An Open-Source Shallow-Transfer Machine Translation Engine for the Romance Languages of Spain

Antonio M. Corbí-Bellot¹, Mikel L. Forcada¹, Sergio Ortiz-Rojas¹, Juan Antonio Pérez-Ortiz¹, Gema Ramírez-Sánchez¹, Felipe Sánchez-Martínez¹, Iñaki Alegría², Aingeru Mayor², Kepa Sarasola²

¹Transducens group, Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03071 Alacant

²IXA Taldea, Informatika Fakultatea,
Euskal Herrikoko Unibertsitatea, E-20071 Donostia

acorbi@dlsi.ua.es, mlf@ua.es, sortiz@dlsi.ua.es, japerez@dlsi.ua.es,
gema@internostrum.com, fsanchez@dlsi.ua.es, acpalloor@si.ehu.es, jibmamaa@si.ehu.es, ksarasasola@si.ehu.es

Abstract. We present the current status of development of an open-source shallow-transfer machine translation engine for the Romance languages of Spain (the main ones being Spanish, Catalan and Galician) as part of a larger government-funded project which includes non-Romance languages such as Basque and involving both universities and linguistic technology companies. The machine translation architecture uses finite-state transducers for lexical processing, hidden Markov models for part-of-speech tagging, and finite-state based chunking for structural transfer, and is largely based upon that of systems already developed by the Transducens group at the Universitat d’Alacant, such as interNOSTRUM (Spanish—Catalan) and Traductor Universia (Spanish—Portuguese). The possible scope of the project, however, is wider, since it will be possible to use the resulting machine translation system with new pairs of languages; to that end, the project also aims at proposing standard formats to encode the linguistic data needed. This paper briefly describes the machine translation engine, the formats it uses for linguistic data, and the compilers that convert these data into an efficient format used by the engine.

1. Introduction

This paper presents the current status of development and the main motivations of an open-source shallow-transfer machine translation (MT) engine for the Romance languages of Spain (the main ones being Spanish (es), Catalan (ca) and Galician (gl)) as part of a larger government-funded project which will also include MT engines for non-Romance languages such as Basque (eu) and involving four universities and three linguistic technology enterprises.¹ The shallow-transfer

¹ Most scholars consider Galician and Portuguese (pt) the same language; however, the official orthography of Galician is very different from the ones used for European and Brazilian Portuguese. Therefore, while grammatical resources will be rather reusable, lexical resources will not easily be.

² TALP (Universitat Politècnica de Catalunya), SLI (Universidade de Vigo), Transducens (Universitat d’Alacant), IXA (Euskar Herriko Unibertsitatea), imaxin|software (Santiago de Compostela), Elhuyar Fundazioa (Usurbil), and Eleka Ingeniaritzak Linguistikoa (Usurbil, coordinator).
The multilingual nature of Spain is recognized, to a varying extent, in laws and regulations corresponding to the various levels of government (the Constitution of Spain and the Statutes of Autonomy granted to Aragon, the Balearic Islands, Catalonia and Valencia (ca), Galicia (gl), and Navarre and the Basque Country (eu)). On the one hand, demand by many citizens in these territories make private companies increasingly interested in generating information (documentation for products and services, customer support, etc.) in languages different from Spanish. On the other hand, the various levels of government (national, autonomic, provincial, municipal) must respect, in the mentioned territories, the linguistic rights recognized to their citizens and promote the use of such languages. Machine translation is a key technology to meet these goals and demands.

Existing MT programs for the es—ca and the es—gl pairs (there are no programs for the es—eu pair) are mostly commercial or use proprietary technologies, which makes them very hard to adapt to new usages, and use different technologies across language pairs, which makes it very difficult to integrate them in a single multilingual content management system.

The MT architecture proposed here uses finite-state transducers for lexical processing, hidden Markov models for part-of-speech tagging, and finite-state based chunking for structural transfer, and is largely based upon that of systems already developed by the Transducers group such as interNOSTRUM\(^3\) (Spanish—Catalan, Canals-Marote et al. 2001) and Traductor Universia\(^4\) (Spanish—Portuguese, Garrido-Alenda et al. 2003); these systems are publicly accessible through the net and used on a daily basis by thousands of users.

One of the main novelties of this architecture is that it will be released under an open-source license\(^5\) (together with pilot linguistic data derived from other open-source projects such as Freeling (Carreras et al. 2004) or created specially for this purpose) and will be distributed free of charge. This means that anyone having the necessary computational and linguistic skills will be able to adapt or enhance it to produce a new MT system, even for other pairs of related languages. The whole system will be released at the beginning of 2006.\(^6\)

We expect that the introduction of a unified open-source MT architecture will ease some of the mentioned problems (having different technologies for different pairs, closed-source architectures being hard to adapt to new uses, etc.). It will also help shift the current business model from a licence-centred one to a services-centred one, and favour the interchange of existing linguistic data through the use of the XML-based formats defined in this project.

It has to be mentioned that this is the first time that the government of Spain funds a large project of this kind, although the adoption of open-source software by administrations in Spain is not new.\(^7\)

The following sections give an overview of the architecture (sec. 2), the formats defined for the encoding of linguistic data (sec. 3), and the compilers used to convert these data into an executable form (sec. 4); finally, we give some concluding remarks (sec. 5).

---

\(^{3}\) http://www.internoustrum.com/

\(^{4}\) http://traductor.universia.net/

\(^{5}\) The license has still to be determined. Most likely, there will be two different licenses: one for the machine translation engine and tools, and another one for the linguistic data.

\(^{6}\) Other attempts at open-source implementations of MT systems have been initiated, such as GPLTrans (http://www.translator.cx) and Traduki (http://traduki.sourceforge.net), but the level of activity in these projects is low and far from reaching the usability levels of the existing interNOSTRUM—Traductor Universia engine inspiring the one in this project.

\(^{7}\) The most remarkable case being the success of Linex (http://www.linex.org), the Linux distribution promoted by the autonomous government of Extremadura.
2. The MT architecture

The MT strategy used in the system has already been described in detail (Canals-Marote et al. 2001; Garrido-Alenda et al. 2003); a sketch will be given here. The engine is a classical shallow-transfer or transformer system consisting of an 8-module assembly line; we have found that this strategy is sufficient to achieve a reasonable translation quality between related languages such as es, ca or gl. While, for these languages, a rudimentary word-for-word MT model may give an adequate translation for 75% of the text, the addition of homograph disambiguation, management of contiguous multi-word units, and local reordering and agreement rules may raise the fraction of adequately translated text above 90%. This is the approach used in the engine presented here.

To ease diagnosis and independent testing, modules communicate between them using text streams (examples below give an idea of the communication format used). This allows for some of the modules to be used in isolation, independently from the rest of the MT system, for other natural-language processing tasks. As in interNOSTRUM or Traductor Universia, implementations of the systems for Linux and Windows architectures will be made available.

The modules are shown in figure 1.

Most of the modules are capable of processing tens of thousands of words per second on current desktop workstations; only the structural transfer module lags behind at several thousands of words per second.

As has been mentioned in the introduction, in addition to this shallow-transfer MT architecture, the project is also designing a deeper-transfer architecture for the es—eu pair (Díaz de Ilarraza et al. 2000). Even though the current prototype is being programmed in an object-oriented framework using code and data from Freeling (Carreras et al. 2004) for es and conventional syntactical and lexical generation for eu, its architecture could easily be rewritten into one which would share modules and format specifications with the shallow-transfer architecture described here (for instance, morphological analysis and generation, part-of-speech tagging, or lexical transfer).

The following sections describe each module of the shallow-transfer architecture in detail.

2.1. The de-formatter

The de-formatter separates the text to be translated from the format information (RTF, HTML, etc.). Format information is encapsulated so that the rest of the modules treat it as blanks between words. For example, the HTML text in Spanish:

\[ \text{vi} \text{ <em>una señal</em>} \]

(“I saw a signal”) would be processed by the de-formatter so that it would encapsulate the HTML tags between brackets and deliver

\[ \text{vi} \text{[ <em>]una señal[</em>]} \]

The character sequences in brackets are treated as simple blanks between words by the rest of the modules.

2.2. The morphological analyser

The morphological analyser tokenizes the text in surface forms (lexical units as they appear in texts) and delivers, for each surface form, one or more lexical forms consisting of lemma, lexical category and morphological inflection information. Tokenization is not straightforward due to the existence, on the one hand, of contractions, and, on the other hand, of multi-word lexical units. For contractions, the system reads in a single surface form and delivers the corresponding sequence of lexical forms (for instance, the es preposition—article contraction del would be analysed into two lexical forms, one for the preposition de and another one for the article el). Multi-word surface forms are analysed in a left-to-right, longest-match fashion; for instance, the
analysis for the \textit{es} preposition \textit{a} would not be delivered when the input text is \textit{a través de} ("through"), which is a multi-word preposition in \textit{es}. Multi-word surface forms may be invariable (such as a multi-word preposition or conjunction) or inflected (for example, in \textit{es}, \textit{echaban de menos}, “they missed”, is a form of the imperfect indicative tense of the verb \textit{echar de menos}, “to miss”). Limited support for some kinds of discontinuous multi-word units is also available. The module reads in a binary file compiled from a source-language morphological dictionary (see section 3.1).

Upon receiving the example text in the previous section, the morphological analyser would deliver

\begin{verbatim}
^vi/ver<vblex><ifi><1><sg>$
[ <em>]
^una/un<det><ind><f>/<sg>/unir<vblex><prs><1><sg>/unir<vblex><prs><3><sg>$
^señal/señal<n><f>/<sg>$[</em>]
\end{verbatim}

where each surface form is analysed into one or more lexical forms. For example, \textit{vi} is analysed into lemma \textit{ver}, lexical category lexical verb (\textit{vblex}), indefinite indicative (\textit{ifi}), 1\textsuperscript{st} person, singular, whereas \textit{una} (a homograph) receives three analyses: \textit{un}, determinant, indefinite, feminine singular, and two forms of the present subjunctive (\textit{prs}) of the verb \textit{unir} (to join). The characters “^” and “$” delimit the analyses for each surface form; lexical forms for each surface form are separated by “/”; angle brackets “<...>”are used to delimit grammatical symbols. The string after the “^” and before the first “/” is the surface form as it appears in the source input text.

### 2.3. The part-of-speech tagger

As has been shown in the previous example, some surface forms (about 30\% in Romance languages) are homographs, ambiguous forms for which the morphological analyser delivers more than one lexical form. The \textit{part-of-speech tagger} chooses one of them, according to the lexical forms of neighbouring words. When translating between related languages, ambiguous surface forms are one of the main sources of errors when incorrectly solved.

The part-of-speech tagger (an open-source program) reads in a file containing a hidden Markov model (HMM) which has been trained on representative source-language texts (using an open-source training program). Two training modes are possible: one can use either a larger...
amount (millions of words) of untagged text processed by the morphological analyser or a small amount of tagged text (tens of thousands of words) where a lexical form for each homograph has been manually selected. The second method usually leads to a slightly better performance (about 96% correct part-of-speech tags). We are currently building a collection of open corpora (both untagged and tagged) using texts published on the web under Creative Commons licenses. The behaviour of the part-of-speech tagger and the training program are both controlled by a tagger definition file (see section 3.2).

The result of processing the example text delivered by the morphological analyser with the part-of-speech tagger would be

`\^ver<vblex><ifi><1><sg>$
[ <em>`\^un<det><ind><f><sg>$
\^señal<n><f><sg>$[</em>]`

where the correct lexical form (determiner) has been selected for the word una.

2.4. The lexical transfer module

The lexical transfer module is called by the structural transfer module (see next section); it reads each source-language lexical form and delivers a corresponding target-language lexical form. The module reads in a binary file compiled from a bilingual dictionary (see section 3.1). The dictionary contains a single equivalent for each source-language entry; that is, no word-sense disambiguation is performed. For some words, multi-word entries are used to safely select the correct equivalent in frequently-occurring fixed contexts. This approach has been used with very good results in Traductor Universia and interNOSTRUM.

Each of the lexical forms in the running example would be translated into Catalan as follows:

- `ver<vblex> → veure<vblex>`
- `un<det> → un<det>`
- `señal<n><f> → senyal<n><m>`

2.5. The structural transfer module

The structural transfer module uses finite-state pattern matching to detect (in the usual left-to-right, longest-match way) fixed-length patterns of lexical forms (chunks or phrases) needing special processing due to grammatical divergences between the two languages (gender and number changes to ensure agreement in the target language, word reorderings, lexical changes such as changes in prepositions, etc.) and performs the corresponding transformations. This module is compiled from a transfer rule file (see section 3.3). In the running example, a determiner-noun rule is used to change the gender of the determiner so that it agrees with the noun; the result is

`\^veure<vblex><ifi><1><sg>$
[ <em>`\^un<det><ind><f><sg>$
\^senyal<n><m><sg>$[</em>]`

2.6. The morphological generator

The morphological generator delivers a target-language surface form for each target-language lexical form, by suitably inflecting it. The module reads in a binary file compiled from a target-language morphological dictionary (see section 3.1). The result for the running example would be

`vaig veure[ <em>]un senyal [</em>]`

2.7. The post generator

The post-generator performs orthographical operations such as contractions and apostrophations. The module reads in a binary file compiled from a rule file expressed as a dictionary (section 3.1). The post-generator is usually dormant (just copies the input to the output) until a special alarm symbol contained in some target-language surface forms `wakes` it
up to perform a particular string transformation if necessary; then it goes back to sleep.

For example, in Catalan, clitic pronouns in contact may change before a verb: *em* (“to me”) and *ho* (“it”) contract into *m’ho*, *em* and *els* (“them”) contract into *me’ls* and *em* and *la* (“her”) are written *me la*. To signal these change, linguists prepend an alarm to the target-language surface form *em* in target-language dictionaries and write post-generation rules to ensure the changes described.

2.8. The re-formatter

Finally, the re-formatter restores the format information encapsulated by the de-formatter into the translated text and removes the encapsulation sequences used to protect certain characters in the source text. The result for the running example would be the correct translation of the HTML text:

```html
vaig veure <em>un senyal</em>
```

3. Formats for linguistic data

An adequate documentation of the code and auxiliary files is crucial for the success of open-source software. In the case of a MT system, this implies carefully defining a systematic format for each source of linguistic data used by the system. The formats used by this architecture (which will not be described in detail for lack of space) are modified versions of the formats currently used by interNOSTRUM and Traductor Universia, which was defined in Garrido et al. 1999. The current DTD and examples of morphological and bilingual dictionaries may be found in [http://www.torsimany.ua.es/eamt2005/](http://www.torsimany.ua.es/eamt2005/).

Morphological dictionaries establish the correspondences between surface forms and lexical forms and contain (a) a definition of the alphabet (used by the tokenizer), (b) a section defining the grammatical symbols used in a particular application to specify lexical forms (symbols representing concepts such as noun, verb, plural, present, feminine, etc.), (c) a section defining paradigms (describing reusable groups of correspondences between parts of surface forms and parts of lexical forms), and (d) one or more labelled dictionary sections containing lists of surface form—lexical form correspondences for whole lexical units (including contiguous multi-word units). Paradigms may be used directly in the dictionary sections or to build larger paradigms (at the conceptual level, paradigms represent the regularities in the inflective system of the corresponding language). Bilingual dictionaries have a very similar structure and establish correspondences between source-language lexical forms and target-language lexical forms, but seldom use paradigms. Finally, post-generation dictionaries are used to establish

10 This is indeed the mechanism by which *de facto* standards appear.

11 Subject to minor modifications as the project progresses.
correspondences between input and output strings corresponding to the orthographical transformations to be performed by the post-generator on the target-language surface forms generated by the generator.

3.2. Tagger definition

Source-language lexical forms delivered by the morphological analyser are defined in terms of fine part-of-speech tags (for example, the word cantábamos [es] has lemma cantar, category verb, and the following inflection information: indicative, imperfect, 1st person, plural), which are necessary in some parts of the MT engine (structural transfer, morphological generation); however, for the purpose of efficient disambiguation, these fine part-of-speech tags may be grouped in coarser part-of-speech tags (such as verb in personal form).

The tagger definition file is also an XML file (the corresponding DTD may also be found in http://www.torsimany.ua.es/eamt2005/) where (a) coarser tags are defined in terms of fine tags, both for single-word and for multi-word units, (b) constraints may defined to forbid or enforce certain sequences of part-of-speech tags, and (c) priority lists are used to decide which fine part-of-speech tag to pass on to the structural transfer module when the coarse part-of-speech tag contains more than a fine tag. The tagger definition file is used to define the behaviour of the part-of-speech tagger both when it is being trained on a source-language corpus and when it is running as part of the MT system.

3.3. Structural transfer

An XML format for shallow structural transfer rules has been recently drafted; a commented DTD may be found in http://www.torsimany.ua.es/eamt2005/.

The rule files contain pattern—action rules describing what has to be done for each pattern (much like in languages such as perl or lex). Using a declarative notation such as XML is rather straightforward for the pattern part of rules but using it for the action (procedural) part means stretching it a bit; we have, however, found a reasonable way to translate the ad-hoc C-style action language used in the corresponding module of interNOSTRUM and Traductor Universia, which was defined in detail in Garrido-Alenda and Forcada (2001), into a simple XML notation having the same expressiveness. In this way, we follow as close as possible the declarative approach used in the XML files defining the linguistic data used for the tagger and for the lexical processing modules.

3.4. De-formatter and re-formatter

The current de-formatter and re-formatter used in Traductor Universia and interNOSTRUM are different for each of the three formats supported (plain ISO-8859-1 text, HTML and RTF). Their behaviour is specified following a pattern—action scheme, with patterns specified as regular expressions and actions written in C code, using lex to generate the executable code. At the time of writing these lines we are still studying whether a declarative XML-based definition of the behaviour of these and new format-processing modules is feasible; the main obstacle being nested dependencies in RTF-like formats, which make a lex-style finite-state reader hard to specify in a declarative way.

4. Compilers

Compilers to convert the linguistic data into the corresponding efficient form used by the modules of the engine are currently under development. Two compilers are used in this project: one for the four lexical processing modules of the system and another one for the structural transfer.

4.1. Lexical processing

The four lexical processing modules (morphological analyser, lexical transfer, morphological generator, post-generator) are currently being implemented as a single program12 which reads binary files containing a

12 The MT programs interNOSTRUM and Traductor Universia use four different compilers, one for each lexical processing task
compact and efficient representation of a class of finite-state transducers (letter transducers, Roche & Schabes 1997); in particular, augmented letter transducers (Garrido-Alenda et al. 2002). These binaries are an improved version of those used in interNOSTRUM and Traductor Universia and are generated from XML dictionaries (specified in section 3.1) using a new compiler, completely rewritten from scratch. The new compiler is much faster (taking seconds instead of minutes to compile the current dictionaries in interNOSTRUM and Traductor Universia) and uses much less memory, thanks to the use of new transducer building strategies and to the minimization of partial finite-state transducers during construction. This makes linguistic data development much easier, because the effect on the whole system of changing a rule or a lexical item may be tested almost immediately.

4.2. Structural transfer

As has been explained in section 3.3, the format of the structural transfer rules is still in a draft phase. When it is fixed, a compiler will be built (largely based on that of Garrido-Alenda and Forcada 2001) which will generate a module which will use finite-state technology to detect the patterns of source-language lexical forms needing processing and will contain fast code to generate the corresponding target-language lexical form patterns.13

5. Concluding remarks

This paper has shown the current state of development of an open-source shallow-transfer machine translation engine for the Romance languages of Spain (the main ones being Spanish, Catalan and Galician). This is one of the machine translation engines that will be developed in a large, government-funded open-source development project (the other one is a deeper-transfer engine for the Spanish—Basque pair, which will be described elsewhere). Furthermore, as a well-documented open-source engine, it could be adapted to translating between other Romance languages of Europe (French, Portuguese, Italian, Occitan, etc.) or even between related language pairs outside the Romance group (Swedish—Danish, Czech—Slovak, etc.).

Some of the components (modules, data formats and compilers) from this architecture will also be useful in the design of deeper-transfer architectures for more difficult language pairs; indeed, the project is also building a MT system for one such pair, Spanish—Basque, and some components of the architecture presented here will be tested on that language pair.

In particular, the shallow-transfer engine will not be designed from scratch but may rather be seen as a complete open-source rewriting of an existing closed-source engine (interNOSTRUM, Canals-Marote et al. 2001; Traductor Universia, Garrido-Alenda et al. 2003) which is currently used daily by thousands of people through the net, and the corresponding redesign of linguistic data formats and rewriting of compilers.

The code, together with pilot Spanish—Catalan and Spanish—Galician linguistic data to demonstrate it, will be released at the beginning of 2006 through the project web page (currently under construction).

Acknowledgements: Work funded by projects FIT-340101-2004-3 (Spanish Ministry of Industry, Commerce and Tourism) and TIC2003-08681-C02-01 (Spanish Ministry of Science and Technology). Felipe Sánchez-Martínez is supported by the Spanish Ministry of Science and Education and the European Social Fund through grant BES-2004-4711.

8
6. References


