3GTM: A Third-Generation Translation Memory

Fabrizio Gotti †, Philippe Langlais †, Elliott Macklovitch †,
Didier Bourigault ⋆, Benoit Robichaud ‡, Claude Coulombe ‡

† RALI/DIRO
Succ. Centre-Ville
H3C 3J7 Montréal
Canada
www-rali.umontreal.ca

‡ Lingua Technologies Inc.
Côtes-des-Neiges Road
H3T 2A9 Montréal
Canada
w3.linguatechnologies.com

⋆ERSS-CNRS-Toulouse
Maison de la Recherche
F-31058 Toulouse Cedex 1
France

Abstract

We report experiments we conducted in the framework of a sub-sentential, third-generation translation memory. We investigate various units that can be considered to build the memory, and present different ways of measuring their coverage on a test corpus.

1 Introduction

A *translation memory* (TM) is a computer-assisted translation tool that allows translators to increase their productivity by making it easy for them to recycle parts of past translations. The TM works by automatically comparing each successive sentence in a new text to be translated with the source-text sentences in the database. If a matching sentence is found, the TM presents the translator with the aligned target sentence, which she may then incorporate into her new translation, edit as needed, or simply ignore.

Among professional translators, TM's are currently among the most popular CAT tools in use, for a number of reasons. For one thing, TM's guarantee a certain level of quality, seeing that they recycle translations generally done by humans (as opposed to machine translation systems). For another, they allow for non-negligible productivity gains, particularly on highly repetitive texts. The problem, however, is that such texts account for only a small portion of the overall demand for high-quality translation. On the great majority of texts that need to be translated, full-sentence repetition is a rather rare phenomenon. And this is a defining characteristic of the first generation of TM systems: not only do they store pairs of complete sentences, but the repetitions they search for are also on the level of the full sentence. In response to this problem, TM developers have designed second generation TM systems, where two source sentences may be considered functionally identical if they differ only slightly (with regard to name entities or edit distance operations). Nevertheless, the improvements in recall rates brought about by fuzzy matching remain quite modest.

Lingua Technologies Inc., the RALI Laboratory at Université de Montréal and Transetix Inc. are currently involved in a project funded by Precarn to develop a *third-generation TM* (3GTM), the defining characteristic of which will be its ability to search for repetitions on a sub-sentential level. In the event that the database does not contain a copy of the current source sentence, our 3GTM will segment that sentence into chunks, which will be searched for in the memory; and once a match of such a chunk is found, it will use a state-of-the-art word alignment techniques to locate and retrieve the corresponding chunk in the aligned target sentence.

Somewhat paradoxically, the principal problem faced by such a system will likely be an overabundance of responses —unlike 2GTM's, where the principal problem is silence. To help resolve this problem, our TM will use context-sensitive filtering techniques in order to prioritize the results and ensure that the user is not drowned by an unmanageable quantity of information.

If organizing a memory at the sentence level is rather simple, building a memory exploiting sub-sentential units is another kettle of fish. Although our 3GTM is still incomplete, the aim of this study is to present the results of the experiments we conducted during the first stages of the project. We tested translation memories built with

different base units and different technologies. To evaluate their quality, we used coverage metrics on various different levels of organization.

This paper is organized as follows. We describe in Section 2 the corpora used in this study and give a rationale for using coverage metrics. Next, we present the experiments we made by querying the memory with arbitrary sequences of words. In Section 4, we report the experiments we conducted with a syntactic parser. We then explain the coverage experiments we made with a dependency-based parser (Section 5) to build a special kind of translation memory. We then conclude our work in Section 6.

2 Experimental Set-up

Our experiments were conducted exclusively on the English-French language pair. For each experiment, we built a database (translation memory) from the same parallel corpus and queried it with the same French test corpus to obtain coverage statistics.

2.1 The translation memory

To populate the translation memory, we used 1.7M pairs of sentences (one being the translation of the other) from the Canadian Hansard Corpus consisting of debates from the Canadian Parliament,[1] produced over the period 1986-1994. This corpus will be referred to as the training corpus in the rest of this article. Some statistics regarding the corpus are presented in Table 1.

Language	English	French
Nb. sentences	1 753 443	1 753 443
Nb. tokens	31 637 775	34 150 039
Nb. types	85 810	106 987
Avg. word/sent.	17.5	19.3

Table 1: Number of sentences, tokens and types in the training corpus.

The main database for the experiments described in Section 3 and Section 4 was built using LUCENE, a very efficient, full-featured text search engine. It is a JAVA library, freely available from Apache Jakarta.[2]

We indexed the full training corpus described earlier, a task which took roughly 40 minutes on a good desktop computer. The resulting index occupied about 360MB of disk space. During the indexing process, we used a simple tokenizer to process the sentences that were fed to LUCENE. This tokenizer (a `WhitespaceAnalyzer`), divides text at whitespace and does not remove stop words from the index. It is not language-specific.

In addition, the full training corpus was aligned at the word level by the method described in (Simard and Langlais, 2003) which recursively splits in two parts both the source and target sentences and allows either a left-to-right alignment (the first part of the source sentence is aligned to the first part of the target sentence, the second parts are aligned together), or an inverted one (the first source part is aligned to the second target one and vice-versa). The best split found at each step is kept and we further split the two parts until we cannot split anymore (that is, when there is at most one token on one side). The computation of the quality of a split is done using a linear combination of two word models (one for each direction) that have been trained on the same training corpus. We used an IBM model 2 (Brown et al., 1993) for that purpose, with parameters trained using the GIZA package (Och and Ney, 2000).

2.2 Test corpus

Our goal is to evaluate how useful different internal units are in practice in the translation memory. In order to measure this, we extracted from the Hansards mentioned earlier a test corpus of 1000 pairs of sentences on which we computed coverage statistics. This test corpus is chronologically distinct from the training corpus and disjoint from the training corpus. Whenever we queried the database, we did so in French (and obtained English material).

2.3 Coverage statistics

Evaluating the quality of the different systems we present in this paper is a challenging task. To truly assess the usefulness of computer-aided translation software, one would wish to submit the system to the evaluation of human translators. How-

[1] `www.parl.gc.ca/cgi-bin/hansard/e_hansard_master.pl`

[2] `lucene.apache.org`

ever, it was not possible in our case, at least at the stage of development the project had reached. We therefore resorted to compute coverage statistics on a test bitext.

Evaluating the coverage on a test corpus can provide us with interesting data. It gives us an idea of the number of translation units the system would be able to find for a sentence to be translated. Moreover, by computing the coverage of the target (reference) sentence with the target material associated with the source units found in the previous step, we get a sense of how meaningful the associations stored in the memory are.

3 Sentence and Word Coverage

3.1 Sentence coverage

Our coverage experiments consisted in systematic queries fed to our LUCENE index using the test corpus to build these queries.

We were first interested in the number of sentences from the test (French) corpus found verbatim in the memory. This simple experiment gave the expected results: a relatively poor coverage, as can be seen on line 2 in Table 2. Only 14.8% of the sentences were covered. This illustrates what we said in the introduction: full-sentence repetition is a rare phenomenon. However, this coverage figure is not entirely mediocre, and is mainly caused by recurring idioms in the Hansards, e.g., in their English form, `I don't know.` or `Mr. Speaker: Order, please.`, which are nonetheless useful to the human translator. It is worth noting though that these sentences found verbatim are on average half as long as those contained in the train or test corpus.

Nb. of sentences	1000
Nb. of sent. found verbatim	148
Avg. size of sent. in test corpus	19.2
Avg. size of sent. found verbatim	11.1

Table 2: Coverage statistics on the source sentences of the test corpus using verbatim match.

3.2 Word coverage

Next, we turned our attention to a sub-sentential word coverage. Again, using our test corpus, we submitted a series of queries to the translation memory. For each source (French) sentence, we found every subsequence of length greater or equal to 2 words and looked them up in the memory. A substring found at least once is considered valid. Then, we computed a source coverage out of all these valid substrings using a dynamic programming algorithm which seeks to maximize the source coverage (in words) while minimizing the number of substrings covering the sentence. No substrings were allowed to overlap in the optimal coverage. We call such a coverage "optimal".

With this source coverage in hand, we then computed a target coverage. To do so, we extract the target (English) material associated with any source substring. This is done by tracking the corresponding beginning and end target word positions in each target sentence, following the word alignment. The example in Figure 1 illustrates the process.

S: `Charlie`$_1$ `et`$_2$ `[la`$_3$ `chocolaterie`$_{4,5}$`]`
T: `Charlie and [the chocolate factory]`
q: `la chocolaterie`
m: `the chocolate factory`

Figure 1: Extraction of target material given a pair of source (S) and target (T) sentences in the memory and the word alignment (word indices). The query (q) corresponds to the target material (m) (target indices 3 to 5).

To compute the target coverage, we try to match the target (English) substrings found this way to the target sentence in the test corpus. If an entire target substring doesn't match the target sentence, the we keep shortening it by one word until it does. To continue the previous example, let's say the target sentence for which we are computing the coverage is `He works at a chocolate factory .`. Since the target substring extracted from the previous step (`the chocolate factory`) doesn't match the target sentence because of the mismatch `the`$\neq$`a`, we shorten it to `chocolate factory` and cover two words instead of three.

Naturally, this method is biased and will overestimate the target coverage, but our goal here is not to produce a system that will automatically be able to cover/reconstruct a target sentence given

a source sentence. We are simply trying to assess how much target material a human translator might extract, in the best of circumstances, from the system given a request she makes to the memory.

Once all these target substrings have been extracted, we compute another optimal coverage on the target sentence using the same dynamic programming algorithm described previously.

Metric	Source	Target
Optimal coverage	93.9%	60.3%
Cov. unit size (words)	4.06	3.17
Number of cov. units	4.44	3.33
Avg. nb. LUCENE queries per sentence: 207.5		

Table 3: Coverage statistics of the test corpora by querying LUCENE with all substrings of length $\geq$ 2 words on the source (French) side, for each sentence. The figures presentended are averages over all 1000 pairs of sentences.

The results of this word-based approach are presented in Table 3. Whereas the first experiment yielded a poor source coverage, this one covers, on average, 93.9% of a source sentence. The problem, as we said in the introduction, is that we are faced with an overabundance of material. Indeed, with an average of 207.5 queries for any given source sentence, selecting the best valid target substrings is both difficult and crucial for the end user. Add to this the fact that each valid substring covering the source sentence typically has many target substring counterparts. Offering all these alternatives in a typical computer-aided translation system would overwhelm the user and render it useless; a problem discussed in Langlais and Simard (2003).

We also observe that 60.3% of the target sentence is covered, which is encouraging: the associations we are able to produce with the training corpus as well as the word alignments are indeed relevant, since we are able to cover the target sentence, which acts as a translation reference.

4 Chunk-Based Coverage

In the very probable case that our translation memory does not contain the whole source sentence we are looking for, our system will segment that sentence into a certain number of chunks, which it would then search in the memory. This will limit the number of requests (speeding up the system) and hopefully avoid overwhelming the end user with too much data.

Moreover, it is our intuition that choosing linguistically motivated chunks to query the translation database may be very beneficial to the system. On the one hand, it could improve the quality of the extraction of target material given source material and a word alignment: this step could indeed be further refined by taking into account not only the word alignments but also chunk boundaries computed on both sides of the training corpus.

On the other hand, querying with source chunks (and proposing target chunks) could help the translator during sentence reconstruction. She would notice few overlaps in the proposed material, and could accept several contiguous target material units without the need for heavy editing.

4.1 Constituent-based parser

Our commercial partner Lingua Technologies Inc. provided us with their syntactic parser to do the chunking. Their morphosyntactic analysis technology GRAMMATICUM was designed in the early 1990s (Coulombe, 1991) to produce grammar checkers. This efficient hand-crafted rule-based parser works with English and French texts.

4.2 Experimental results

To compute coverage statistics, we proceeded in a manner similar to the one described in Section 3. However, this time, instead of considering random-length substrings, we used chunks to cover the source sentences from the 1000-sentence test bitext.

Each source (French) sentence from the test corpus was first chunked using the tool from Lingua Technologies Inc. On average, 28.35 chunks were found for each source sentence (the same word can be part of several chunks). Only the chunks of size greater than 1 word were kept; there are 11.7 of these per source sentence on average. These source chunks were then used to query the database created by LUCENE. We then proceeded like we described in the previous section to compute coverage statistics.

Our findings are presented in Table 4. The source coverage is well below the one observed in the previous section since the number of potential chunks that can cover a source sentence is much smaller than all the possible substrings of a source sentence. The target coverage, however, does not seem to have suffered: it remains unchanged at around 60%. This is very encouraging: although we restrict very significantly the number of queries made to our database (compare 207.5 queries on average in Table 3 with only 11.7 queries here), the target material proposed to the end user seems to be sufficient to cover a large part of the reference translation.

This seems to argue in favor of our chunk-based approach to a translation memory also because we reduce the risk of flooding the translator with too much target material.

Metric	Source	Target
Optimal coverage	59.9%	59.3%
Cov. unit size (words)	3.73	2.99
Number of cov. units	3.08	3.47
Avg. nb. LUCENE queries per sentence: 11.7		

Table 4: Coverage statistics of the test corpora by querying LUCENE with chunks of length ≥ 2 words from the source (French) side, for each sentence. The figures presentended are averages over all 1000 pairs of sentences.

5 Tree-Phrase Coverage

Up to this point, we have only considered contiguous units as queries. This does not necessarily reflect the way an innovative translation memory should work. Indeed, a repository could, provided its design is adapted, benefit from syntactic information when it is queried. Such a system could for instance recognize that the query `a good friend` matches the sentence `a very good friend`, fed to it during the creation of its database, because it "knows" that the intervening word `very` can be skipped for the match.

Therefore, we wish to draw on a previous study in the context of example-based machine translation (Langlais et al., 2005) which considers a new kind of unit: a Tree-Phrase (TP), a combination of a treelet and a non-contiguous phrase.

Several authors have used treelets as prime units (Gildea, 2003; Ding and Palmer, 2004; Quirk et al., 2005), but mostly with the idea of projecting a source treelet into its target counterpart, and in the context of producing a translation.

Here, we do not address the issue of projecting a treelet into a target one, but take the bet that collecting (without structure) the target words associated to the words encoded in the nodes of a treelet will suffice to handle the task at hand. We call this set of target words an Elastic Phrase (EP). An elastic phrase is not only possibly a non-contiguous sequence of words, but also has the characteristic of having "gaps" of arbitrary size, which is not the case for the phrases considered by Simard et al. (2005).

The objective of our TP approach is to show whether a memory populated with TP can offer coverage advantages over one built only from contiguous units. With the goal of answering this question, we first parsed the French material of the Hansards training corpus with a dependency parser called SYNTEX (Bourigault and Fabre, 2000) (see Section 5.1).

We collected from this parsed material a set of depth-one treelets that we associated with their target EP's, using the word alignment we computed offline. The programs for this part of the study were written in C++, rather than JAVA and are independent of LUCENE.

5.1 Syntex

SYNTEX (Bourigault and Fabre, 2000) is a robust and efficient syntactic parser allowing the identification of syntactic dependency relations between words. Currently, the main relation types identified by this tool are subject, direct object, prepositional complement, adjectival modifier, and subordination. Each dependency relation identifies two words: one that acts as a governor, and another one that is its dependent. An English and a French version are available.

For example, given the French source sentence "on demande des crédits fédéraux" (request for federal funding), SYNTEX outputs several dependency links that we can represent by the structure in Figure 2, where a root node contains the word governing the words of all its child nodes, which are called its dependents. The number of child

nodes is arbitrary, and is not limited to 2.

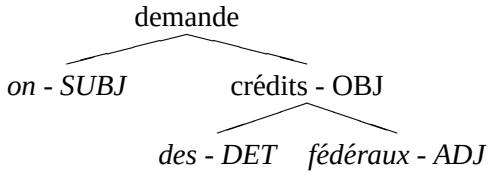

Figure 2: Parse of the sentence "on demande des crédits fédéraux" (request for federal funding).

5.2 The TP Memory

We parsed with SYNTEX the 1.7M French sentences of our training bitext. From this material we extracted all dependency subtrees of depth 1 from the complete dependency trees found by SYNTEX. For instance, the two treelets in Figure 3 will be collected out of the parse tree of Figure 2. To find the target words associated to nodes of the treelets, we used the word alignment information we computed, as we described in Section 2.1.

alignment: demande ≡ request for — fédéraux ≡ federal — crédits ≡ funding

treelets:

tree-phrases:

TL* {{on@-1} demande {crédits@2}}
EP* |request@0||for@1||funding@3|

TL {{des@-1} crédits {fédéraux@1}}
EP |federal@0||funding@1|

Figure 3: The Tree-Phrases collected out of the SYNTEX parse for the sentence pair of Figure 2. Non-contiguous structures are marked by a star.

An illustration of the output of this alignment procedure is provided for the running example in Figure 3. Once both the word alignment and the treelets are computed, populating the memory with tree-phrases is just a matter of collecting them, and keeping their count over the total training corpus. The format we use for representing the treelets (see Figure 3) is similar to the one proposed in (Quirk et al., 2005): the left and right dependents of a given governor word are listed in order in two separate lists along with their respective offsets (the governor/root token always has the default offset 0). An elastic phrase is simply the list of tokens aligned to the words of the corresponding treelet as well as the respective offsets at which they were found in the target sentence, relative to the first token position. Note that TLs as well as EPs might not be contiguous as is for instance the case for the first pair of structures listed in Figure 3.

The tree-phrases (TPs) are stored in a database loaded into memory. Out of 1.7M pairs of sentences, we collected more than 3 million different kinds of TLs from which we projected 6.5 million different kinds of EPs. The treelets range in size from 2 to 8 and the phrases, from 1 to 9. Slightly less than half of the treelets are contiguous ones (that is, involving a sequence of adjacent words); 40% of the EPs are contiguous. When the respective frequency of each TL or EP is factored in, we have roughly 11 million TLs and 10 million EPs.

With this TP memory available, we computed once again coverage statistics on the test corpus.

5.3 Coverage analysis

Match methods A match between a treelet or an elastic phrase and a sentence is different than a match between a simple substring an a sentence (like in the previous sections). Here, we say that a treelet matches a source sentence if all its tokens are found in the source sentence and they are in the same order (determined by each node's offset).

When a treelet matches, its corresponding phrases are retrieved from the memory and matched against the target (English) sentence. We say we have a match if the tokens of the phrase are encountered in the same order in the target sentence. However, we added another constraint to the target match. Indeed, this method allows the tokens of a phrase that are only separated by, say, 2 tokens to match a sentence where they become separated by 18 tokens. This goes against our intuition that the word gaps in non-contiguous phrases must not be stretched beyond a certain limit. We therefore added a constraint to the match, by which we limit the "elasticity" of those

gaps to a maximum of 3 times their original size.

Source coverage We first find an optimal source coverage, then find the corresponding target material and find the optimal (target) coverage. The idea behind the source coverage algorithm is to select the minimum number of TLs covering as much as possible of the source sentence. We first find all the TLs matching the source sentence, then find the optimal decomposition of the source sentence with these TLs.

Conceptually, the algorithm builds the set of all the *valid* hypotheses that match the source sentence S. A valid hypothesis is a set of treelets that (at least) partially covers S and satisfies a certain number of properties, the main one being that none of the dependencies captured in the set of TLs is allowed to cross another one. Once all such hypotheses are built, the algorithm picks the one with the best score. In our case it is the one which covers S the most with the minimum number of treelets, as we said earlier.

Target coverage Once a corrected source coverage is computed, we apply another algorithm to select among all the EPs that are associated with the TLs selected, the ones that maximally cover the target sentence T, once again, with the minimum number of phrases.

The candidate EPs are those associated with the TLs obtained from the source coverage computation. These candidate EPs must also match the target sentence. The criteria used to find the score of a coverage hypothesis are, in order of importance, the target coverage (maximisation) and the number of covering EPs (minimisation). No target token is allowed to be covered by more than one EP (no overlapping EPs). However, we did allow EPs to cover the uncovered target tokens contained in the "gaps" left by another EP.

One additional constraint that this algorithm enforces is that no two EPs in the corrected target coverage can share the same source treelet in the set of treelets matching the source sentence.

Coverage results The optimal coverage figures are presented in Table 5.

Once again, the coverage statistics obtained are quite fair, and are comparable with those yielded by LUCENE using word- and chunk-based coverages. The coverages are almost identical to those presented in Table 4: a source and target coverage of roughly 60%. It is interesting to note that the contribution of discontiguous treelets and elastic phrases is non-negligible: they account for roughly 15% of the source and target coverage, respectively. Although it is not clear at this stage if traditional contiguous chunk-like units would have been able to cover these words in the test corpora, these discontiguous TPs may add some power and flexibility to a translation memory, when it is queried with substrings not found in a traditional database.

Metric	Source	Target
Optimal coverage	62.7%	56.4%
Cov. by contiguous TPs	46.0%	38.6%

Table 5: Coverage statistics of the test corpora with Tree-Phrases. The figures presentended are averages over all 1000 pairs of sentences.

If the reader is interested in more details concerning this approach, we have considered various source and target match policies as well as other types of coverages in (Langlais et al., 2005).

6 Conclusion

We proposed three distinct approaches to query a translation memory and tested each of them with the same training and test corpora, extracted from the Hansards debates. Each one seems to have its strengths and weaknesses and provides a different compromise between flooding the end user with unmanageable amounts of information to find the smallest of repetitions in the memory and being completely silent when queried with large (and therefore rare) sub-sentential units. In all cases, however, a relatively good coverage of source and target material indicates that we are on the right track: the memory "recognizes" most of the queries, and the associations it stores are significant.

Querying a database built with LUCENE using whole sentences yielded a poor coverage, while using random substrings of the initial sentence gave excellent coverage, but with a number of queries and hits that render the system almost useless without a good post-query filter, possibly

aware of more than just the source query.

Using the chunker provided by our commercial partner Lingua Technologies Inc. proved to be an improvement: even though the number of chunks available to cover a whole source sentence is rather limited, the source coverage can be quite good and LUCENE proves once again very useful in retrieving target candidates, using our word alignment system. More importantly, it has the significant advantage over random substring querying of severely limiting the number of queries made to the memory, and reducing the amount of proposed material. Some fine tuning is still necessary, but the results are promising for the 3GTM project.

The treelet-elastic phrase approach showed that non-contiguous units can be memorized and retrieved by a memory and be useful to the end user. Also, the coverages obtained, similar to those computed from the chunk-based approach, seem to validate the whole idea of TPs as a query unit in a translation database. Naturally, skipping some words always has the potential drawback of introducing some ambiguity, but we believe that the potential gains for the translator will outweigh this weakness.

Therefore, it seems that the chunk-based approach is viable in the framework of a 3GTM, and preferable to the random substring approach. Moreover, the TP approach can add some additional power by considering different units. How these approaches can be combined is still under investigation.

These coverage measures are approximations: no dynamic programming algorithm can accurately predict the use that the human translator will make of the target material that is presented to him or her by the system. Ultimately, the best experimental setup is an in situ test of the memory with actual translation professionals. This is why we are currently working at defining "user scenarios" (see Langlais and Simard (2003) for such scenarios), adapted to different kinds of unit retrievals (sequential units or not, variable-size units, etc.) from a translation memory.

Finally, it is worth noting that the kind of user interface provided to the end user is crucial for the performance of such systems. An ergonomic and user-friendly design, built to unobtrusively suggest target material, greatly affects productivity and is therefore an essential part of any translation aid software.

References

D. Bourigault and C. Fabre. 2000. Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de Grammaire*, (25):131–151. Toulouse le Mirail.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

C. Coulombe. 1991. Les qualités attendues d'un correcteur orthographique et syntaxique. In *Traitement automatique de la langue et industries de l'information*, Paris, France.

Y. Ding and M. Palmer. 2004. Automatic learning of parallel dependency treelet pairs. In *First International Joint Conference on Natural Language Processing*.

D. Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of ACL*, pages 80–87, Sapporo, Japan.

P. Langlais and M. Simard. 2003. De la traduction probabiliste aux mémoires de traduction (ou l'inverse). In *Proceedings of the 10e TALN*, pages 195–204, Batz-sur-Mer, France, June 11-14.

P. Langlais, F. Gotti, D. Bourigault, and C.Coulombe. 2005. EBMT by Treephrasing: a Pilot Study. to appear in Proceedings of the 2nd Worskshop on EBMT, Phuket, Thailand, September.

F.J. Och and H. Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440–447, Hongkong, China.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd ACL*, pages 271–279, Ann Arbor, Michigan, June.

M. Simard and P. Langlais. 2003. Statistical translation alignment with compositionality constraints. In *HLT-NAACL workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 19–22, Edmonton, Canada, May.

M. Simard, N. Cancedda, B. Cavestro, M. Dymetmann, É. Gaussier, C. Goutte, P. Langlais, A. Mauser, and K. Yamada. 2005. Une approche à la traduction automatique statistique par segments discontinus. In *Proceedings of the 12th TALN*, Dourdan, France, June 6-10.