AUTOMATYKA • 2010 • Tom 14 • Zeszyt 3/1

Szymon Grabowski*, Cezary Draus**, Wojciech Bieniecki*

Recognizing Non-Translatable Symbols in a Multi-Lingual Computer-Assisted Translation System for DTP Documents

1. Introduction

Recent years have witnessed a significant progress in Computer-Assisted Translation (CAT) tools, facilitating the work of a human translator. Such applications, as opposed to automated translation tools (e.g. Yahoo! Babel Fish or Google Translate), do not replace a human being but rather help him work faster and more efficiently.

Such programs work on small text segments (snippets) like sentences or expressions. Translated segments are stored in a database (called Translation Memory, TM [1]), ready to be used again. If a segment to be translated is found in the existing database (exact matching), the corresponding translation is proposed to be reused. If more than one translation of a given snippet is found, then several suggestions are proposed and it is up to the translator to decide which one to apply.

Some CAT tools work with raw files, while others are designed to handle more sophisticated formats of documents like word-processing, desktop publishing (DTP) files, web pages or presentation documents. Among them, desktop publishing documents represent a great challenge to the CAT tool family, because of their graphical diversity.

A 2006 survey [2] among 874 translation professionals from 54 countries revealed that although over 80% of the polled individuals do work with a translation memory system, most of them do not use a TM tool for all content, and the main reported reasons were: hardcopy documents available only; lack of support for the desired file format; the TM tools are too complicated for short texts; the repetition rate is too low; the tool is not suitable for the user's text types; complex layout. Obviously, better and more user-friendly applications could increase the use of this kind of software.

* Computer Engineering Department, Technical University of Lodz, Poland, {sgrabow,wbieniec}@kis.p.lodz.pl
** Cezary.Draus@gmail.com

This paper is a follow-up to [3], where we presented our web application, its assumptions, functional requirements and used technologies, and also presented algorithmic solutions to two problems: translation suggestions for short words and recognition of non-translatable symbols. Here we improve the solution for the latter problem, for which we also consider a 'subproblem' (ignored in [3]): how to detect (registered or unregistered) trademarks, i.e., the boundaries of phrases ended with ™ or ® symbol. The trademarks are non-translatable phrases and as such should be automatically copied (preserving the order, if a given entry contains more than one trademark) into the target field.

2. Catalogue and advertising brochure translations

Translation of desktop publishing documents like catalogues and advertising brochures is characterized by their graphical layout, where text, usually in form of short sentences or expressions, is formatted with different fonts and often mixed with tables, graphics, drawings, or pictures. Text contains many technical data with many symbols, abbreviations and numbers, which should not be translated. Many of DTP documents are printed regularly with a large part of its content unchanged (for example, annually reviewed and updated versions of catalogues). In this case even 80–90% of expressions may already be in the database and the translation of them can be done automatically. In DTP documents, text together with graphics, tables and pictures are integral parts of the page. The meaning of the text depends strongly on the graphical context and translation cannot be done independently from it. That is why the translation process of such documents requires another approach than translation of documents with continuous text (like books or articles).

Most of the existing CAT programs allow to translate texts of desktop publishing documents separately from the graphical context of the text and are not adapted to the specificity of such documents (short text segments containing many numbers and symbols).

In this kind of documents (like corporate catalogues, brochures, leaflets) the style and the layout should be absolutely preserved in the target version. In different language versions of the same document the text changes, but the graphical layout must be conserved. In the real life many companies publishing DTP documents in multiple languages do not have procedures and tools to improve this process.

The MediaTrad project that we work for uses Adobe InDesign format (IDML) to design the brochures. It is an XML-based Document Object Model (DOM) representation of the whole brochure file which describes all properties and components of the document including page layout, raster and vector-based graphics and tables. Using this format it is possible to replace some text boxes with other preserving all formatting and layout. This gives a possibility of minimum-labor translation.

For the purpose of flexible text processing, the IDML files we deal with have their equivalents in MySQL database. The elements of the document, that are most important to the process of translation are: pages, text boxes and character groups having the same formatting (together with attributes determining their position and order within the page). These elements are put in two database tables: *file_pages* and *file_texts*. In *file_pages* there are data associating the page with the specific document, to determine the order, location and type of the page.

In *file_texts* each record contains a piece of text (from an individual text box) and its equivalent translation, together with its identification vector: order counter and location coordinates.

The text may contain some special characters, that cannot be edited during the translation process. Specifically, those include tabs (\t), line-breaks (\r), new paragraphs (\n) and some InDesign symbols.

For the purpose of our algorithm, text strings from *file_texts* are extracted and sorted with respect of the "order" field. The form in which the data is extracted from the database may cause some difficulties while processing (i.e., text box content may not be the whole sentence). This issue has been raised in the next section.

3. Recognizing non-translatable symbols

Some phrases in the source text should not be translated, as they represent numbers, units of measurement, particular model names etc. Recognizing those non-translatable symbols basically serves two purposes: (*i*) it identifies the fields which should be copied verbatim[1], and (*ii*) it helps detect gaps in a phrase which then can be hinted with a TM entry (if e.g. the source field is *Shunt trip 1* and the word *1* is recognized as a "symbol", then the TM is searched for *Shunt trip* or *Shunt trip [symbol]*, and if it contains e.g. *Shunt trip 2*, its translation will be prompted with the number 2 replaced with 1).

In [3] we assumed the symbols will be recognized at word level, where a "word" is a maximum contiguous sequence of non-whitespace characters. Roughly speaking, the rules for classifying a word as a symbol were based on prevalence of digits and / or uppercase letters and existence of atypical (at least for a given language) sequences of letters, like *VDC*. The algorithm presented there, and being also the basis of the currently presented one, made use of digram statistics gathered for each used language (taken from some plain-text book taken e.g. from Project Gutenberg). Details can be found in [3].

Currently we inspect the issue of words with several glued segments, like *16-way*, where analysis based on digram statistics and classes of characters only may be misleading. Instead, we should split the phrase using the character – (in this example) and analyze both resulting segments. Still, to label some short words as non-symbols requires knowledge

[1] This is not always correct. Sometimes e.g. same units of measurement have different abbreviations in different languages.

about a given language. Widely used examples could be: *wrt* (abbreviation of "with respect to"), *np.* (abbreviation of Polish "na przykład", which means "for example"), *i.e.* (abbreviation of Latin "id est" = "that is, in other words", frequently used in written English), *SVP* (abbreviation of French "S'il vous plaît" = "please; if it pleases you"), *w/o* (="without"). A vocabulary list of those commonly used abbreviations should be first checked to detect such frequently used abbreviations. Note that such a list has at most a few hundred items per language, hence can easily fit a CPU cache.

The second phase, applied only to those words which have not been found in the aforementioned list and labeled as non-symbols, is called here "stripping and splitting". If a given word starts and ends with a pair of [], (), {}, <>, " or "" characters, those are stripped off before further analysis. If then the word contains at least one of the characters from the class [–/:;], it is split on those characters and each resulting segment analyzed separately, using the routine from [3]. If all the segments are then labeled as "symbols", the whole input word is also labeled as a non-translatable symbol. Otherwise, the given word is labeled as non-symbol and it is up to the human to translate it. This solution prefers false negatives (labeling a genuine symbol as a non-symbol) than false positives, as it is better to leave the human translator a little more work but be more certain about correct output.

As an example, let us consider a hypothetical word *1073741824-way*. This will be split into *1073741824* and *way*, and since the digram analysis of *way* will lead to labeling it as a non-symbol, the whole word will be left for a human translator to handle. This might not be the case in "global" word analysis, due to prevalence of digits.

4. Recognizing trademarks

A trademark is a sign or indicator used by e.g. business organization to identify that the products or services labeled with it come from a unique source and should be distinguished from other entities. The owner of a trademark may start legal proceedings against those that use that trademark without a proper authorization.

The problem of recognizing boundaries (that is, the beginning) of a phrase ended with ® or a similar symbol must be attacked statistically, without much preconception about the content of the tentative trademark. Our algorithm has a few (usually filtering) phrases:

1. collect all entries from a given table (or more precisely, its specific column) containing ® or a similar symbol; make sure that words in each entry have a single space between them, with no leading or preceding spaces in the entries; the parsing into words is trivial and the resulting words may contain punctuation marks at their beginning (e.g., the opening parenthesis) or end (e.g., the comma or period mark);
2. reverse the word order for each entry, then sort the (modified) entries lexicographically;
3. traverse each entry word by word; if a given word is "the" (with any possible capitalization) or contains the comma symbol, then truncate the entry (without the current word); if a word contains the opening parenthesis the entry is truncated just after this word; finally all opening and closing parentheses are removed from the entries;

4. retain all one-word entries; truncate the longer entries according to the following rules: the maximum entry length is 3 words, if the first word starts with a capital letter or a digit then the following words also have to start with a capital or digit, otherwise the entry is truncated on the first non-compliant word;
5. sort the entries according to their length in words, with the one-word ones put at the beginning, take them one by one inserting into an initially empty set data structure if only none of their word-based proper suffixes is already in the set; reverse the word order of each item of the set structure and finally the set structure contains the desired trademarks.

The procedure requires some comments. Step 1 chooses the entries of interest and normalizes its whitespaces. Step 2 facilitates further processing. Step 3 is based on our assumption that trademarks should not contain the English word “the” either the comma mark (of course the list of banned symbols and words could be prolonged, also with regard to other languages). In step 4 some simple rule about prolonging a trademark phrase is applied, based on the classes of the starting symbols in words. The last step is based on the assumption that no trademark is a proper suffix of another.

Finally, we present brief test results in two tables. In Table 1 we can see the set of phrases containing trademarks (and truncated just after the trademark symbol if the phrase is longer). In Table 2 recognized isolated trademarks are shown. The trademarks’ suffixes are hidden but for better readability we have applied matching letter cases, e.g. for fictious trademarks Ping-Pong and HANDY KIT we would have Pxxx-Pxxx and HANXX KIX, respectively.

Table 1
Selected input phrases for the proposed algorithm

Powerxxx®
The Powerxxx®
material like the Powerxxx®
The design of the Powerxxx®
features on all Powerxxx®
with ExtxxCoxx®
ExtxxCoxx®
cooled unit by adding the COXX ARX®
assembly (+ COXX ARX®
V111-S Inverxxx®
V222-S Inverxxx®
The Inverxxx®
K12345-1 Inverxxx®

* The trademarks’ suffixes, in Table 1 and 2, were replaced with ‘x’ (lowercase letters) or ‘X’ (uppercase letters) characters, to use them freely (without the trademark indicator) without a permission.

Table 2
Recognized trademarks

Powerxxx®
ExtxxCoxx®
COXX ARX®
Inverxxx®

5. Our work in context

The problem of detecting *non-words*, i.e., sequence of characters non-existing in any reasonable dictionary of a given language, is widely recognized [4]. Standard solutions either are based on a plain lookup in a dictionary base, or make use of *n*-grams. Using a readily available dictionary could be a perfect solution as long as this structure comprises a comprehensive collection of words, which is hardly the case for many practical cases (informal texts, with abbreviations, jargon etc.; names and symbols in product catalogues and brochures). The use of *n*-grams is a generalization of our digram idea (see Sect. 3) and the choice of $n = 3$ was often advocated as faring reasonably well [5, 6]. In this approach a sequence is labeled as a non-word if at least one of its (overlapping) *n*-grams does not occur in the statistics table of *n*-grams gathered over large and high-quality corpus data (note that our solution from Sect. 3 is less strict as a small number of atypical *n*-grams in a long "word" can be accepted). There are two variants of this methods, non-positional and positional *n*-grams, where in the former the frequency of a given *n*-gram in the corpus is taken overall, no matter its position in corpus words, while in the latter the position does matter. Positional *n*-grams offer better accuracy [6, 7] but require significantly more space. Similar ideas in this vein work on syllables or iteratively selected frequent *n*-grams (of varying *n*) [6].

The problem we consider in this paper is however somewhat different to the one discussed above, and we were unable to find it in the literature. "Symbols" as we perceive them are something different than typos or other errors; they are often valid words with attached digits or other characters, in other cases those are groups of characters which could be easily described with a simple regular expression (e.g., a range of temperature), but due to abundance of different possible expressions it is practically impossible to follow this approach for all particular cases.

6. Conclusions

We presented preliminary solutions of two algorithmic problems that popped up during the development of a computer-assisted translation system. Both problems revolve around detecting non-translatable phrases: one is to detect "symbols", like units of measurements, numbers etc., the other is to detect beginning of a trademark (whose end boundary is known as marked with ® or a similar symbol). The first problem is solved by statistical analysis of the characters inside a phrase (word) to check, splitting it also, if needed, on tentative separator characters (like a hyphen) and here some (very compact) general knowledge of a given language is used. The other problem requires analysis of other entries in the database since the trademarks tend to occur several times.

A natural possibility to strengthen the algorithm for the first problem is to use a full (rather than limited to abbreviations, as suggested in Sect. 3) dictionary for each language in question, but this requires more resources (memory, possibly costlier search). On the other hand, our proposed solutions can be considered lightweight.

It is common to have similar expressions within a single document. This reminds of an old idea of Morris and Cherry [8] to use frequency statistics of trigrams gathered not from a general corpus of text, but the actual document being inspected. We could apply this idea with classes of characters (e.g., digits) to match specific phrases which can be used for semantic labeling of “words” and expressions (e.g., \dºC, where \d stands for any digit, could be recognized as a temperature). Looking for such partial expressions can be faster and simpler to implement than regular expression search. Semantic labels can be useful in document indexing and proofreading.

References

[1] Rico Perez C., de Santa Olalla A.M., *New Trends in Machine Translation*. Meta, XLII, 4, 1997.

[2] Lagoudaki E., *Translation Memories Survey 2006. Translation Memory systems: Enlightening users' perspective*. Imperial College London, 2006, 39, available at http://www3.imperial.ac.uk/portal/pls/portallive/docs/1/7307707.PDF.

[3] Nowak G., Grabowski Sz., Draus C., Zarębski D., Bieniecki W., *Designing a computer-assisted translation system for multi-lingual catalogue and advertising brochure translations*. Proc. of 6th Int. Conf. MEMSTECH 2010, Lviv–Polyana, Ukraine, 2010, 175–180.

[4] Kukich K., *Techniques for Automatically Correcting Words in Text*. ACM Comput. Surv., 24(4), 1992, 377–439.

[5] Zamora E.M., Pollock J.J., Zamora A., *The use of trigram analysis for spelling error detection*. Inf. Process. Manage. 17(6), 1981, 305–316.

[6] Bressan S., Irawan R., *Morphologic Non-Word Error Detection*. DEXA Workshop, Zaragoza, Spain, 2004, 31–35.

[7] Hull J.J., Srihari S.N., *Experiments in Text Recognition with Binary N-Gram and Viterbi Algorithms*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 4(5), 1982, 520–530.

[8] Morris R., Cherry L.L., *Computer Detection of Typographical Errors*. IEEE Trans. on Professional Communication, 18(1), 1975, 54–64.