Designing a computer-assisted translation system for multi-lingual catalogue and advertising brochure translations

Grzegorz Nowak, Szymon Grabowski, Cezary Draus, Damian Zarębski, Wojciech Bieniecki

Abstract – Computer-Assisted Translation (CAT) tools help a human translator in his work. Such programs maintain a database with short translated expressions, as a source of suggestions. A major problem is lack of mature applications for handling desktop publishing (DTP) files, which pose a difficulty because of their graphical diversity. In this paper we present functional and technical requirements of a system currently being developed, and present experiments with algorithmic solutions to short word suggestion lists and recognition of non-translatable symbols.

Keywords – Computer-assisted translation, suggestion lists, text pattern recognition.

I. COMPUTER-ASSISTED TRANSLATION TOOLS

In the time of global communication and rapid information technology development, new horizons have opened to the translation activity. An important annual growth of the translation market and recent technological innovations have contributed to the development of Machine Translation and Computer-Assisted Translation. Machine Translation tools [4,5], often available on-line (e.g. Alta-Vista Translator or Google Translator), consist in translating text from one natural language to another fully automatically. In opposite, Computer-Assisted Translation (CAT) tools are computer programs that don’t replace a human translator, but help him work faster and more efficiently.

Such programs work on small text segments like sentences or expressions. Translated segments are stored in a database (called Translation Memory, TM), 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 used. If more than one translation of a given segments is found, then several suggestions are proposed and it is up to the translator to decide which one to apply.

Fuzzy-matching algorithms can propose translations for expressions which are not in the database, but represent a certain similarity to the texts being translated.

CAT tools are designed to save time and effort of the translator, to collect translations into reusable Translation Memory [6]. They contribute to the consistence of translation terminology and provide the translator with many useful statistics (for ex. translated/reused word & character count).

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 [1] 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.

II. CATALOGUE & 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 ex. 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 most common problems to solve are:

- the correct exporting strategy of texts from the original document;
- presentation of the text for translation in the form close to the original with the possibility to enter translations (preserving styles, fonts, tables, and layout used in the original document);
- truthful import of translations (replacing the original texts, preserving the existing styles);
- collecting and sharing of translation terminology (Translation Memory) within the corporate domain;
- maintaining translation consistence in the translated document and in all corporate documents (within the whole company);
- maximal automation of the translation process.

Our system is conceived to deal with these challenges.

III. OUR SYSTEM, FUNCTIONAL REQUIREMENTS

The typical localization process is composed of the following steps: separation of text from the graphical presentation, text translation, replacing the original text by the translated text, layout modification (the text length varies in different languages), verification and correction. This is laborious and manual process. Our main purpose is to reduce the time of the entire procedure. That is why we propose a functional architecture, to satisfy requirements of all the steps of the localization process. The schema of this architecture is shown on Fig. 1.

First, the document is uploaded for localization. Then, the translation is done in the graphical context, with the possibility of using automatic pre-translations and suggestions by means of glossaries. Expressions and their translations are automatically extracted and stored in the form of glossaries (Translation Memory), ready for the reuse. Now, the translated document should be verified and corrected. The final version of the document can be downloaded.

The main units which take part in the presented functional architecture are the following:

- Web interface. The application should be available for clients (firms) everywhere in the world to create localized and customized versions of their documents. Therefore a convivial user interface provides the necessary functions (login, document upload for localization, translation interface, localized document generation, final document download).
- Translator unit. Interface for the translator with a possibility of using automatic translations and suggestions, with context preview. Every translated phrase can be saved locally (in the current context) or globally (for all occurrences of the same original phrase).
- Dedicated database. Because the majority of phrases and expressions are used and reused in different corporate documents, it is important to collect them with the corresponding translations. Such database (Translation Memory) can significantly accelerate the translation process (80–90% of expressions may be already in the database and the translation of them can be done automatically). The database can be also a source of suggestions for new phrases that are translated for the first time. Every new translated document can be added to the existing database as a new glossary and
can be re-used in future translations. After corrections and proof-reading, glossaries are adjusted by deleting every incorrect translations of the expression and replacing it with the final one. Every glossary can be saved as private (usage reserved for the owner), common (usually specialized by domain and shared with other users), or main (kind of global dictionary).

The database concept not only allows the traditional bilingual translation relation between terms, but also monolingual ones such as synonyms.

The use of glossaries reduces significantly the amount of work needed to translate texts. It can increase the speed and the quality of translations and ensure the correctness and consistency of the translator’s work.

- Search unit. This feature is very useful when an expression was not found in the database as a translation or suggestion. Finding already translated similar expressions or sentences where a particular word was used very often requires searching many old files or even printing materials. The search engine is based on string comparison and allows finding any character strings in the original or translation. Found (original, translation) pairs are usually displayed with their context highlighting the searched text.

- The unit for generation of translated document in its original graphical form (layout).


There are some important functionalities worth to be mentioned. The first one is recognition of words for translation. A lot of corporate documents such as catalogues contain many expressions, symbols, reference numbers etc. that should not be translated (cf. Section VI). Automatic recognition of them reduces the translation volume and time.

The next one is populating a database in an automated manner. Consistent use of terminology for single translators or translation teams is one of the challenging aspects of translation quality. Unfortunately, manual creation of appropriate standardized glossaries is a very tedious and time-consuming task. Our system analyzes documents and automatically extracts expressions and their translations. All numbers, symbols and non-relevant expressions are eliminated and important terms and expressions are put into the database. Created glossaries can be easily validated, edited and reused. It is possible to import glossaries and translation memories and to export databases in a form of plain text files (csv) or the industry-standard TMX memory exchange format.

Another proposed functionality concerns the quality and consistency assurance. It is important to provide users with quality check functions that verify the consistent use of terminology and translations and can tell whether specific expressions or have been translated in more than one way. Using this feature, the translator has much greater control over the quality of his translations, and is able to detect and correct any possible oversights.

All these functionalities are easy to use. Modern web interface is designed to be intuitive, ergonomic and user friendly. Users do not need any training or instructions to start working with the system.

IV. TECHNICAL REQUIREMENTS AND USED TECHNOLOGIES

TECHNOLOGIES
The system is based on two powerful technologies: WWW and DTP. WWW service is realized in popular and widely tested web technologies, which are Apache Web Services and Database Connectivity. PHP was used as the programming language of dynamic web pages. Scalability is simple and fast since the Zend Framework Architecture was used, which covers the Model-View-Controller (MVC) paradigm and a subversion system to support team work. PHP as a popular web language assures also a communication with the MySQL database which necessary to store the DTP data and additional system information such as Translation Memory, user accounts, etc. The biggest challenge for the used database is to store and quickly operate on dictionariedata, needed especially for fast searching of translations.

Finally, the PHP allows to communicate directly with the DTP server. In this way, DTP services are realized in background and are transparent for users. DTP server is Adobe InDesign CS4 Server controlled with a script language – Adobe Script, an extension of Java Script.

To assure autonomy and stability of WWW and DTP parts, the virtualization technology was used, separating both parts on two distinct virtual machines: Linux Fedora (WWW part with Apache, PHP, MySQL) and Windows Server 2008 (DTP with InDesign Server CS4). Services communicate using http protocol and Microsoft Windows file sharing.

TECHNICAL REQUIREMENTS
The system requirements (server side) are the following. The server DTP (InDesign Server CS4) needs to be deployed on Windows OS or MAC OS. The DTP server is responsible for all DTP works, in particular text extraction and generating previews. In the presented system, 64-bit Windows Server 2008 was used. The minimal requirements for InDesign Server CS4 on Windows are as follows:

- 2GHz or faster x86 or x64 processor,
- Microsoft® Windows Server® 2003 with Service Pack 2 (32-bit and 64-bit) or Windows Server 2008 (32-bit and 64-bit),
- .NET Framework 2.0 required for Windows Service,
- 2 GB of RAM plus 256 MB per additional instance,
- 1.8 GB of available hard-disk space; additional free space required during installation,
- The Windows virtual memory paging file size must be increased by 2 GB for each instance of InDesign Server,
- 1024 x 768 screen resolution.

To assure the reliability and flexibility for the research and development, the web server is deployed on 64-bit Fedora which allows for easy and free use of all necessary web technologies.
Technical requirements for the end user (client side) are rather modest:

- Internet connection speed 512 kb/s,
- web browser compatible with Internet Explorer 6+, Mozilla Firefox 2+,
- support for JavaScript.

V. TRANSLATION SUGGESTIONS FOR SHORT WORDS

The larger the database, the wider range of suggestions it can offer to a translator. One problem is however with short entries, where a reliable similarity measure is needed not to prompt too many irrelevant phrases. It can easily happen (particularly for short entries) when matching comprises cases in which the current entry is only part of some entry in the database.

Our algorithm first gets the Soundex string for the given entry via a readily available MySQL function. The classic Soundex algorithm, devised before 1920 for indexing names, returns the first letter of a string followed by three digits, based on the successive consonants in the string (if it is too long, its suffix is simply ignored). For example, the consonants 'j' and 'v' have the same digit code, as e.g. Smirnov and Smirnoff are closely related surnames. The goal to match homophones (words with different spelling but identical pronunciation) makes Soundex the historically first (and still widely used) of the “phonetic algorithms”.

Now, in the first filtering step we are interested only in those glossary entries \( E(s) \) that have the Soundex code identical to the Soundex code \( sdx(s) \) of the current entry \( s \). If \( s \) is among them, then obviously it is returned (together with its translation) as the first suggestion, and removed from \( E(s) \).

There is a limitation imposed on the maximum number of suggestions as a too long list would distract the translator; for the experiments we assumed a limit of 7 items. If in any step the limit is reached, the suggestion list is ready and the algorithm terminates.

For the following, another phonetic algorithm is used, Metaphone [3], and it is run on the Soundex string of \( E(s) \). If the Metaphone code \( mtph(s) \) has length 1 (which we denote as \( |mtph(s)| = 1 \)), only those entries \( e_i \) of \( E(s) \) are added as the following suggestions for which \( mtph(s) = mtph(e_i) \) and \( Lev(s, \ e_i) < |s| \), where \( Lev(\cdot) \) is the well-known Levenshtein distance. If however \( |mtph(s)| > 1 \), then we add as suggestions those \( e_i \) for which \( mtph(e_i) \subset mtph(s) \). The selected entries are removed from \( E(s) \) again. Finally, if more suggestions are yet allowed, the phrase in \( E(s) \) having a longest common affix (i.e. prefix or suffix, whichever longer) with \( s \) is found and if there are more such entries, they are all added to the suggestion list, provided the assumed limit is not exceeded.

Finally, a measure \( m \) similar to the length of the longest common affix is used. The measure \( m(s_1, s_2) \) can be defined as \( \max(\text{maxlen}(s_1, s_2), \text{maxlen}(s_1, s_2)) \), where \( \text{maxlen}(s_1, s_2) = |s_2| \) if \( s_2 \) is fully contained in \( s_1 \) and 0 otherwise, and \( \text{maxlen}(s_1, s_2) \) is the length of the longest common affix (i.e. prefix or suffix, whichever longer) of \( s_1 \) and \( s_2 \). We calculate \( m(s, e_i) \) for all entries \( e_i \) in \( E(s) \) and select the entry for which \( m(s, e_i) \) is maximized; if there are more such entries, they are all added to the suggestion list, provided the assumed limit is not exceeded. All string matching operations are performed in the case-insensitive manner.

The algorithm has been implemented in PHP 5.2.4 and it uses standard functions and structures for text processing. The first step of filtering is done mainly on database level (using MySQL’s soundex() to get list of words with the same code and using PHP to find matching words on this list). Then, using PHP’s metaphone(), we calculate Metaphone codes for every word. To calculate the last measure, we use our affixLen() function, which relies on PHP’s multibyte string functions, for a full Unicode compatibility. The last of standard functions, levenshtein(), is used in sorting suggestions, to remove words which have the same Metaphone code, but the Levenshtein distance between them and the source is critically high.


c| | ![Table](https://example.com/table.png)

The algorithm was run on an exemplary large table storing data about electric equipment. Table 1 presents some statistical data about the obtained suggestion lists for selected entries (of length up to 5 characters). The second column stores the numbers of items with identical Soundex code from the database; larger values mean that more sequences had to be filtered by the PHP script. The third column shows the first suggestion with the Soundex criterion only; this is just to
show how different from the input word the strings returned by Soundex can be. The last two columns show the length of the final suggestion list (at most 7, but often much less) and the first suggestion, respectively.

In Table 2, selected suggestion lists are fully shown. The order of suggestions on the lists is preserved. The source language was English and the target language was Polish. Each suggestion in its right column is accompanied with the (Polish) translation of the whole field from the database.

<table>
<thead>
<tr>
<th>Source</th>
<th>Suggestions (English and Polish transl.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80mm</td>
<td>80mm, 80mm mm, 35mm, 60mm, 60mm 110mm, 110mm 160mm, 160mm 210mm, 210mm</td>
</tr>
<tr>
<td>[cd]</td>
<td>Code, Nr kat.</td>
</tr>
<tr>
<td>kA</td>
<td>kA, kA : 100 kA Icu = Ics = Icw, : 100 kA Icu = Ics = Icw</td>
</tr>
<tr>
<td>n/a</td>
<td>N/A, N/A no, nie</td>
</tr>
<tr>
<td>0,6 s</td>
<td>0,6 s, 0,6s s, s 0,06 s, 0.06 s 0,065 s, 0.065s</td>
</tr>
<tr>
<td>6yr</td>
<td>6yr, 6 lat 3yr, 3 lata</td>
</tr>
<tr>
<td>Cap</td>
<td>--</td>
</tr>
<tr>
<td>Coil</td>
<td>COIL, CSWL Coil, Cewka</td>
</tr>
<tr>
<td>Red</td>
<td>Red, Czerwona Red, Czerwone red, czerwony RED, CZERWONY</td>
</tr>
<tr>
<td>SI</td>
<td>SI, SI: s, s</td>
</tr>
</tbody>
</table>

From the obtained results several conclusions can be drawn. If there exists an exact match, it is shown first. The list lengths vary depending on how many similar entries have been found. The letter case does not matter for suggestions, although if more case variations are matching exactly, the same case is preferred (Red suggested earlier than red when Red was the input). The current implementation does not remove duplicate suggestions, so very easily some of the lists may get shorter without any loss in quality. Still, some more ideas are needed to improve suggestions for some very short entries (cf. [cd] → Code in Table 2). We note however that perfect suggestion accuracy is impossible in the current solution that starts with Soundex filtering. As an example, cap triggered no suggestions (see Table 2), although there exists the word Caps in the glossary. The reason is simple: the Soundex codes for the two words are different (C100 vs. C120). Hence, as a future task a more sophisticated preliminary filtering measure should be implemented, run and stored on the database server. This step may be costly (and repeated periodically since the glossaries are dynamic) but should amortize over many expected searches.

VI. 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 verbatim1, 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 I and the word I 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).

We assume the symbols will be recognized at word level, where a “word” is a maximum contiguous sequence of non-whitespace characters. Commonsense rules for classifying a word as a symbol are:

- prevalence of digits and / or uppercase letters,
- possibly non-letters and non-digits inside (e.g., the ‘-’ or ‘#’ character),
- possibly mixed character case (i.e., at least two lowercase and two uppercase characters in a word),
- a non-letter at the beginning of the word,
- atypical (at least for a given language) sequences of letters, like VDC.

Our proposed algorithm works as follows.

- digram statistics are gathered for each used language (we use plain-text books for that purpose, e.g. from Project Gutenberg; the books are first converted to lower case);
- if at least $\text{thresh}_1 = 80\%$ of digrams in the considered word are non-existent or rare enough (less than $\text{thresh}_1 = 0.02\%$ relative frequency) in the gathered statistics for a given language, we mark this word as a symbol;
- otherwise, if the word consists only of digits (where the character '-' is also added to “digits”) and uppercase consonants (only standard Latin alphabet is currently used and the set of consonants excludes ‘A’, ‘E’, ‘I’, ‘O’, ‘U’, ‘Y’), and the number of both digits and capital letters (consonants) is greater than 0, then we mark this word as a symbol;
- otherwise, if the word has no other symbols than digits (including ‘-’), punctuation marks and uppercase letters (now, each of those classes may have zero characters in the word), it is also marked as a symbol;

1 This is not always correct. Sometimes e.g. same units of measurement have different abbreviations in different languages.
• in all other cases the word is not considered to be a symbol.

The algorithm has detected, among many others, the following words as symbols: MN31W06-A III MH42M10-A “B” F3ST60D 60068-2 WDFX41SNZ 240V 15° (/C) GPS-B: 0.5A (20-50KHz) (AC-3/AC-4) N** 250A (CDC) (56 DC 6. +40° 2m MP3 300W. /5 MH32M20-A (12V-5mA) (50kA) 12x12 -25 RCMK20002 0.25...35 1.5-12 ms 80mn WL-250 17/18 (and of course all integer values). However, there is still some problem with recognition of words from Latin-derived alphabets. Because of low occurrence of digrams containing diacritics, some words with diacritics may be interpreted as symbols and not translated. Possibly, it can be prevented, or at least mitigated, by selecting a diverse collection of books for gathering digram statistics.

VII. CONCLUSION

We presented general characteristics of a computer-assisted translation system, as well as preliminary solutions of two algorithmic problems that popped up during the early development of the application. A rather unique feature of the system is its orientation for desktop publishing (rather than pure text) documents, where preserving the exact layout of often complex mix of text and graphical elements is of utmost importance. Our proposed string similarity measures for the two problems discussed in Sections V and VI fare rather well, although future improvements seem only possible with more workload on the database server (this can be amortized over multiple searches). Exploring those possibilities, with efficiency analysis, will be subject of our future work.

ACKNOWLEDGMENT

This work was partially supported by Operational Programme – Innovative Economy: Priority I. “Research and Development (R&D) of modern technologies” (enhancing the significance of the education sector in the economy by means of realisation of R&D assignments), No: UDA-POIG.01.04.00-10-016/09.

REFERENCES