Web-based Refining of Machine Translations

Renzo Angles, Rodrigo Paredes, Federico Meza, Marcos Gutiérrez, Felipe Valdebenito, Danilo Yáñez
Departamento de Ciencias de la Computación
Universidad de Talca
Curicó, Chile
Email: {rangles,raparede,fmeza}@utalca.cl, {magutierrez, fvaldebenito, dyanez}@alumnos.utalca.cl

***Abstract*—In this article we describe the architecture of our Web-based platform for refining machine translations. The main idea is to use the Web as a database of phrases, and use this information in order to improve the quality of translations. The platform considers three modules, namely: crawling, indexing, and refining. This is an ongoing work, and currently we are capable to take an English phrase and produce its refined Spanish translation.**

***Keywords*-Machine translation; Similarity searching.**

I. Introduction

The hypothesis of our work is that automatic translations [1] produced by Web services can generate inconsistent translations of the source text, and such mismatches can be refined by using the Web as a source of well-written phrases.

To illustrate our point, in Table I we show the results obtained by several translation Web services for the phrase "*I used to be in love*," whose correct Spanish translation is "*Solía estar enamorado*." As can be seen, only one of the tools can generate the right phrase. Note, however, that some of those translations can be easily refined so as to obtain the correct translation by changing or introducing a few words.

Table I
Examples of machine translations of the phrase: "*I used to be in love*."

Translation service	(URL)	Result
Google translate	(translate.google.com)	Yo solía ser en el amor
Yahoo! Babel Fish	(babelfish.yahoo.com)	Estaba en amor
SYSTRANet	(www.systranet.com)	Estaba en amor
Babylon	(traductor.babylon.com)	Yo solía estar en amor
yacom	(traductor.ya.com)	Solé estar enamorado
elmundo.es	(www.elmundo.es/traductor)	Solía estar enamorado

The main ideas of our proposal consist on building a database of (likely) well written phrases obtained from the Web, and later use it as a knowledge base for refining automatic translations. This obeys the intuition that well written phrases are more common than the wrong ones, so we can use the occurrence frequency of a phrase as a ranking criterion for similarity searching.

There are two main sources of related work. The first (and more important) comes from Machine Translation [1]. There are several approaches to cope the automatic translation problem, most of them based on corpus and statistical techniques, however there is not an effective one [2]. The second source correspond to Information Retrieval [3]. To the best of our knowledge, we are not aware of any previous published work on the subject of using the Web as a source in order to improve the quality of automatic translations.

A. Indexing methods

In order to efficiently retrieve information out of a textual database, first we need to index it [4]. Since our retrieving problem consists on obtaining phrases from a dataset containing an specific set of query words, a well suited data structure is the inverted index [3], also called postings file or inverted file.

An *inverted index* is a data structure storing a mapping from words to all their locations in the document collection. For the sake of building the index, we first compute the vocabulary of the whole collection. Later, for each vocabulary term we store all the locations (for instance, the document identifiers) containing it. In order to save space, we can only store the document identifier that contain the word. However, we can also be more specific by storing additional location information, but spending more space. So, given a query, that is, a set of words $q = \{w_1, \ldots, w_k\}$, we obtain the posting lists for each query word and intersect them. If the resulting set is not empty, all the hits in the intersection of the posting lists contain all the query words, so they can be considered as relevant. In order to refine the result, we can rank the resulting set according to a scoring function.

II. Refining model

A *machine translation* is a function $MT : L_s \rightarrow L_t$ where L_s is a source natural language and L_t is a target natural language. Considering that there is not an effective approach for machine translation, we have that for some text $t \in L_s$ it applies that $MT(t)$ is not a valid translation for t.

We aim that there is a function $\Re$ which transforms an invalid translation $MT(t)$ in a valid one by applying some transformations to $MT(t)$. The function $\Re$ is called a *refining function* for machine translation.

In order to simplify the definition of our refining model, we reduce the problem to translate phrases. A *phrase* is a group of words that form a single unit in the syntax of

a sentence and has a meaning by itself. Two phrases are similar if they have the same meaning.

Let F be a database of well-written phrases in a natural language L and S be a similarity metric between phrases. Given a phrase f in L, we define the function $\Re_S^F(f) = f'$ such that $f' \in F$ and $S(f, f') > S(f, f'')$ for every phrase $f'' \in F - \{f'\}$.

Let f be a machine translation of a phrase f_s, and f_v be a valid translation of f_s satisfying that $f \approx f_s$. We aim that $S(\Re_S^F(f), f_v) > S(f, f_v)$.

III. Implementation

In order to validate our refining model, we have designed a refining system composed of three modules, namely: Crawling, Indexing, and Refining. Figure 1 shows the general architecture of the system. In this section we describe the work done in each module.

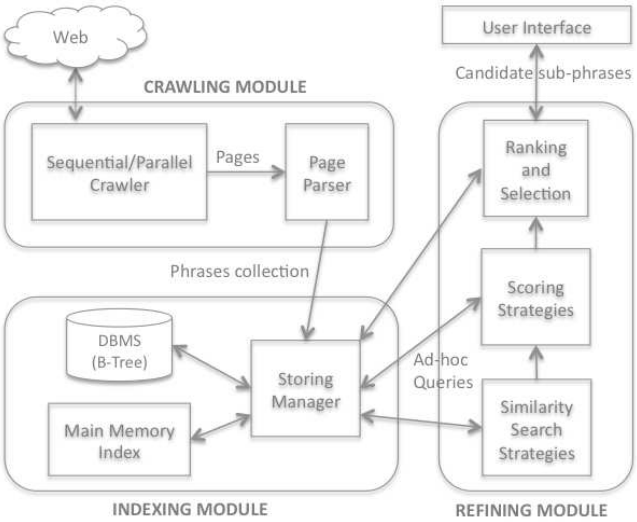

Figure 1. Architecture of the refining system.

The indexing module implements suitable structures for storing and querying phrases, words, and the relations among all of them. The refining module implements the refining function based on two similarity metrics between phrases.

A. Crawling module

The crawling module is responsible for collecting and parsing Web pages in order to build the database of phrases in the target language.

Our crawler is a parallel program comprising processes of two kinds: *readers* and *writers*. Readers download and analyze web pages; writers store on plain text files the phrases collected by the readers, storing the associated URL's on a database. Communication between readers and writers is accomplished by Remote Method Invocation (RMI). Three tasks are performed in parallel: URL scheduling, download

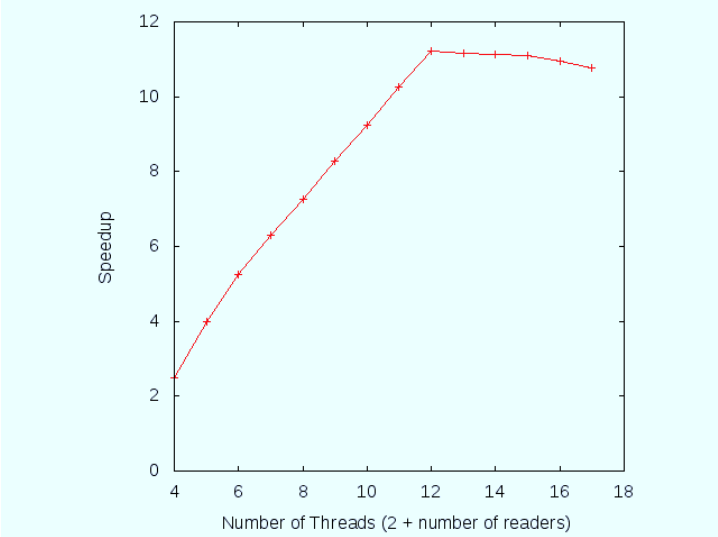

Figure 2. Speedup of the parallel crawler for different configurations.

and analysis of web pages, and storage of information on disk.

In order to find the best configuration for the parallel crawler, we performed a series of experiments comparing its performance against a sequential single-threaded crawler. Both the sequential and the parallel crawler were setup to retrieve 5000 pages selected at random. Figure 2 shows the speedup for several configurations. It can be seen that maximum efficiency is achieved by using 12 threads, that is, 10 readers.

Aiming to find the optimal number of writer processes, we performed an experiment where the crawler was run for 20 minutes measuring the number of writers working when a writer is needed. We conclude that the best configuration includes 3 writers.

B. Indexing module

We adapt the inverted index to our problem as follows. For each vocabulary term, we store the identifier of the phrase containing it. So given a query, a set of words, for each of them we retrieve its posting list and later compute their intersection. See [5] for further details.

This module implements suitable structures for storing and querying phrases, words, and the relations among all of them. The basic process is to take a phrase, split it according to our definition of phrase, and then store both the phrase and its words in the inverted list index [3] running on the database. For this sake, we use a standard relational database (PostgreSQL 8.3) indexed by B-trees and Hash.

We validate the effectiveness of the database indices. As expected, Figure 3 shows indexing the tables with b-trees or hash tables dramatically improves the results, and both

indices show mild differences in performance. On the other hand, we run experiments to measure the size of each table. When we use no index, hash index and b-tree index we spend 171, 325 and 277 bytes per phrase, respectively, in the database tables. Hence, we prefer to index the tables using b-trees in order to obtain fast response reducing the space usage.

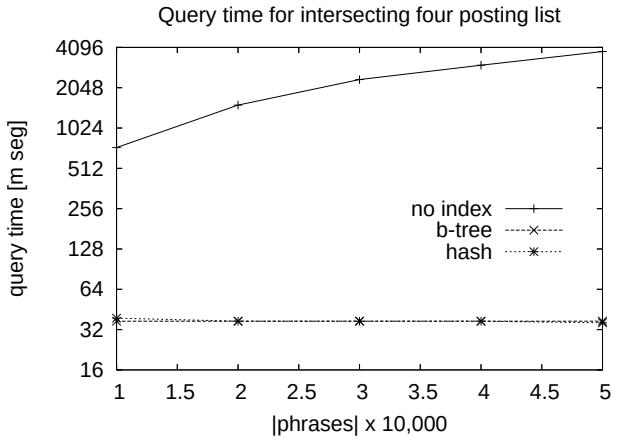

Figure 3. Time for solving queries composed by three words.

C. Refining module

This module implements the refining function presented in Section II. First we describe some criteria to determine the similarity between word. Then we define two metrics that can be used to implement a similarity search strategy. A detailed description of this module can be found in [6].

Along this section, we assume that we have a set of well-written phrases obtained from the crawling module, and accessible via the indexing module. Additionally, we define a *phrase* as a sequence of at most n words separated by prepositions or punctuation symbols. After computing some statistics on the database, we fixed $n = 7$.

In order to determine the similarity between two phrases, we consider the following criteria:

1) *Coincident words*: This refers to take into account the number of words that occur in both phrases. We consider exact and fuzzy coincidence. Since it is possible that several phrases tie, the fuzzy criterion tries to break the tie by taking into account the number of stopwords that occurs in the phrases.
2) *Relevant words*: This criterion is directly related with the criterion of coincident words. It consists in increasing the similarity degree when the coincident words are relevant, and decreasing the degree in case of being stopwords (irrelevant words). Note that relevant words are directly related with the meaning of a phrase whereas stopwords just act as connectors.
3) *Order of words*: This criterion evaluates whether the coincident words are in the same order. It assumes that having words in the same order increases the similarity between phrases.
4) *Number of words*: This criterion measures the difference of length (in terms of the number of words) between phrases. This follows the intuition that the greater the difference between the length of two phrases, the more likely they have different meaning.
5) *Verbal tense*: This criterion is given by the equality between the verbal tenses of the phrases.
6) *Phrase context*: The context of a phrase refers to the topic where the phrase is meaningful. The context can be determined by the occurrence of particular words in the phrase.

Similarity search: We combine and evaluate the criteria described above, in order to select a search strategy for retrieving, from the list of well-written phrases, a subset of phrases considered candidate for the refining process.

Consider a formal definition for a general similarity search strategy. Given a phrase f, a set of phrases F, and a similarity criteria S, the function $Search(f, F, S)$ returns a set of phrases $F' \subseteq F$, such that every phrase in F' satisfies the similarity criteria S with respect to f. Every phrase in F' is called a *candidate phrase*.

We test four similarity criteria for searching:

(1) Exact coincidence, considering the order in the sequence of word.
(2) Exact coincidence, considering relevant words.
(3) Fuzzy coincidence by using combinations of relevant words.
(4) Fuzzy coincidence by using all combinations among words.

In order to determine the best searching strategy, we consider three factors to evaluate: precision to obtain all candidate phrases; computation time, and the number of phrases returned (no necessarily candidates). The results of the experiments, over a database of 30.000 phrases, are shown in Table II). Considering that the most important selection criterion is the precision of the method, we select method (4) as the best candidate selection method.

criteria	Time(ms)	Phrase found	Candidates
(1) Exact coincidence	68	0	0
(2) Exact coincidence + Relevant words	63	1	1
(3) Fuzzy coincidence + Partial combination	136	8	7
(4) Fuzzy coincidence + Total combination	188	107	12

Table II
EVALUATION OF CRITERIA FOR SIMILARITY SEARCHING

IV. Conclusions and Future Works

We have developed a proof of concept application aimed to improve the quality of poorly translated phrases from English to Spanish. The experimental evaluation has shown that, given a source phrase and its translation, both similarity phrase metrics successfully rank the well-written phrase among the top 10 most relevant phrases. Although neither metric performs better than the other when computing the rank of relevant phrases, evidence suggests that they can work together in order to obtain better results.

Our refining approach has several advantages:

1) The refining system is independent of the source language.
2) The system model is able to smoothly incorporate other target languages.
3) The system can obtain the original translation from any machine-translation Web service.

A. Future work

For future work, we plan to enhance the correctness and quality of the refined texts. So, we need both to improve the quality and the size of the phrase database. For this sake, we plan to extend our crawler to support other significant phrase sources (for instance, electronic books).

On the other hand, it is crucial to support a massive knowledge database. In fact, our preliminary tests show that a traditional DBMS (or an ad-hoc index in a single machine) could not be able to support the storage and querying requirements of the system. This motivates us to research on ad-hoc indices for secondary memory, and also on a distributed refining platform.

We need to improve the candidate selection method by introducing some notion of ranking, in order to obtain the most similar phrase among the candidates. We also consider to customize the refining to an specific variant or dialect of a given language which could be done by using phrase sources from such specific language variant.

Finally, we also plan to develop a user-friendly interface which allows the final user to visualize and select the candidate phrases.

References

[1] W. J. Hutchins and H. L. Somers, *An Introduction to Machine Translation*. Academic Press, 1992.

[2] M. W. Madsen, "The limits of machine translation," Master's thesis, Department of Scandinavian Studies and Linguistics, Faculty of Humanities, University of Copenhagen, December 2009.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001. [Online]. Available: http://mitpress.mit.edu/algorithms/

[5] D. Yáñez, "Efficient data retrieval for phrase-based refining of machine translations using the Web as a knowledge base," March 2011, Universidad de Talca, Computer Science final project. In Spanish.

[6] M. Gutiérrez, "Design and implementation of phrase similarity criteria," March 2011, Universidad de Talca, Computer Science final project. In Spanish.