Using multilingual content on the web to build fast finite-state direct translation systems

Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03071 Alacant, Spain.
mlf@dlsi.ua.es

Abstract

In this paper I try to identify and describe in certain detail a possible avenue of research in machine translation: the use of existing multilingual content on the web and finite-state technology to automatically build and maintain fast web-based direct machine translation systems, especially for language pairs lacking machine translation resources. The term *direct* is used to refer to systems performing no linguistic analysis, working similarly to pretranslators based on translation memories. Considering the current state of the art of (a) web mining for bitexts, (b) bitext alignment techniques, and (c) finite-state theory and implementation, I discuss their integration toward the stated goal and sketch some of the remaining challenges. The objective on the horizon is a web-based translation service exploiting the multilingual content already present on the web.

1 Introduction

This paper tries to identify and sketch the details of a possible avenue of research in machine translation: the use of existing multilingual content on the web and finite-state technology to automatically build and maintain fast web-based direct machine translation systems. The word *direct* is used here to refer to systems performing no linguistic analysis beyond text segmentation, working much like the pretranslators found in translation-memory applications. These resources are especially important for language pairs lacking machine translation or other linguistic resources such as parsers but having considerable presence on the web, and may be automatically discovered (section 2) using state-of-the-art techniques. Current bitext alignment techniques may yield subsentential correspondences or translation units; section 3 discusses the use of subsentential translation units, not only to translate the source text but also to actually segment it before translation. Section 4 briefly describes finite-state techniques and how they may be used both as a very efficient and easily-maintained storage for subsentential translation units and to perform translation-unit-driven segmentation and translation of the source text. Section 5 sketches possible strategies for updating the kind of subsentential finite-state translation memories proposed with information from new bitexts. In section 6, I give some features of the kind of web-based machine translation application aimed at. Finally, closing comments may be found in section 7.

2 The web as a source of translation units

The efficient use of publicly-available multilingual content in the world wide web for linguistic purposes has been the subject of a number of studies. In particular, Resnik (1999) describes a system which automatically discovers *bitexts* (bilingual texts) on the web, that is, pairs of texts that may be considered to be translations of each other; Chen & Nie (2000) describe in detail a complete architecture, partially built upon Resnik's proposal, to perform this task using statistical alignment techniques. As a side effect of bitext validation, each text is (or may easily be) divided into segments which are found to correspond to segments in the counterpart text (these bitext segments are sometimes called *translation units* or TUs for short). But there are a large number of bitext-alignment methods which might be used to validate or refine these alignments (Gale & Church 1991; Brown et al. 1991; Melamed 1996). An obvious application of these bilingual segments could be the translation of new documents in either direction — this is indeed the basis of a well-known technology, *translation memory* (TM). In view of the number of bitexts available on the web, the coverage would be very large for some language pairs, subjects, and registers. In particular, it would be specially interesting to explore those language pairs for which no translation software or other costly linguistic resources are available and for which there are, however, a large number of bitexts (for example Spanish–Basque): resources automatically generated from these bitexts would recycle the translation knowledge and effort invested in their preparation.

3 Using subsentential translation units

3.1 Smaller segments for coverage

Most commercial TMs[1] rely on non-linguistic or weakly-linguistic strategies (such as sentence-boundary determination, format analysis, or proper name identification) when it comes to determine the alignment between two texts. As a result, the segments are usually whole sentences (in regular running text) or list items (in tables or menus). Substantial manual validation of segments is sometimes required; most TM packages offer interactive alignment tools to perform this task. It is the case that sentence-long segments are usually too large to give good coverage for new texts, but may be decomposed in smaller segments which are still acceptable translation units (some commercial programs indeed try to find smaller segments, partial matches, etc.). The lack of coverage may be partly alleviated by using more advanced, sub-sentence or even subword alignment strategies (Simard & Langlais 2001), or statistical machine translation techniques. For instance, Marcu (2001) uses statistically-obtained source–target word alignments to obtain TU candidates: pairs of segments such that all words included in the source segment are aligned only with words included in the target segment and vice-versa ("contiguous alignments"). Subsentential strategies are expected to work very well for related languages (Tomás & Casacuberta 2001; Ribeiro et al. 2001). It is true that, even if subsentential TUs will clearly improve translation coverage, they are more likely to be ambiguous than complete-sentence TUs. Therefore, any application designed to use subsentential TUs has to be based on

[1] See `http://lisa.org/tmx/` for a comprehensive list of major translation memory vendors, namely, those adhering to the *translation memory exchange* (TMX) standard.

a robust and reliable subsentential alignment strategy and has to perform careful TU selection and validation in order to strike an adequate balance between coverage and precision. Annotating TUs with register (academic, newspaper, informal, etc.), language variety (i.e., American vs. Iberic Portuguese) and subject (politics, mechanics, banking, etc.) is crucial for getting a better balance.

3.2 Segment-driven segmentation

When segmenting a new text to be translated, TMs use the same non-linguistic or weakly-linguistic strategies as when aligning bitexts, but do not use the TM content, that is, existing TUs, to perform the segmentation. This is adequate for sentence-long TUs obtained using the same content-independent strategies, but it is important to note that preexisting sub-sentential translation units (TUs) may be used both to segment source sentences when generating a new translation and to help align new bitexts when enlarging the TU database. For example, if the TM contains the subsentential TU ("web bitexts", "los bitextos en la web") and a Spanish-English bitext contains "Web bitexts are a valuable resource." aligned with "Los bitextos en la web son un recurso muy valioso" one can use the TU to align the first part of the sentence and obtain the candidate TU ("are a valuable resource", "son un recurso valioso"). Some TM packages are advertised as being capable to produce new translations by *assembling* parts of the stored sentence-long TUs, perhaps using a related technique. However, I am not aware of any commercial TM that uses subsentential TUs to guide bitext alignment. To perform subsentential alignnment tasks efficiently, a simple protocol has to be designed, to be used both when feeding the TM and when using it to translate new text. Both tasks have to be efficiently implemented to power web-based publicly-available translation engines (such as the ones described by Macklovitch et al. 2000) and robots crawling the web for new TUs (such as the ones described by Chen & Nie 2000).

4 Finite-state storage of TU databases

4.1 Finite-state technology

If existing TUs are going to be used to segment new texts in addition to translating them, TU lookup is very intensive and has to be very efficient. Finite-state technology (Roche & Schabes 1997) may be part of the solution, as will be discussed below; since TUs are input–output pairs, they may be encoded as paths in a finite-state transducer (FST). Substrings inside these pairs may be shared with other pairs, using specialized versions (Mohri 1997) of customary finite-state minimization algorithms (Hopcroft & Ullman 1979:68) to obtain substantial space savings. Moreover, resources (FSTs) may be distributed across the web itself, and redundancy may be very helpful to obtain error-tolerant results, even if resulting translations for the same documents are found to be slightly different for different web traffic or server availability conditions.

A common way of using finite-state technologies relates to a very simple and efficient strategy for segmenting the input: the *left-to-right, longest-match* (LRLM) strategy, a greedy algorithmic scheme used both in lexical scanners for programming languages such as the ones generated by `lex` (Lesk 1975) and in the simplified fixed-length-phrase parsers of web-based machine translation (MT) systems such as Reverso (`http:`

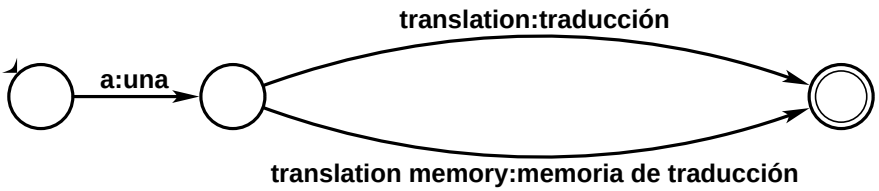

Figure 1: A finite-state transducer for the TUs ("a translation","una traducción") and ("a translation memory","una memoria de traducción") Double circles denote acceptance states.

//www.reverso.net) or Transcend RT (http://www.freetranslation.com). Finite state transducers (Oncina et al. 1993; Mohri 1997; Roche & Schabes 1997), when being used to segment the input in a LRLM way (by selecting the longest text token taking the FST to an acceptance state), may produce in each state transition partial outputs whose concatenation produces the target-language segment. LRLM segmentation may be seen as implementing a greedy (therefore suboptimal) cover of the source texts which roughly takes linear time with respect to text length. When translating the sentence "A translation memory is an archive..." (In Spanish, "Una memoria de traducción es un archivo..."), LRLM segmentation of the sentence would start by choosing the TU ("a translation memory","una memoria de traducción") rather than ("a translation","Una traducción") (a FST containing just these two TUs is shown in figure 1); after producing the corresponding output, the FST is restarted at the beginning of the remaining text ("is an archive...."). If no TU matches the beginning of the remaining text, a word is consumed from the input and copied to the output (flagged, perhaps, as "unknown"), and the FST is restarted.

4.2 Letter transducers for easy maintenance

One of the simplest models of FSTs, *letter transducers* (LTs, Roche & Schabes 1997), are basically isomorphic to finite-state automata (Hopcroft & Ullman 1979); therefore, they may be manipulated with well-established algorithms such as automaton minimization, and may be very efficiently implemented by generalizing existing techniques succesfully used to beat the customary techniques (e.g., hashing) when querying and updating string tables (Bentley & Sedgewick 1997). LTs have state-to-state transitions which read either one or zero characters and write either one or zero characters, and therefore may be seen as implementing the three basic edit operations at the character level: insertion, substitution or deletion[2]. Algorithms exist (Wagner & Fischer 1974, described also in Aho et al. 1986, p. 156) which minimize the number of operations (and therefore the number of state transitions) to obtain the target segment from the source segment, producing an optimal (minimum-length) character-level alignment. Such an alignment is likely to be shared by similar TUs, with additional spatial savings when using these alignments in the letter transducer. For example, the TUs ("two cases","dos casos") and ("ten cases","diez casos") may be compactly stored as shown in figure 2. Even when words are not cognates, similar TUs

[2]State transitions reading nothing and writing nothing are usually avoided.

containing them may share character-level alignments: when aligning the three TUs (“forget”,“olvidar”), (“forgotten”,“olvidado”) and (“forgetting”,“olvidando”) with the algorithm mentioned (Wagner & Fischer 1974), one finds that they share the prefix alignment (“␣forg”,“olvid”).

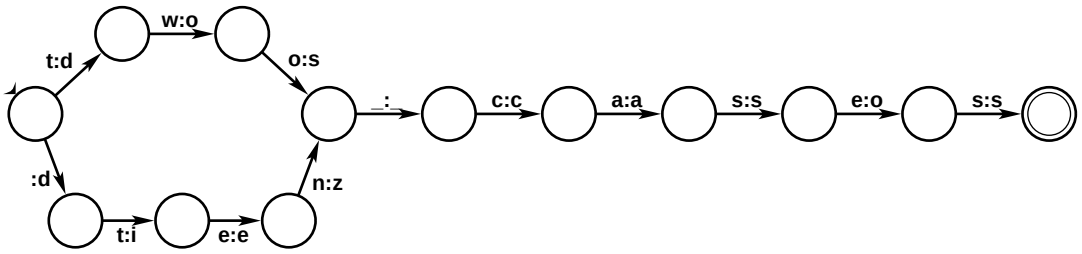

Figure 2: A finite-state letter transducer storing the TUs (“two cases”,“dos casos”) and (“ten cases”,“diez casos”). An underscore stands for any amount of valid inter-word whitespace.

As with most string search algorithms, the speed of finite-state machines degrades only very gracefully (logaritmically) with the number of entries. Also recently, a number of researchers (Daciuk et al. 2000; Carrasco & Forcada 2002; Garrido-Alenda et al. 2002) have found ways to incrementally maintain finite-state automata and LTs so that entries may be added to them or deleted from them while keeping them minimal (i.e, very efficient spatially), allowing not only for the easy addition of new TUs, but also for the deletion of TUs which are found to be in error.

5 Updating the finite-state translation memory

5.1 Updating during bitext alignment

Intuitively, a consistent way to process a new bitext to obtain new translation units to be used in an FST-based machine translation memory would be to perform this processing using also a LRLM scheme, to build a LRLM-compliant, partial cover of the current bitext. For that purpose one would

- start the FST storing the current TM at the beginning of the source text,
- look for the longest prefix of the input matching the source segment of an existing TU,
- check the target segment of that TU against the output,
- skip source words and target words when the target segment does not match the output before trying a new match,

and use heuristics to extend the matched TUs with skipped words or to combine matched TUs and skipped words into larger TUs (the examples in the next section will illustrate possible strategies). The word-skipping heuristic has to be carefully determined so that it is computationally feasible and does not lead to alignments which

are not reasonable in the LRLM scheme (i.e., skipping 50 source words while skipping 0 target words). Finite-state transducers are indeed a very convenient implementation of a simultaneous input-output matching procedure: one can run both the source text and the target text through the transducer, and a match is found when an input-output path leads the transducer to an acceptance state.

It is assumed that initially, the FSTs would contain a default set of common word-for-word or multiword TUs obtained from the (e.g. statistical) alignment of a set of texts (for multiword TUs, one could use the contiguous alignments described by Marcu 2001). One could also extend the FST so that it contains an identity-function loop mapping all unknown source words into themselves, to align proper names or other entities not needing translation (i.e., numerical quantities).

5.2 Example strategies

The following examples are designed to illustrate possible strategies to process a bitext to obtain new TUs, rather than to give a complete description of an algorithm. A complete, robust procedure would have to be designed by carefully selecting these strategies so that the whole procedure is computationally feasible as well as linguistically sensible. Let S be the set of source segments (word strings) s in the TM and T the function that assigns the corresponding target segments to them; that is, $(s, T(s))$ is a TU. When processing a bitext using the LRLM scheme sketched above one could find that the input has been segmented as follows:

Source: $\#\ s_1\ s_2\ s_3\ \ldots$

Target: $\#\ T(s_1)\ T(s_2)\ T(s_3)\ \ldots,$

that is, a sequence of source longest-matches that perfectly explains the target and therefore does not provide any new TUs (note that it is convenient to have a TU translating the beginning of a sentence, text, etc. into itself: ("#","#"), or $T(\text{“}\#'') = \text{“}\#''$). But in general, mismatches will be found; I will not try to be exhaustive, but instead, I will show some situations and hypothesize about how they can be handled. One possible situation is this:

Source: $\ldots\ s_k\ s_{k+1}\ldots$

Target: $\ldots\ T(s_k)\ \ldots,$

where $T(s_{k+1})$ is *not* a prefix of the text following $T(s_k)$. This indicates that the current TM needs updating, because a LRLM cover of the bitext is not possible with the existing TUs. A possible strategy is the following: look for the longest source segment $\hat{s}_k$ that is a prefix of the text following s_k and that is translated into a prefix $T(\hat{s}_k)$ of the target text immediately following $T(s_k)$; if there is one such prefix, extend the translation unit for s_k so that $T(s_k\hat{s}_k) = T(s_k)T(\hat{s}_k)$ and go on aligning the remaining text as before (this forces a new LRLM segmentation of the source text which may be more adequate than the one that failed). At a certain point, no new longest match or adequate extension of the previous longest match will be found and it will be necessary to skip words until the following situation is found:

Source: $\ldots\ s_l\ s'_l\ s_{l+1}\ \ldots$

Target: $\ldots\ T(s_l)\ t'_l\ T(s_{l+1})\ \ldots$,

where s'_l and t'_l are (possibly empty) sequences of words skipped until the match $(s_{l+1}, T(s_{l+1}))$ was found. One could simply add a new translation unit $T(s'_l) = t'_l$. This would be a good choice in some cases (e.g., when a proper name has been found). But further examination of the text could be possible; linguistically-motivated (and computationally feasible) heuristics could be used to obtain better TUs. Imagine that t'_l is null (zero words):

Source: $\ldots\ s_l\ s'_l\ s_{l+1}\ \ldots$

Target: $\ldots\ T(s_l)\ T(s_{l+1})\ \ldots$.

here, a reasonable choice would perhaps be to add the TU $T(s_l s'_l s_{l+1}) = T(s_l)T(s_{l+1})$. The converse may also occur: s'_l is null and target text t'_l seems to appear in a way that cannot be explained by the source

Source: $\ldots\ s_l\ s_{l+1}\ \ldots$

Target: $\ldots\ T(s_l)\ t'_l\ T(s_{l+1})\ \ldots$,

perhaps with a similar solution, $T(s_l s_{l+1}) = T(s_l)t'_l T(s_{l+1})$. But one could examine further and pay attention to situations like

Source: $\ldots\ s_l\ s_{l+1}\ s_{l+2}\ s_{l+3}\ \ldots$

Target: $\ldots\ T(s_l)\ t'_l\ T(s_{l+1})\ T(s_{l+3})\ \ldots$,

which in some cases may be interpreted as being due to a reordering, and remedied by adding $T(s_{l+1}s_{l+2}) = t'_l T(s_{l+1})$ or even a longer TU (a reordering would indeed be confirmed if one found that t'_l matches $T(s_{l+2})$).

A computationally-feasible approach would take a set of linguistically-sensible configurations such as the ones just described into account and combine them efficiently. This opens a complete line of research.

5.3 Counting for robustness

Statistical relevance is also an important issue: one does not want to make extensive changes to a TM simply because it does not explain a single example which may simply be an error or a very unlikely translation. One possibility is to store lists of alternate TUs for a source segment and their counts, and to keep only the most frequent TU in the main transducer. If that TU is finally found to be less frequent than an alternate TU, it may be removed from the transducer before the new TU is added using the incremental algorithms mentioned in section 4; if one wants TUs to compete in this way, the heuristics described in section 5.2 must be modified accordingly.

6 A web-based application

The techniques mentioned above may be implemented to power a web server which could easily be accessed in various modes. The server, in many respects, would be similar to a search engine (for example, it would crawl the web and mine it for bitexts as Resnik (1999) and Chen & Nie (2000) describe). Translation speeds for FST-based TMs like the ones described in this paper may be expected to be of the order of 100,000 words per second in a dedicated server; speed is crucial for a server which could be intensively used. Here is a list of the services that could be offered by such a system:

A fast pre-translator: When translating new texts, the best-ranked pretranslation for the longest-matched fragments would be produced left to right, with clearly-flagged default translations filling in the gaps, and aligned with the source text, for easy postedition, as in regular commercial translation memories; accordingly, users could correct the target text and realign TUs in a client-side application, which would give them the option to submit the resulting candidate TUs for consideration by the server.

Translated browsing: The current version of the direct translation could be used to obtain instantaneous translations (gists) of webpages during browsing, with adequate link manipulation, much in the same way as some current search engines (`http://www.google.com`, `http://www.altavista.com`) do. Even with partial coverage, having a low-quality translation available may be very useful for users unable to read the document directly in the source language.

A TU lookup interface: Professional translators translating documents by hand could use a simple form interface to look up the TM for a particular fragment they need to translate.

Contributing bitexts: Users could contribute candidate bitexts (pairs of URLs); the server would then check them, and, if adequate, align them and use them to update the TM.

7 Concluding remarks

It is envisioned that a careful integration of (a) web mining for bitexts, (b) bitext alignment techniques, (c) finite-state technologies, and (d) left-to-right, longest-match, subsentential segmentation heuristics may be a promising way to devise strategies for the automatic construction of very fast web-based direct MT systems from existing multilingual content on the web, which would work very much like "translation unit search engines". The current state of the art of web mining for bitexts and bitext alignment on the one hand, and of finite-state technology on the other hand make it possible for complete systems like the one described here to be made available on the web in a few years time. This kind of linguistic application is especially desirable for language pairs having a strong presence on the web but no machine translation systems or other computational linguistic resources available for them.

Acknowledgements: Partial support from the Spanish Comisión Interministerial de Ciencia y Tecnologia through project TIC2000-1599-C02-02 is acknowledged. Thanks go to Juan Antonio Pérez-Ortiz for useful discussions.

References

Aho, Alfred V., Ravi Sethi & Jeffrey D. Ullman: 1986, *Compilers: Principles, Techniques, and Tools*, Addison Wesley, ISBN 0-201-10194-7.

Bentley, Jon L. & Robert Sedgewick: 1997, 'Fast algorithms for sorting and searching strings', in *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*.

Brown, P., J. Lai & R.L. Mercer: 1991, 'Aligning sentences in parallel corpora', in *Proc. of the Assoc. Comput. Ling.*, Berkeley, pp. 169–176.

Carrasco, Rafael C. & Mikel L. Forcada: 2002, 'Incremental construction and maintenance of minimal finite-state automata', *Computational Linguistics*, **28**(2), forthcoming.

Chen, Jiang & Jian-Yun Nie: 2000, 'Parallel web text mining for cross-language IR', in *Proceedings of RIAO-2000*, Paris.

Daciuk, Jan, Stoyan Mihov, Bruce W. Watson & Richard E. Watson: 2000, 'Incremental construction of minimal acyclic finite-state automata', *Computational Linguistics*, **26**(1): 3–16.

Gale, W. & K. Church: 1991, 'A program for aligning sentences in bilingual corpora', in *Proceedings of the Association for Computational Linguistics*, Berkeley.

Garrido-Alenda, Alicia, Mikel L. Forcada & Rafael C. Carrasco: 2002, 'Incremental construction and maintenance of morphological analysers based on augmented letter transducers', in *Proc. of TMI-2002*, Keihanna, Japan.

Hopcroft, J.E. & J.D. Ullman: 1979, *Introduction to automata theory, languages, and computation*, Reading, MA: Addison–Wesley.

Lesk, M.E.: 1975, 'Lex — a lexical analyzer generator', Tech. Rep. Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J.

Macklovitch, Elliott, Michel Simard & Philippe Langlais: 2000, 'TransSearch: A free translation memory on the world wide web', in *Proc. of the 2nd Intl. Conf. on Lang. Resources and Evaluation LREC 2000*, Athens, Greece, vol. 3.

Marcu, Daniel: 2001, 'Towards a unified approach to memory- and statistical-based machine translation', in *Proc. ACL 2001*, Toulouse, France.

Melamed, I. Dan: 1996, 'A geometric approach to mapping bitext correspondence', in *Proc. 1st Conf. Empirical Methods in Natural Language Processing*, Philadelphia, PA, U.S.A.

Mohri, Mehryar: 1997, 'Finite-state transducers in language and speech processing', *Computational Linguistics*, **23**(2): 269–311.

Oncina, Jose, Pedro García & Enrique Vidal: 1993, 'Learning subsequential transducers for pattern recognition interpretation tasks', *IEEE Trans. on Pattern Anal., Machine Intell.*, **15**: 448–458.

Resnik, Philip: 1999, 'Mining the web for bilingual text', in *Proc. 34th Annu. Meeting Assoc. Comput. Ling. (ACL'99)*, Maryland, U.S.A.

Ribeiro, António, Gaël Dias, Gabriel Lopes & João Mexia: 2001, 'Cognates alignment', in Bente Maegaard, ed., *Proc. Machine Translation Summit VIII*, Santiago de Compostela, Spain., pp. 287–292.

Roche, E. & Y. Schabes: 1997, 'Introduction', in E. Roche & Y. Schabes, eds., *Finite-State Language Processing*, Cambridge, Mass.: MIT Press, pp. 1–65.

Simard, Michel & Philippe Langlais: 2001, 'Sub-sentential exploitation of translation memories', in Bente Maegaard, ed., *Proc. Machine Translation Summit VIII*, Santiago de Compostela, Spain., pp. 335–339.

Tomás, Jesús & Francisco Casacuberta: 2001, 'Monotone statistical translation using word groups', in Bente Maegaard, ed., *Proc. Machine Translation Summit VIII*, Santiago de Compostela, Spain., pp. 357–361.

Wagner, Robert A. & Michael J. Fischer: 1974, 'The string-to-string correction problem', *J. ACM*, **21**(1): 168–173.