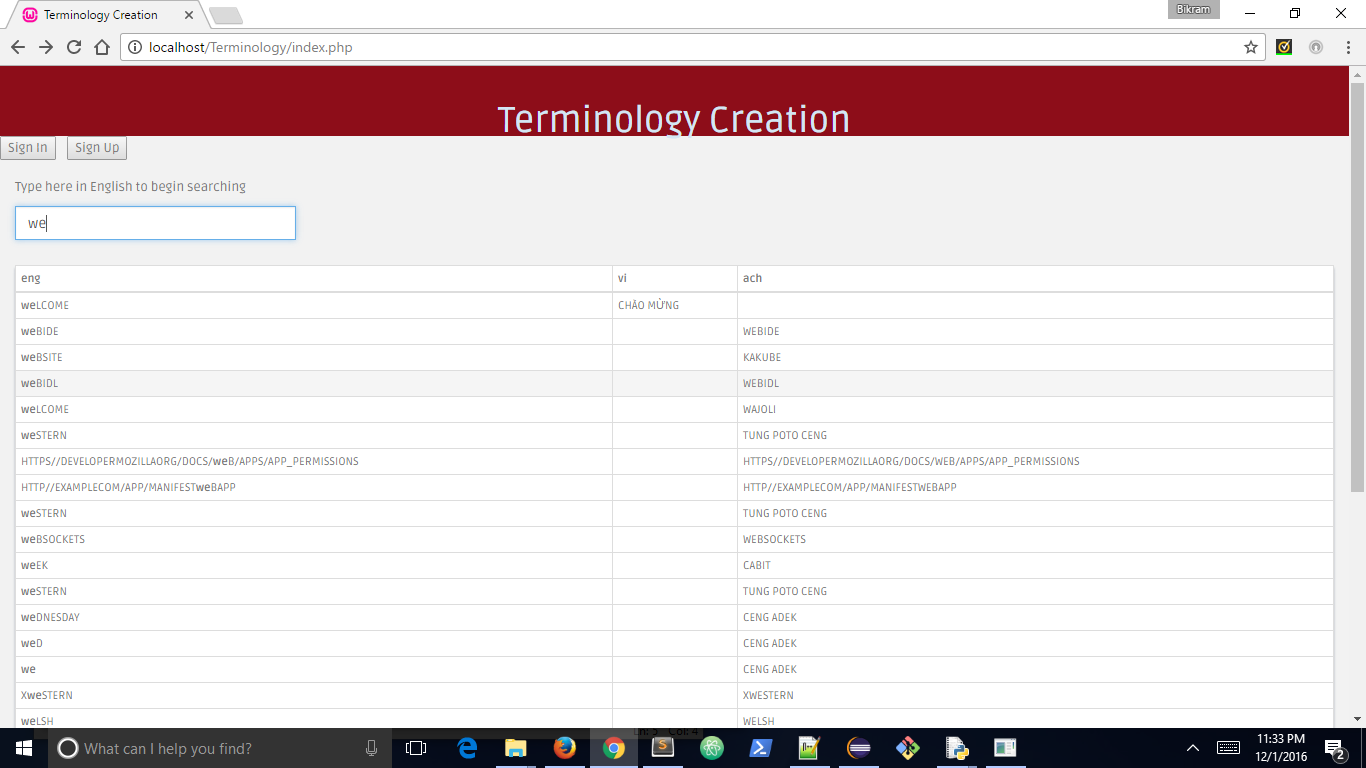
We are using PHP to manipulate our SQL database. Looking at Figure 3.0, the database consists of a table that stores a list of technical terms in English and their translations. It uses the output of our algorithm to populate the table. In addition, we have two entities “users” and “password” that store registration information and hash password of users.



***Figure 3.0 Database Schema***

**2.3 JavaScript Website**

Our website is running on JavaScript. The user can select a language to be translated to. Once the user starts typing the word, the website will display a list of the corresponding technical terms in English and their translations in a given language. This demo can be seen in Figure 4.0. If a translation does not exist, the space will be left empty, so any user with an account can login and update the database by editing or adding to the translations.



***Figure 4.0 Terminology Creation Website***

**Section 3 – Third Party Technologies**

We use polib module in our algorithm. Polib is a Python library made for editing and creating gettext files (i.e. po files). In addition, we use an open source language library provided by Mozilla Firefox, containing over 200 languages and their translations of technical terms.

**Section 4 – Project Timeline**

For the remainder of the project cycle, we will look into Damani’s related works of cPMId. According to Damani, it has been found that corpus-level significant co-occurrence alone does not play a big role in improving PMI algorithm. We will need to enforce document-level significant co-occurrence, which determines whether a large fraction of a word-pair’s occurences within a given document have smaller spans than that under a null model.

By the end of this semester, we will provide a more accurate method of calculating probability. To do this, we can ignore common words that would result in a very high probability. Additionally, we consider groups of more than one word because, for many languages, one English term may translate into a term of multiple words. We also create our username and password database for user registration and a working front-end to our website.

By midterm of next semester, we will look into Damani’s related works of cPMId. According to Damani, it has been found that corpus-level significant co-occurrence alone does not play a big role in improving PMI algorithm. We will need to enforce document-level significant co-occurrence, which determines whether a large fraction of a word-pair’s occurences within a given document have smaller spans than that under a null model. Furthermore, we would like to evaluate the accuracy of our algorithm by comparing it against a dictionary for at least 100 languages and continuing to debug and test the application to ensure a working front-end and back-end. Lastly, by the end of next semester, we want to have a fully functional, bug-free application.

**Source:**

O. Damani. (2013). *Improving Pointwise Mutual Information (PMI) by Incorporating Significant Co-occurrence.* In S. Bulgaria. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (pp. 20-28). Association for Computational Linguistics.

Polib Documentations: https://pypi.python.org/pypi/polib