# TS3: an Improved Version of the Bilingual Concordancer TransSearch

**Stéphane Huet, Julien Bourdaillet** and **Philippe Langlais**
DIRO - Université de Montréal
C.P. 6128, succursale Centre-ville
H3C 3J7, Montréal, Québec, Canada
{huetstep,bourdaij,felipe}@iro.umontreal.ca

## Abstract

Computer Assisted Translation tools remain the preferred solution of human translators when publication quality is of concern. In this paper, we present our ongoing efforts conducted within TS3, a project which aims at improving the commercial bilingual concordancer TransSearch. The core technology of this Web-based service mainly relies on sentence-level alignment. In this study, we discuss and evaluate the embedding of statistical word-level alignment.

## 1 Introduction

Although the last decade witnessed an impressive amount of effort devoted to improving the current state of Machine Translation (MT), professional translators still prefer Computer Assisted Translation (CAT) tools, among which *translation memory* (TM) systems and *bilingual concordancers*. Both tools exploit a TM composed of a *bitext*: a set of pairs of units (typically sentences) that are in translation relation. Whereas a TM system is a translation device, a bilingual concordancer is conceptually simpler, since its main purpose is to retrieve from a bitext, the pairs of units that contain a *query* (typically a phrase) that a user manually submits. It is then left to the user to locate the relevant material in the retrieved target units. As simple as it may appear, a bilingual concordancer is nevertheless a very popular CAT tool. In (Macklovitch et al., 2008), the authors report that TransSearch,[1] the commercial web-based concordancer we focus on in this study, received an average of 177 000 queries a month over a one-year period (2006–2007).

Figure 1 provides a screenshot of a session with the current concordancer TransSearch. A user submitted the multi-word query in keeping with to which the system responded with a webpage showing the first 25 pairs of sentences in the TM that contain an occurrence of the query. As can be observed, nothing in the target material retrieved is emphasized, which forces the user to read the examples retrieved until an appropriate translation was found.

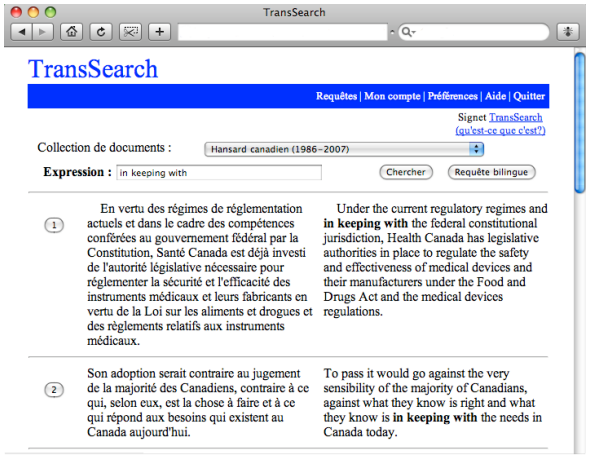


Figure 1: Screenshot of TransSearch. Two of the first 25 matches returned to the user for the query in keeping with.

The main objective of the TS3 project is to automatically identify (highlight) in the retrieved material the different translations of a user query. Identifying translations offers interesting prospects for user-efficient interactions. Although the definitive look-and-feel of the new prototype is not settled yet, Figure 2 shows an interface where the user can consult the most likely translations automatically


[1] www.tsrali.com

identified. Of course, she can still consult the pairs of sentences containing the query, but can as well click a given translation to see related matches.

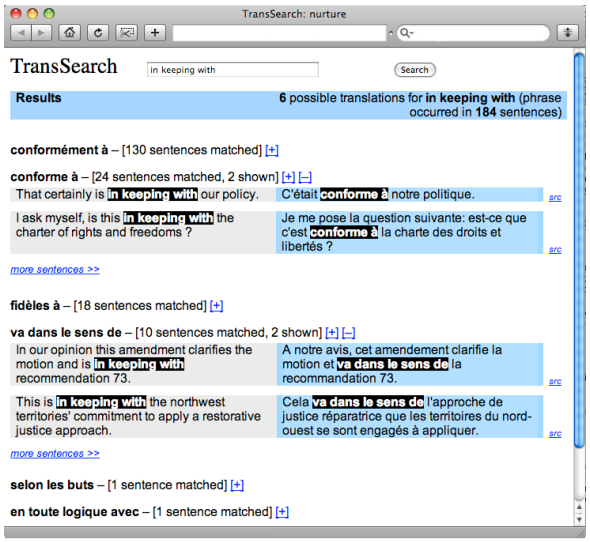


Figure 2: A hypothetical interface which exploits translation spotting.

The reminder of this paper is organized as follows. We first describe in Section 2 the translation spotting technique we implemented. Since translation spotting is a notoriously difficult problem, we discuss two novel issues that we think are essential to the success of a concordancer such as `TransSearch`: the identification of erroneous alignments (Section 3) and the grouping of translation variants (Section 4). We present the data we used in Section 5 and report on experiments in Section 6. We conclude our discussion and propose ongoing avenues in Section 7.

## 2 Transpotting

*Translation spotting*, or *transpotting*, is the task of identifying the word-tokens in a target-language (TL) translation that correspond to the word-tokens of a query in a source language (SL). It is therefore an essential part of the `TS3` project. We call *transpot* the target word-tokens automatically associated with a query in a given pair of units (sentences). For instance in Figure 2, `conformément à` and `va dans le sens de` are two of the six transpots displayed to the user for the query `in keeping with`.

### 2.1 Word Alignment

As mentioned in (Simard, 2003), translation spotting can be seen as a by-product of word-level alignment. Since the seminal work of (Brown et al., 1993), statistical word-based models are still the core technology of today's Statistical MT. This is therefore the alignment technique we consider in this study.

Formally, given an SL sentence $S = s_1...s_n$ and a TL sentence $T = t_1...t_m$ in translation relation, an IBM-style alignment $a = a_1...a_m$ connects each target token to a source one ($a_j \in \{1, ..., n\}$) or to the so-called NULL token which accounts for untranslated target tokens, and which is arbitrarily set to the source position 0 ($a_j = 0$). This defines a word-level alignment space between $S$ and $T$ whose size is in $O(m^{n+1})$.

Several word-alignment models are introduced and discussed in (Brown et al., 1993). They differ by the expression of the joint probability of a target sentence and its alignment, given the source sentence. For computational reasons, we focus here on the simplest form, which corresponds to IBM models 1&2:

$$p(t_1^m, a_1^m | s_1^n) = \prod_{j=1}^{m} \sum_{i \in [0,n]} p(t_j | s_i) \times p(i | j, m, n)$$

where the first term inside the summation is the so-called transfer distribution and the second one is the alignment distribution.

Given this decomposition of the joint probability, it is straightforward to compute the so-called Viterbi alignment, that is, the one maximizing the quantity $p(a_1^m | t_1^m, s_1^n)$. This approach often produces discontiguous alignments, which poses problems in practice. Furthermore, most of the queries in our logfile are contiguous ones, therefore we expect their translations to be contiguous as well.

### 2.2 Transpotting Algorithm

In order to enforce contiguous transpots, we implemented a variant of the transpotting algorithm initially proposed by Simard (2003), and which shares close similarities with phrase extraction technique described in (Venugopal et al., 2003). The idea is to compute for each pair $\langle j_1, j_2 \rangle \in [1, m] \times [1, m]$, two Viterbi alignments: one between the phrase $t_{j_1}^{j_2}$ and the query $s_{i_1}^{i_2}$, and one between the remaining material in the sentences $\bar{s}_{i_1}^{i_2} \equiv s_1^{i_1 - 1} s_{i_2+1}^n$ and $\bar{t}_{j_1}^{j_2} \equiv t_1^{j_1 - 1} t_{j_2+1}^m$. This method, which finds the translation of the query

according to:

$$\hat{t}_{\hat{j}_1}^{\hat{j}_2} = \underset{(j_1,j_2)}{\text{argmax}} \begin{cases} \max_{a_{j_1}^{j_2}} p(a_{j_1}^{j_2}|s_{i_1}^{i_2}, t_{j_1}^{j_2}) \\ \times \\ \max_{\bar{a}_{j_1}^{j_2}} p(\bar{a}_{j_1}^{j_2}|\bar{s}_{i_1}^{i_2}, \bar{t}_{j_1}^{j_2}) \end{cases},$$

has a complexity in $O(nm^3)$. It ranked first among several other alternatives we investigated (Bourdaillet et al., 2009).

### 2.3 The Need for Post-processing

Frequent queries in the TM receive numerous translations by the previously described transpotting process. Figure 3 illustrates the many transpots returned by our transpotting algorithm for the query `in keeping with`. As can be observed, some transpots (those marked by a star) are clearly wrong (*e.g.* `à`), many others (in italics) are only partially correct (*e.g.* `conformément`). Also, it appears that many transpots are indeed very similar (*e.g.* `conforme à` and `conformes à`).

| | |
|---|---|
| `conforme à` (45) | `conformément à` (29) |
| `à`$^\star$ (21) | `dans`$^\star$ (20) |
| ... | |
| `conforme au` (12) | `conformes à` (11) |
| `avec`$^\star$ (9) | *`conformément`* (9) |
| ... | |
| `correspond à` (1) | `respectent` (1) |
| `d'actualité`$^\star$ (1) | `gestes en`$^\star$ (1) |

Figure 3: Subset of the 273 different transpots retrieved for the query `in keeping with`. Their frequency is shown in parentheses.

Since in `TS3` we want to offer the user a list of retrieved translations for a query, strategies must be devised for bypassing alignment errors and delivering as many as possible translation variants to the user. We investigated two avenues in this study: detecting erroneous transpots (Section 3) and merging together variants of the same canonical translation (Section 4).

## 3 Refining Transpotting

We investigated the learning of classifiers trained to distinguish good transpots from bad ones. We tried several popular classifiers:[2] a *voted-perceptron* algorithm (Freund and Schapire, 1999) which has been reported to work well in a number of NLP tasks (Collins, 2002); a *support vector machine* (SVM), commonly used in supervised learning (Cristianini and Shawe-Taylor, 2000); a *decision stump*, a very simple one-level decision tree; *AdaBoost* using a decision stump as weak classifier (Freund and Schapire, 1996); and a *majority voting* classifier between a voted-perceptron, an SVM and AdaBoost (Kittler et al., 1998).

Each classifier was trained in a supervised way thanks to an annotated corpus we devised (see Section 5). We computed three groups of features for each example, that is, each query/transpot pair $(q, t)$. The first group is made up of features related to the size (counted in words) of $q$ and $t$, with the intuition that they should be related. The second group gathers various alignment scores computed with word-alignment models (min and max likelihood values, etc.). The last group gathers clues that are more linguistically flavored, among which the ratio of grammatical words in $q$ and $t$, or the number of prepositions and articles. In total, each example is represented by at most 40 numerical features.

## 4 Merging Variants

Once erroneous transpots have been filtered out, there usually remain many translations for a given query. For instance, the best classifier we trained identified 91 bad transpots among the 273 candidat ones. Among the remaining transpots, some of them are very similar and are therefore redundant for the user (see Figure 3). This phenomenon is particularly acute for the French language with its numerous conjugated forms for verbs. Another problem that often shows up is that many transpots differ only by punctuation marks or by a few grammatical words.

Of the 182 transpots surviving the filter, we estimate that no less than 37 interesting *canonical* translations exist for the query `in keeping with`. Therefore, it is important from the user perspective to identify them. We investigated ways to merge together close variants. This raises several difficulties. First, the transpots must be compared together, which represents both a tricky and time consuming process. Second, we need to identify groups of similar variants. We describe our solutions to these problems in the sequel.

[2]We used Weka in our experiments `www.cs.waikato.ac.nz/ml/weka`

### 4.1 Word-Based Edit Distance

A word-level specific edit distance was empirically developed to meet the constraints of our application. Different substitution, deletion and insertion costs are set according to the grammatical classes or possible inflections of the words; it is therefore language dependent. We used an in-house lexicon that lists, for both French and English, the lemmas of each inflected form and its possible parts-of-speech.

A minimal substitution cost was empirically given between two inflected forms of the same lemma. A score has been engineered which increasingly penalizes in that order edit operations involving punctuation marks, articles, grammatical words (prepositions, conjunctions and pronouns), auxiliary verbs and lexical words (verbs, nouns, adjectives and adverbs).

### 4.2 Neighbor-Joining Algorithm

Comparing the transpots pairwise with the distance we defined is an instance of multiple sequence alignment, a well studied problem in bioinformatics (Chenna et al., 2003). We adopted the approach of progressive alignment construction. This method first computes the word-based edit-distance between every pair of transpots and stores the results in an edit-matrix. Second, a greedy bottom-up clustering method called *neighbor-joining* (Saiou and Nei, 1987) is conducted; it builds a tree by joining together either two transpots, that is two leaves of the tree, or a transpot and a node in the tree already aggregating several translations. At each step, the most similar pair is merged and added to the tree, until no transpot remains unaligned.

Finally, the neighbor-joining algorithm returns a tree whose leaves are transpots. Closest leaves in this tree correspond to the most similar variants. Therefore, clusters of variants can be formed by traversing the tree in a post-order manner. The transpots associated with two neighboring leaves and which differ only by grammatical words or by inflectional variants are considered as sufficiently similar to be merged into a single cluster. This process is repeated until all the leaves have been compared with their nearest neighbor and no more similar variants remain.

Figure 4 illustrates this process. The two neighbor transpots `conforme à` and `conformes à` are first grouped together, so are `conforme au` and `conforme aux`. Then, those two groups are merged into a single cluster. The transpot `correspondant à` being too different is not aggregated into this cluster.


Figure 4: Merging of close transpots.

### 4.3 Naive Joining Algorithm

We also implemented a conceptually simpler merging algorithm which relies on the frequencies of transpots. The algorithm compares the most frequent variant with all the other ones. Those that are close enough (according to our distance) are aggregated into a cluster. This process is iteratively applied on the remaining variants until no more cluster can be formed.

## 5 Corpora

### 5.1 Translation Memory

The largest collections in `TransSearch` come from the Canadian Hansards, that is, parallel texts in English and French drawn from official records of the proceedings of the Canadian Parliament. For our experiments, we indexed with Lucene[3] a TM comprising 3.3 million pairs of French-English sentences aligned at the sentence-level by an in-house aligner. This was the maximum amount of material we could train a statistical word-alignment model on, running the `giza++` (Och and Ney, 2003) toolkit on a computer equipped with 16 gigabytes of memory.

### 5.2 Automatic Reference Corpus

We developed a reference corpus (REF) by intersecting our TM with a bilingual lexicon, and some user queries. We used an in-house bilingual-phrase lexicon we collected over various projects, which includes 59 057 English phrases with an average of 1.4 French translations each. We extracted from the logs of `TransSearch` the 5 000 most frequent queries submitted by users to the system. 4 526 of those queries actually occurred in our TM, and of these, 2 130 had a sanctioned translation in our bilingual lexicon. We collected up to 5 000 pairs of sentences for each of those 2 130 queries,

[3] `http://lucene.apache.org`

leading to a set of 1 102 357 pairs of sentences, with an average of 517 pairs of sentences per query. For each of the 2 130 queries, the bilingual lexicon enabled us to extract a mean of 3.5 different transpots (and a maximum of 37). This results in a set of 7 472 different pairs of query/translation.

### 5.3 Human Reference

In order to train the classifiers described in Section 3, four human annotators were asked to identify bad transpots among those proposed by our transpotting algorithm. We decided to annotate the query/transpot pairs without their contexts of occurrence, which allows a relatively fast annotation process,[4] but leaves some cases difficult to annotate. For instance, in our running example, a transpot such as `conforme à` is straightforward to annotate, but others such as `dans le sens de` or `tenir compte de` gave annotators a harder time since both can be valid translations in some contexts. We ended up with a set of 531 queries that have an average of 22.9 transpots each, for a total of 12 144 annotated examples. We computed the inter-annotator agreement and observed a 0.76 kappa score, which indicates a high degree of agreement.

## 6 Experiments

### 6.1 Transpotting

For each of the 1 102 357 pairs of sentences of REF, we evaluated the ability of the transpotting algorithm described in Section 2.2 to find the reference translation $\hat{t}$ for the query $q$, according to recall and precision ratios computed as follows:

$$\text{recall} = |t \cap \hat{t}|/|\hat{t}| \quad \text{precision} = |t \cap \hat{t}|/|t|$$
$$\text{F-measure} = 2 \times |t \cap \hat{t}|/(|t| + |\hat{t}|)$$

where $t$ is the transpot identified by the algorithm, and the intersection operation is to be understood as the portion of words shared by $t$ and $\hat{t}$. A point of detail is in order here: since several pairs of sentences often contain the same given query/reference translation pair $(q, \hat{t})$, we first average for a given pair the ratios measured for all the occurrences of that pair in the reference corpus. Then, we average the scores over the set of all different pairs $(q, \hat{t})$ in the corpus. This avoids biasing our evaluation metrics toward frequent pairs in the REF corpus.[5]

[4] On the order of 40 seconds per query.

[5] Without this normalization, results would be increased by a range of 0.2 to 0.4 points.

| | prec. | rec. | F-meas. |
|---|---|---|---|
| transpotting | 0.30 | 0.60 | 0.38 |
| transpotting + voting | 0.37 | 0.76 | 0.46 |

Table 1: Transpotting results before and after filtering (REF). See next section for an explanation of line 2.

Our transpotting algorithm (see line 1 of Table 1) achieves a precision of 0.30, and a recall of 0.60. At a first glance, these figures might seem rather low. However, recall that our normalization prevents frequent queries that are often correctly aligned from being counted several times. Thus, this reinforces the score measured for infrequent queries, which in turn tend to be worse aligned. Also, we observed that very often the reference translation is a subset of the transpot found, which lowers precision. This is the case of the example shown in Figure 5.

`Une telle restriction ne s' inscrit pas` **`dans le sens des`** `pratiques actuelles.`

Figure 5: Transpot (underlined) and reference translation (in bold) for the query `in keeping with`.

### 6.2 Training Classifiers

As described in Section 3, we trained various classifiers to identify spurious transpots, representing an example (a query/transpot pair) by three kinds of feature sets. All these variants plus a few challenging baselines are evaluated according to the ratio of Correctly Classified Instances (CCI). Since in our application we are interested in filtering out bad transpots, precision, recall and F-measure rates related to this class are computed as well.

We report in Table 2 the figures we measured by a 10-fold stratified cross-validation procedure. To begin with, the simplest baseline we built (line 1) classifies all instances as good. This results in a useless filter with a CCI ratio of 0.62. A more sensible baseline —that we engineered after we investigated the usefulness of different feature sets— classifies as bad the transpots whose ratio of grammatical words is above 0.75. It is associated with a CCI ratio of 0.78 (line 2).

We started by investigating the voted-perceptron

| | | | Bad | | |
|---|---|---|---|---|---|
| Classifier | Features | CCI | precision | recall | F-measure |
| Baseline: all good | | 0.62 | 0.00 | 0.00 | 0.00 |
| Baseline: grammatical ratio $> 0.75$ | | **0.78** | 0.88 | 0.49 | **0.63** |
| Voted-Perceptron (VP) | size | 0.73 | 0.75 | 0.47 | 0.58 |
| | IBM | 0.78 | 0.69 | 0.78 | 0.73 |
| | grammatical | 0.79 | 0.88 | 0.52 | 0.65 |
| | all | 0.83 | 0.81 | 0.73 | 0.77 |
| SVM | all | 0.83 | 0.84 | 0.70 | 0.76 |
| Decision Stump | | 0.81 | 0.77 | 0.70 | 0.74 |
| AdaBoost | | 0.83 | 0.71 | 0.83 | 0.76 |
| Majority-Voting (VP+SVM+AdaBoost) | | **0.84** | 0.84 | 0.71 | **0.77** |

Table 2: Performance of different algorithms for identifying bad transpots.

and the contribution of each feature sets on its performance.[6] When the voted-perceptron is trained using only one set of features, the one making use of the grammatical features obtains the best CCI ratio of $0.79$ and an F-measure of $0.65$. Even if the configuration based on IBM model 2 word alignment scores obtains a slightly inferior CCI ratio of $0.78$, it has a much higher F-measure of $0.73$ and can be considered as the best feature set. When using all feature sets, the voted-perceptron clearly surpasses the baseline with a CCI of $0.83$ and an F-measure of $0.77$. It should be noticed that while the best baseline has a better precision than the best voted-perceptron, precision and recall are more balanced for the latter. Because it is not clear whether precision or recall should be favored for the task of bad transpot filtering, optimizing the F-measure is preferred.

When training the other classifiers using all feature sets, no significant gain can be observed. Nevertheless the majority-voting classifier obtains the best CCI ratio of $0.84$ and an F-measure of $0.77$. The figures obtained by the decision stump, a one-level decision tree, are surprisingly high. The rule used by this classifier considers the minimal word alignment probability inside a Viterbi alignment based on an IBM model 2. At the very least, this confirms the interest of this feature set.

Once the best classifier had been obtained, *i.e.* majority-voting, we evaluated the impact of transpotting filtering against the REF corpus. Results are shown in Table 1 (line 2). We observe a significant gain in F-measure which increases from $0.38$ to $0.46$. The higher gain is in recall, from $0.60$ to $0.76$. Referring to the example of Section 6.1, this means that filtering eliminates too short transpots. Inspections revealed that short bad transpots, such as grammatical words, are frequently identified as bad by the classifier. This demonstrates the interest of filtering bad transpots.

### 6.3 Merging Variants

The interest of grouping together similar variants is clear from a user perspective. However, the granularity with which we should aggregate variants is not obvious.[7] We studied two approaches. The first method aims at grouping together transpots that differ by punctuation marks or that are inflectional variants of the same lemma. It is based on an edit distance, called $D_1$, which uses the same edit-costs for grammatical and lexical words. The second method groups together variants with looser constraints. It resorts to an edit distance, named $D_2$, that associates lower edit-costs with grammatical words than with lexical words.

From the transpots obtained for the 5 000 queries of the REF corpus (and filtered by our best classifier), this method leads to an average of 69 clusters per query (Table 3, columns 2 and 4), whereas there are on average 85 unique transpots per query (Table 3, column 1). The same level of grouping is observed for the two joining algorithms described in Section 4.3.

As expected, the use of $D_2$ dramatically reduces the number of clusters to an average of 45 per query (Table 3, columns 3 and 5). Contrary to $D_1$, $D_2$ allows the merging process to gather similar variants such as `sur des années` and

[6] Similar results concerning the different feature set have been observed for the other classifiers and are not presented here.

[7] This would certainly require tests with real users.

| baseline | naive joining | | neighbor-joining | |
|---|---|---|---|---|
| | $D_1$ | $D_2$ | $D_1$ | $D_2$ |
| 85 | 69 | 45 | 69 | 45 |

Table 3: Average number of responses per query.

durant des années. However, it occasionally leads to erroneous groupings such as tout à fait (fully) and fait tout (do everything).

In what follows, we measure quantitatively the improvement from the point of view of the quality of the first responses suggested for each query.

**Experimental Setup** Table 4 shows the 5 most frequent transpots computed for two queries by the original transpotting algorithm (baseline) and obtained after grouping together variants (with the naive joining method). We observe the tendency of the baseline to propose inflectional variants of the same translation, while merging variants leads to more diversity, which is preferable since the number of variants that can be displayed in TransSearch without scrolling is limited. Indeed, we think that presenting the user with around 5 transpots and some sentences where they occurred is a good compromise (see Figure 2).

In order to simulate this, we measure in what follows the diversity of the best 5 transpots proposed by different methods. The baseline keeps the 5 most frequent transpots as returned by our transpotting algorithm, while the other methods allow for clustering the transpots. The 5 most frequent clusters are considered,[8] and the most frequent variant in each cluster is retained. Therefore each method delivers at most 5 transpots.

The best 5 transpots are considered as bags of unigrams, bigrams or trigrams and compared to reference translations turned also in bags of $n$-grams. All the words are lemmatized, and short words (less than 4 characters) are discarded as a proxy to remove grammatical words. For instance, the transpots returned by the baseline method in Table 4 for the first query are turned into {décrire, comme}.

**Results** The comparison of the generated bags-of-words with the reference ones is done by computing precision and recall. The reference used here is the resource described in Section 5.3. Table 5 reports results obtained with the metrics based on bags of $n$-grams without joining variants (line 1) and when using either the neighbor-joining algorithm (lines 2 and 3) or the naive method (lines 4 and 5). Their comparison shows an improvement in terms of F-measure for unigrams, bigrams and trigrams when variants are merged. If the precision slightly decreases for unigrams w.r.t. the baseline, a significant improvement is obtained especially with the edit distance $D_2$. These results are correlated with the more diversified translations obtained when variants are grouped together.

[8]The frequency of a cluster is the cumulative frequency of all the variants it groups.

## 7 Discussion

In this study, we have investigated the use of statistical word-alignment for improving the commercial concordancer TransSearch. A transpotting algorithm has been proposed and evaluated. We discussed two novel issues that are essential to the success of our new prototype: detecting erroneous transpots, and grouping together similar variants. We proposed our solutions to these two problems and evaluated their efficiency. In particular, we demonstrated that it is possible to detect erroneous transpots better than a fair baseline, and that merging variants leads to transpots of better diversity.

For the time being, it is difficult to compare our results to others in the community. This is principally due to the uniqueness of the TransSearch system, which archives a huge TM. To give a point of comparison, in (Callisson-Burch et al., 2005) the authors report alignment results they obtained for 120 selected queries and a TM of 50 000 pairs of sentences. This is several orders of magnitude smaller than the experiments we conducted in this study.

There are several issues we are currently investigating. First, we only considered simple word-alignment models in this study. Higher-level IBM models can potentially improve the quality of the word alignments produced. At the very least, HMM models (Vogel et al., 1996), for which Viterbi alignments can be computed efficiently, should be considered. The alignment method used in current phrase-based SMT is another alternative we are considering.

## Acknowledgements

This research is being funded by an NSERC grant in collaboration with Terminotix.[9]

[9]www.terminotix.com

| | | | | | |
|---|---|---|---|---|---|
| baseline | décrits | décrite | décrit | tel que décrit | comme l'a |
| naive joining $D_2$ | décrits | prévu | comme l'a | tel que prescrit | comme le propose |
| baseline | s'est révélé | s'est avéré | s'est avérée | s'est révélée | a été |
| naive joining $D_2$ | s'est révélé | s'est avéré | a été | s'est montré | a prouvé |

Table 4: 5 most frequent responses for the queries `as described` and `has proven to be` when a joining method is used or not.

| | | unigrams | | | bigrams | | | trigrams | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | prec. | rec. | FM | prec. | rec. | FM | prec. | rec. | FM |
| baseline | | 0.93 | 0.45 | 0.61 | 0.82 | 0.35 | 0.49 | 0.68 | 0.30 | 0.41 |
| naive joining | $D_1$ | 0.93 | 0.51 | 0.65 | 0.86 | 0.40 | **0.55** | 0.72 | 0.33 | 0.45 |
| | $D_2$ | 0.90 | 0.57 | **0.69** | 0.79 | 0.40 | 0.53 | 0.72 | 0.33 | 0.45 |
| neighbor-joining | $D_1$ | 0.93 | 0.50 | 0.65 | 0.86 | 0.41 | **0.55** | 0.72 | 0.34 | **0.46** |
| | $D_2$ | 0.90 | 0.56 | **0.69** | 0.80 | 0.40 | 0.53 | 0.71 | 0.34 | **0.46** |

Table 5: Evaluation of quality of the variants merging process for the 5 most frequent groups retrieved for 531 queries.

## References


Bourdaillet, J., S. Huet, F. Gotti, G. Lapalme, and P. Langlais. 2009. Enhancing the bilingual concordancer TransSearch with word-level alignment. In *22nd Conference of the Canadian Society for Computational Studies of Intelligence*, Kelowna, Canada.

Brown, P., V. Della Pietra, S. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Callisson-Burch, C., C. Bannard, and J. Schroeder. 2005. A compact data structure for searchable translation memories. In *10th European Conference of the Association for Machine Translation (EAMT)*, pages 59–65, Budapest, Hungary.

Chenna, R., H. Sugawara, T. Koike, R. Lopez, T. J. Gibson, D. G. Higgins, and J. D. Thompson. 2003. Multiple sequence alignment with the Clustal series of programs. *Nucleic Acids Research*, 31(13):3497–3500.

Collins, M. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8, Philadelphia, PA, USA.

Cristianini, N. and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press.

Freund, Y. and R. Schapire. 1996. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning (ICML)*, pages 148–156, Bari, Italy.

Freund, Y. and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Kittler, J., M. Hatef, R. P.W. Duin, and J. Matas. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Macklovitch, E., G. Lapalme, and F. Gotti. 2008. TransSearch: What are translators looking for? In *18th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 412–419, Waikiki, Hawai'i, USA.

Och, F. J. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Saiou, N. and M. Nei. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.

Simard, M. 2003. Translation spotting for translation memories. In *HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and beyond*, pages 65–72, Edmonton, Canada.

Venugopal, A., S. Vogel, and A. Waibel. 2003. Effective phrase translation extraction from alignment models. In *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 319–326, Sapporo, Japan.

Vogel, S., H. Ney, and Tillmann C. 1996. HMM-based word alignment in statistical translation. In *16th Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark.