Free/Open-Source Rule-Based Machine Translation

Cristina España-Bonet and Aarne Ranta (eds.)

Proceedings of a Workshop Held in Gothenburg 14-15 June, 2012

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg
Sweden

Technical report no 2013:03

ISSN 1652-926X

Contents

Introduction **v**

1 An IDE for the Grammatical Framework
John J. Camilleri
University of Gothenburg **1**

2 Evaluating North Sámi to Norwegian assimilation RBMT
Trond Trosterud and Kevin Brubeck Unhammer
University of Tromsø, Kaldera språkteknologi **13**

3 Choosing the correct paradigm for unknown words in rule-based machine translation systems
V. M. Sánchez-Cartagena, M. Esplà-Gomis, F. Sánchez-Martínez, J. A. Pérez-Ortiz
Universitat d'Alacant **27**

4 An Open-Source Toolkit for Integrating Shallow-Transfer Rules into Phrase-Based Statistical Machine Translation
V. M. Sánchez-Cartagena, F. Sánchez-Martínez, J. A. Pérez-Ortiz
Universitat d'Alacant **41**

5 A rule-based machine translation system from Serbo-Croatian to Macedonian
Hrvoje Peradin, Francis Tyers
University of Zagreb, Universitat d'Alacant **55**

6 Deep evaluation of hybrid architectures: Use of different metrics in MERT weight optimization
Cristina España-Bonet, Gorka Labaka, Arantza Díaz de Ilarraza, Lluís Màrquez, Kepa Sarasola
UPC Barcelona, University of the Basque Country **65**

Introduction

This volume is a collection of papers presented at FreeRBMT12, the Third International Workshop on Free/Open-source Rule-based Machine Translation, held in Gothenburg, Sweden, on 13-15 June 2012.

The FreeRBMT series of workshops aims to bring together the experience of researchers and developers in the field of rule-based machine translation who have decided to get on board the free/open-source train and are effectively contributing to creating a commons of explicit knowledge: machine translation rules and dictionaries, and machine translation systems whose behaviour is transparent and clearly traceable through their explicit logic. The workshops are also open for hybrid systems, which combine statistical and rule-based translation methods.

The six papers in this volume address general methods and tools (Papers 1, 3, 4), systems for particular languages (Papers 2, 5), and evaluation (Paper 6). In addition to the contributed papers, the workshop featured an invited talk, *The New Machine Translation—Getting blood from a stone* by Martin Kay, as well as tutorials, demos, and discussions.

Acknowledgments

All submissions were carefully reviewed by the programme committee:

Krasimir Angelov, Chalmers University of Technology
Emily M. Bender, University of Washington
Pierrette Bouillon, University of Geneva
Lauri Carlson, University of Helsinki
Marc Dymetman, XRCE Xerox
Cristina España-Bonet, UPC Barcelona
Mikel Forcada, Universitat d'Alacant
Michael Gasser, Indiana University
Philipp Koehn, University of Edinburgh
Hrafn Loftsson, Reykjavik University
Lluís Màrquez, UPC Barcelona
Lluís Padró, UPC Barcelona
Juan Antonio Pérez-Ortiz, Universitat d'Alacant
Aarne Ranta, University of Gothenburg (Programme Chair)
Manny Rayner, University of Geneva
Kepa Sarasola, University of the Basque Country

Kevin Scannell, Saint Louis University
Khalil Sima'an, University of Amsterdam
Felipe Sánchez-Martínez, Universitat d'Alacant
Anna Sågvall Hein, Uppsala University and Convertus AB
Antonio Toral, Dublin City University
Trond Trosterud, University of Tromsø
Francis Tyers, Universitat d'Alacant
Luis Villarejo, UOC Barcelona

The workshop was sponsored by the project MOLTO funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement FP7-ICT-247914.

Barcelona and Gothenburg, February 2013

Cristina España-Bonet *Aarne Ranta*

Chapter 1

An IDE for the Grammatical Framework

John J. Camilleri
University of Gothenburg

Abstract

The GF Eclipse Plugin provides an integrated development environment (IDE) for developing grammars in the Grammatical Framework (GF). Built on top of the Eclipse Platform, it aids grammar writing by providing instant syntax checking, semantic warnings and cross-reference resolution. Inline documentation and a library browser facilitate the use of existing resource libraries, and compilation and testing of grammars is greatly improved through single-click launch configurations and an in-built test case manager for running treebank regression tests. This IDE promotes grammar-based systems by making the tasks of writing grammars and using resource libraries more efficient, and provides powerful tools to reduce the barrier to entry to GF and encourage new users of the framework.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-ICT-247914.

1.1 Introduction

1.1.1 Grammatical Framework (GF)

GF is a special-purpose framework for writing multilingual grammars targeting multiple parallel languages simultaneously. It provides a functional programming language for declarative grammar writing, where each grammar is split between an abstract syntax common to all languages, and multiple language-dependent concrete syntaxes, which define how abstract syntax trees should be linearised into the target languages. From these grammar components, the GF compiler derives

both a parser and a lineariser for each concrete language, enabling bi-directional translation between all language pairs. (Ranta, 2011)

Apart from being a standalone logical and natural language framework, there also exists an open-source collection of GF resource grammars for a number of natural languages, collectively known as the *Resource Grammar Library* (RGL) (Ranta, 2009). Currently comprising 24 natural languages from around the world, the libraries cover low-level syntactic features like word order and agreement in each particular language. These details are abstracted away from the application grammar developer through the RGL's common language-independent API, making it possible to write multilingual grammar applications without necessarily having any extensive linguistic training.

1.1.2 GF grammar development

As a grammar formalism, GF facilitates the writing of grammars which can form the basis of various kinds of rule-based machine translation applications. While it is common to focus on the theoretical capabilities and characteristics of such formalisms, it is also relevant to assess what software engineering tools exist to aid the grammar writers themselves. The process of writing a GF grammar may be constrained by the framework's formal limits, but its effectiveness and endurance as a language for grammar development is equally determined by the real-world tools which exist to support it.

Whether out of developer choice or merely lack of anything better, GF grammar development typically takes place in traditional text editors, which have no special support for GF apart from a few syntax highlighting schemes made available for certain popular editors[1]. Looking up library functions, grammar compilation and running of regression tests must all take place in separate windows, where the developer frequently enters console commands for searching within source files, loading the GF interpreter, and running some test set against a compiled grammar. GF developers in fact often end up writing their own script files for performing such tasks as a batch. Any syntax errors or compiler warnings generated in the process must be manually interpreted.

While some developers may actively choose this low-level approach, the number of integrated development environments (IDEs) available today indicate that there is also a big demand for advanced development setups which provide combined tools for code validation, navigation, refactoring, test suite management and more. Major IDEs such as Eclipse, Microsoft Visual Studio and Xcode have become staples for many developers who want more integrated experiences than the traditional text editor and console combination.

1.1.3 Motivation

The goal of this work is to provide powerful development tools to the GF developer community, making more efficient the work of current grammar writers as well as promoting the Grammatical Framework itself and encouraging new developers to use the framework.

By building a GF development environment as a plugin to an existing IDE platform, we are able to obtain many useful code-editing features "for free". Thus rather than building generic development tools, we only need to focus on writing IDE customisations which are specific to GF, of course reducing the total effort required.

The rest of this paper is laid out as follows: section 2.2 describes the design choices which guided the plugin's development, section 1.3.1 then covers each of the major features provided by the plugin, and in section 3.8 we discuss our plans for evaluation along with some future directions for the work.

[1] See the GF Editor Modes page at `http://www.grammaticalframework.org/doc/gf-editor-modes.html`

1.2 Design choices

1.2.1 Eclipse

Eclipse[2] is a multi-language software development environment which consists of both a standalone IDE, as well as an underlying platform with an extensible plugin system. Eclipse can also be used for the development of self-contained general purpose applications via its Rich Client Platform (RCP). The Eclipse Platform was chosen as the basis for a GF IDE for various reasons:

1. It is written in Java, meaning that the same compiled byte code can run on any platform for which there is a compatible virtual machine. This allows for maximum platform support while avoiding the effort required to maintain multiple versions of the product.
2. The platform is fully open-source under the Eclipse Public License (EPL)[3], is designed to be extensible and is very well documented.
3. Eclipse is a widely popular IDE and is already well-known to a number of developers within the GF community.
4. It has excellent facilities for building language development tools via the Xtext Framework (see below).

1.2.2 Xtext

Xtext[4] is an Eclipse-based framework for development of programming languages and domain specific languages (DSLs). Given a language description in the form of an EBNF grammar, it can provide all aspects of a complete language infrastructure, including a parser, linker and compiler or interpreter. These tools are completely integrated within the Eclipse IDE yet allow full customisation according to the developer's needs. Xtext can be used both for creating new domain specific languages, as well as for creating a sophisticated Eclipse-based development environment.

By taking the grammar for the GF syntax as specified in Ranta (2011, appendix C.6.2), and converting it into a non-left recursive (LL(*)) equivalent, we used Xtext's ANTLR[5]-based code generator to obtain a basic infrastructure for the GF programming language, including a parser and serialiser. With this infrastructure as a starting point, a number of GF-specific customisations were written in order to provide support for linking across GF's module hierarchy system. Details of this implementation as well as other custom-built IDE features are described in section 1.3.1.

1.2.3 Design principles

Preserving existing projects

As users may wish to switch back and forth between a new IDE and their own traditional development setups, it was considered an important design principle to have the GF IDE *not* alter the developer's existing project structure. To this end, the GF Eclipse Plugin does not have any folder layout requirements, and never moves or alters a developer's files for its own purposes. For storing any IDE-specific preferences and intermediary files, meta-data directories are used which do not interfere with the original source files.

Preventing application tie-in in this way reduces the investment required for users who want to switch to using the new IDE, and ensures that developers retain full control over their GF projects.

[2] `http://www.eclipse.org/`
[3] `http://www.eclipse.org/legal/epl-v10.html`
[4] `http://www.eclipse.org/Xtext/`
[5] `http://www.antlr.org/`

This is especially important for developers using version control systems, who would want to use the plugin without risking any changes to their repository's directory tree.

Interaction with GF compiler

It is clear that an IDE which provides syntax checking and cross-reference resolution is in some sense replicating the parsing and linking features of that language's compiler. With this comes the decision of what should be re-implemented within the GF IDE itself, and what should be delegated to the existing GF compiler. In terms of minimising effort required, the obvious option would be to rely on the compiler as much as possible. This would conveniently mean that any future changes to the language, as implemented in updates to the compiler, would require no change to the IDE itself.

However, building an IDE which depends entirely on an external program to handle all parsing and linking jobs on-the-fly is not a practical solution. Thanks to Xtext Framework's parser generator as described above, keeping all syntax checking within the IDE platform becomes a feasible option, in terms of effort required versus performance benefit. When it comes to reference resolution and linking however, it was decided that the IDE should delegate these tasks to the GF compiler in a background process (see section 1.3.4). This avoids the work of having to re-implement GF's module hierarchy system within the IDE implementation. Communication of scope information from GF back to the IDE is facilitated through a new "tags" feature in the GF compiler, as described in section 1.3.3. This delegation occurs in a on-demand fashion, where the GF compiler is called asynchronously and as needed, when changes are made to a module's header.

1.3 The GF Eclipse Plugin (GFEP)

This section covers the major features provided by the plugin and their relevance to developers of GF grammars.

1.3.1 Code editing

Figure 1.1 shows a screenshot of the main IDE window. Note how multiple editor panes can be viewed simultaneously, by partitioning the workbench into arbitrary tabbed sections. Various source code-level features such as code folding, block-level indentation and commenting, and matching bracket highlighting are also provided. These basic code editing features, including the project navigation view in the top-left of the screen, are all provided directly by the Eclipse Platform.

Automatic formatting The built-in code formatter can be used to tidy one's code automatically, adhering it to the line break and indentation conventions as used in the GF book (Ranta, 2011). Figure 1.2 shows screenshots before and after invoking the code formatter.

Wizards The plugin also provides some *wizards* for guiding developers in quickly creating new resources in the project, such as creating a new GF module from scratch, or cloning an existing module in one language into a new one.

Syntax validation As the basic language infrastructure for the IDE was generated from a grammar of the GF syntax, the plugin provides fully customisable syntax highlighting as well as instant syntax validation and marking of lexical errors. A variety of semantic warnings may also be shown to the user, for example indicating that a linearisation rule has no corresponding abstract function, or that an implemented interface has not been fully instantiated. Note that these features

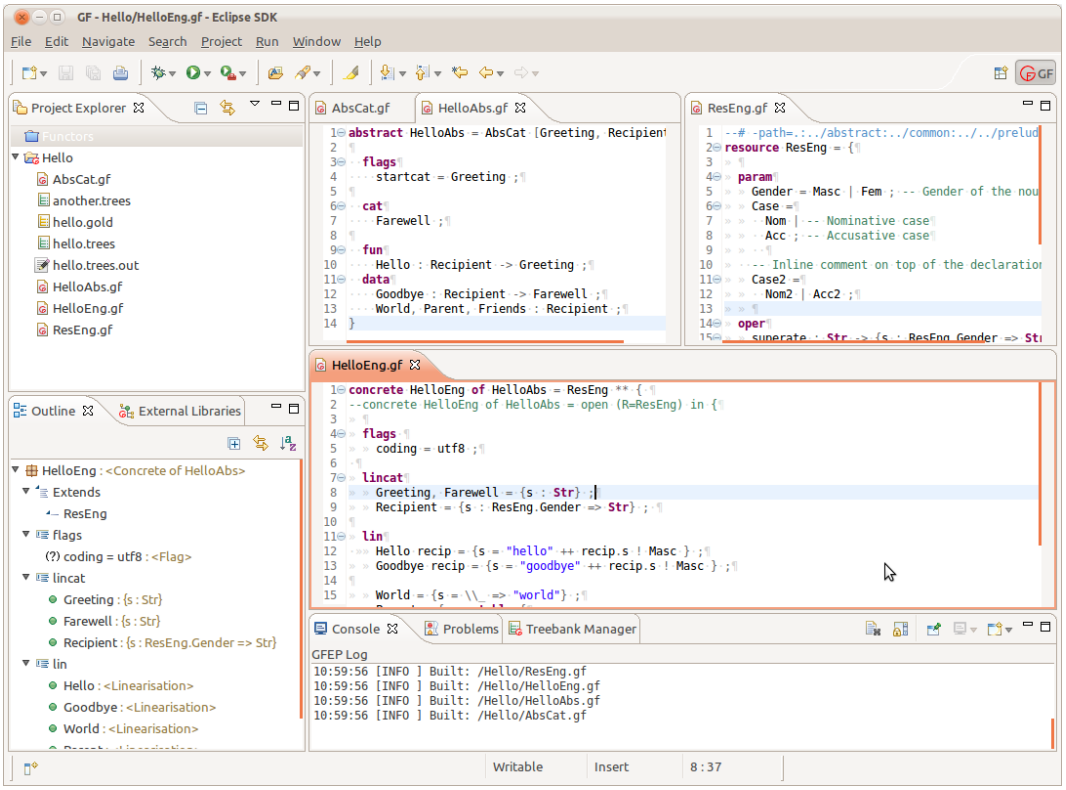


Figure 1.1: Screenshot of the GF Eclipse Plugin in use.

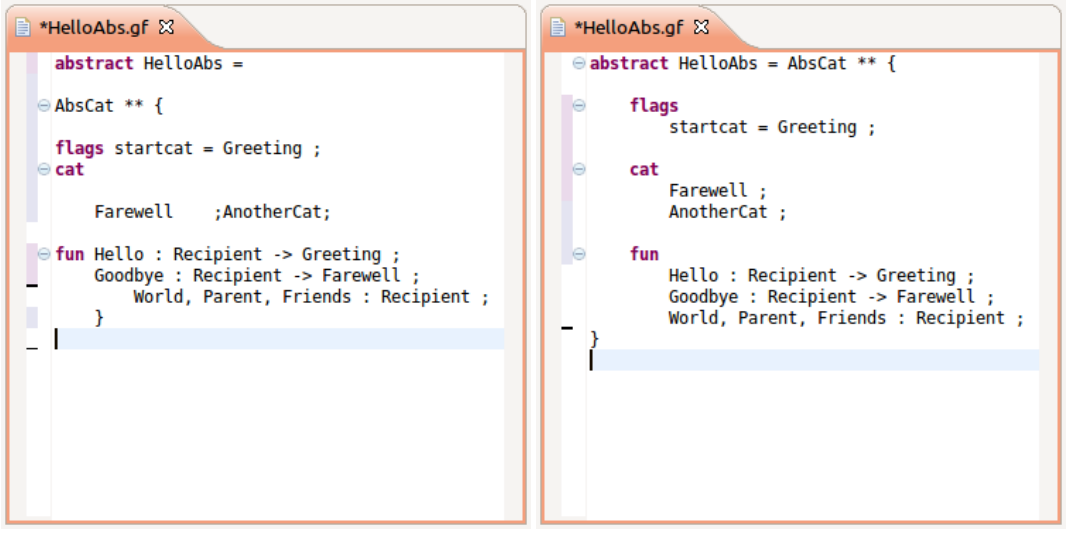


Figure 1.2: Before and after applying the automatic code formatting feature.

are all provided directly by the plugin implementation, without needing to call the standard GF compiler in the background.

Outline view The outline view in the bottom-left of figure 1.1 offers a complete overview of the current module structure. Every definition in the module is listed in a tree structure, along with its type information and helpful icons for quickly distinguishing the different judgement types. Clicking any of the terms will make make cursor jump to that point in the file, allowing for easy and quick navigation in large modules.

1.3.2 Launch configurations

Making use of Eclipse's launching framework, grammar writers have the ability to compile and run their modules with GF with a single click or button press. The plugin allows multiple *launch configurations* to be set up; each specifying the source modules to be compiled, any additional compiler flags, and any commands which should be passed to the GF shell for batch processing. Launch configurations can also be configured to automatically linearise treebank files for grammar regression testing (for more about this, refer to section 1.3.5). The GF compiler can also optionally be launched into interactive shell mode, such that the user can interact with the GF interpreter in the traditional way without leaving the development environment.

Once set up, any launch configuration can be run quickly from within the IDE, avoiding the need to type in long terminal commands or scroll through one's shell history each time. A screenshot of the options available in the launch configuration dialog window is shown in figure 1.3.

1.3.3 Cross-reference resolution and scoping

As in most other programming languages, GF comes with a hierarchical module system which allows grammars to be split between multiple source files (modules), and for these modules to import and extend each other in an inheritable way. An identifier in a module which points to a function or value defined in another module is known as a *cross-reference*. The GF IDE must thus link all such cross-references between modules, allowing the developer to "jump" to their original points of definition, and indicate when a referenced identifier cannot be resolved.

A byproduct of this is the ability to display a list of all functions available in the module hierarchy, which are visible from any given point in a grammar. This is provided as an auto-completion pop-up dialog, which filters the displayed list of available functions by the characters preceding the current cursor position.

All this is achieved through the scoping infrastructure of the GF Eclipse Plugin, which can quickly find all visible definitions (i.e. the scope) for any part of a grammar. As this scope calculation is highly specific to GF's module system and inheritance syntax, rather than attempting to re-implement this behaviour within the IDE, it was decided that this task should be handled by the standard GF compiler system. The delegation of this work from the IDE to the GF compiler is handled by a custom Eclipse builder (see section 1.3.4). In order to facilitate communication between the IDE and the standalone compiler, a new tags-generation feature was added to GF. This is described in the following section.

GF tags generation

Tags files are used as a means of providing module scope information to the IDE from the GF compiler, when the latter is invoked as a background process via the GFEP automatic builder as depicted in figure 1.4. The tags generation in GF is inspired by popular tools like Ctags and Etags[6].

[6] `http://ctags.sourceforge.net/ctags.html`

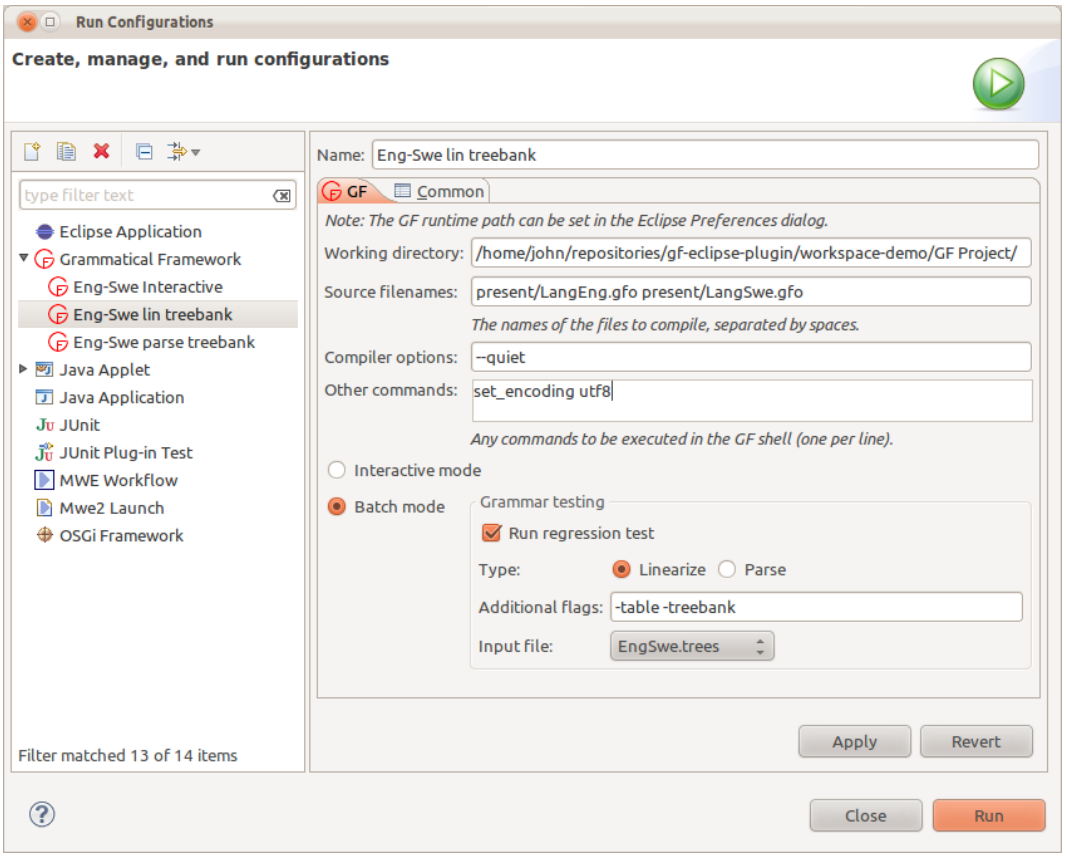


Figure 1.3: The launch configuration dialog, allowing developers to save their compilation flags and arguments for quick re-use.

As of GF version 3.3.3, running the compiler with the `-tags` flag will begin the regular compilation pipeline, starting with the usual phases for parsing and analysing of the grammar code but stopping before any actual code generation. Instead, the compiler will write a set of `.gf-tags` files (one for each `.gf` source module) containing lists of every identifier in the scope of the current module. These files are saved in a tab-delimited format with one identifier per line, as shown in figure 1.5.

The first two fields of each line indicate the identifier name and the *kind* of declaration; that is, the keyword that is used for introducing the identifier, i.e. `fun`, `cat`, `lin`, `lincat` or `oper`. If the identifier is defined in the current module, then the third field contains the path to the source file along with the line number(s) for the definition. When the type is either `fun`, `oper` or `overload` then the final field contains the type signature for the identifier.

In addition to the declaration kinds listed above, the kind could also be specified as `indir`, which indicates that the identifier is imported from some other module, and that its definition should be looked up there. In this case, the following fields on the same line respectively contain the module name and alias under which the identifier was imported (where applicable), and the path to the `.gf-tags` file which contains the actual definition of the identifier. This is exemplified in the final line of figure 1.5 (`mkNoun`).

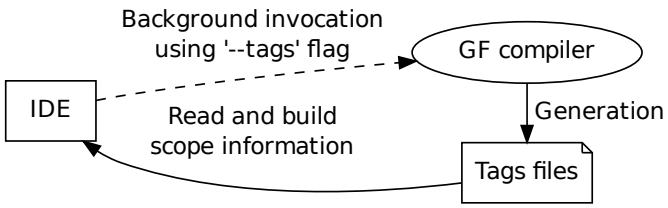


Figure 1.4: Tags files created by the GF compiler in a background process are used by the GF Eclipse Plugin for building scope information about the GF source files opened in the IDE.

```
mkN3  oper-def   .../ParadigmsEng.gf:406
mkN3  oper-type  .../ParadigmsEng.gf:118  {s : Number => Case => Str;...
mkPN  overload-def   .../ParadigmsEng.gf:390-393
mkPN  overload-type  .../ParadigmsEng.gf:390-393  Str -> {s : Case =>...
mkPN  overload-type  .../ParadigmsEng.gf:390-393  {s : Number => Case...
mkNoun  indir  ResEng  R  ResEng.gf-tags
```

Figure 1.5: Example of the `.gf-tags` file format, for the resource grammar library module `ParadigmsEng.gf` (some lines truncated for brevity).

1.3.4 Automatic builder

The reliance on the GF compiler for providing scoping information means that repeated calls to this external program must be made by the IDE. This is handled by a custom Eclipse *builder*, which listens for changes in the project workspace, analyses the resource deltas and calls the GF compiler to refresh the scoping information. This generally happens each time a file is saved, however the plugin also attempts to detect when changes to the current module may have effects on its dependents, in which case it will update the scoping information for these descendants also. To reduce the total number of calls to the builder, the scoping information is only refreshed when changes are made to the module's *header* information.

In addition to obtaining scoping information as described in section 1.3.3 above, calling the GF compiler as a background task also allows any type errors not caught by the IDE directly to still be relayed back to the user. Since all GF grammars written in the IDE will ultimately have to be compiled with GF, it is important that all errors are bubbled up to the developer as soon as possible so that they do not go undetected for long.

1.3.5 Test case manager

As described in Ranta (2011, section 10.5), the typically recommended development-test cycle for GF grammars is as follows:

1. Create a file `test.trees` which contains a list of abstract syntax trees (one per line) to be tested.

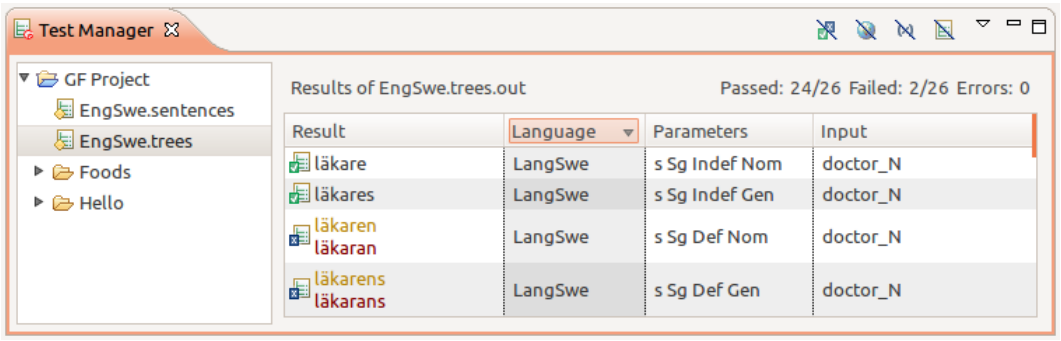


Figure 1.6: Viewing regression test output in the Test Manager view. In this example we see that the input tree `doctor_N` is incorrectly linearised as *läkaran* and *läkarans* for the singular definite cases in Swedish. The correct forms are *läkaren* and *läkarens*, respectively.

2. Compile the grammar and linearise each tree to all forms, using a command such as:

   ```
   rf -lines -tree -file=test.trees | l -table -treebank
   ```

   and capture the output in a file `test.trees.out`.
3. Manually correct the output in `test.trees.out` and save it as your gold standard file `test.trees.gold`.
4. Each time the grammar is updated, repeat step 2 and compare the new output against the gold standard using Unix `diff` or some other comparison tool.
5. Extend the tree set and gold standard file for every new implemented function.

The Tree Manager view in the GF Eclipse plugin provides a convenient graphical interface for managing this treebank testing process. This feature works together with the launch configurations to make the process of running grammar regression tests and gold standard comparisons quick and easy. As shown in the left-hand side of figure 1.6, all valid test input files in the project are shown together, and a simple double click on any will invoke the GF compiler, linearise the trees with the current version of the grammar and present comparisons against the corresponding gold standard in the right-hand panel. Various options are available for sorting and filtering the test results given, so that developers can quickly locate in which cases their grammar is failing.

Parsing While grammar testing is often focused on the linearisation of abstract syntax trees, the same procedure can be used equally as effectively for testing the *parsing* performance of the grammar under development. In this case, one would use `.sentences` instead of `.trees` files, containing plain-text sentences instead of abstract syntax trees, and the gold standard and output files would conversely contain the parse trees produced by the grammar.

1.4 Conclusions

Based on the Eclipse Platform an the Xtext framework, we have built a development environment for GF to replace the standard text editor and console window combination. While the GF Eclipse Plugin is not any more powerful in a computational sense, it does make available a number of development tools and user interfaces for speeding up the writing and testing of GF grammars, as well as the use of existing resource libraries in application grammars.

The GF Eclipse Plugin is a new tool for the GF community, and as such its popularity and ability to increase grammar writing productivity remain to be seen.

1.4.1 IDE use and evaluation

Inevitably, it will often be the case that seasoned GF developers are happy with their current development environments and would be unwilling to switch to a new IDE-based setup. As a result, such developers are not considered the primary target for users of the GF Eclipse Plugin. Rather, the expected target group would be those developers who are already familiar with Eclipse or at least some similar IDE platform, even if they are not necessarily experienced in GF.

For this reason, plans are underway for an objective evaluation of the plugin to be carried out by a private company who already work in Eclipse but are new to GF. The experience of these developers with the new IDE will provide valuable information about the effectiveness of the GF plugin, where the normal learning curve for Eclipse itself will not be an issue.

1.4.2 Future work

Apart from optimisations in performance and addressing the issues already identified with the plugin to date, the following two major directions for future work have been identified.

Refactoring tools A highly useful component of many IDEs—which is currently missing from the GF Eclipse Plugin—is the availability of source code refactoring tools. Such tools could include generic refactoring tasks such as renaming identifiers (both locally and across modules) and moving function definitions, to more GF-specific ones such as extracting functors from groups of concrete syntaxes. Such tools have the potential to minimise time spent on repetitive programming tasks, minimise human error and indirectly promote adherence to coding conventions.

Source module API In order to perform syntax checking and module scoping the plugin must build internal models of a GF module's source code using the Eclipse Modelling Framework (EMF) and the derived language infrastructure as described in section 1.2.2. These models are only used internally and not exposed via any API. However, having this level of access to GF modules could open up many interesting possibilities, including graphical tools for grammar writing and integration with ontology management software. Implementing such an interface to the plugin's inner modelling information is certainly possible, although the effort required could only be justified if an appreciable demand for such a feature was expressed.

1.4.3 Availability

The GF Eclipse Plugin is freely available and any be used for any purpose. It is open source and released under the GNU General Public License (GPL)[7] (note that Xtext and the Eclipse Platform are covered by the Eclipse Public License[8]).

The official GF Eclipse Plugin web page[9] contains installation instructions, a user guide and tutorial screencast, the plugin's release history and links to the project's source code repository and issue tracker.

[7] `http://www.gnu.org/licenses/gpl-3.0.txt`
[8] `http://www.eclipse.org/legal/epl-v10.html`
[9] `http://www.grammaticalframework.org/eclipse/`

Bibliography

Ranta, Aarne. 2009. The GF resource grammar library. *Linguistic Issues in Language Technology* 2(2).

Ranta, Aarne. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: CSLI Publications. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

The Eclipse Foundation. 2011. Xtext 2.1 documentation. `http://www.eclipse.org/Xtext/documentation/2_1_0/Xtext%202.1%20Documentation.pdf`. Accessed March 2012.

Chapter 2

Evaluating North Sámi to Norwegian assimilation RBMT

Trond Trosterud and Kevin Brubeck Unhammer
University of Tromsø, Kaldera språkteknologi

2.1 Introduction

2.1.1 Roadmap

We describe the development and evaluation of a rule-based machine translation (MT) assimilation system from North Sámi to Norwegian Bokmål[1], built on a combination of Free and Open Source Software (FOSS) resources: the Apertium platform and the Giellatekno HFST lexicon and Constraint Grammar disambiguator. We detail the integration of these and other resources in the system along with the construction of the lexical and structural transfer, and evaluate the translation quality using various methods, focusing on evaluating the users' *comprehension* of the text. Finally, some future work is suggested.

We begin with an introduction to the languages and the technology used, followed by a description of how the system was developed. Then we evaluate the system with respect to *assimilation*: MT with the purpose of letting users understand, or get the gist of, a text written in a language foreign to them (as opposed to *dissemination*, where the purpose is MT for post-editing). Finally, we discuss the results and ideas for improvements.

2.1.2 The Languages

North Sámi (`sme`) is a Finno-Ugric language spoken by between 15,000 and 25,000 people in the northern parts of Norway, Sweden and Finland. Norwegian Bokmål (`nob`) is a North Germanic language with about 4.5 million speakers, mostly in Norway. North Sámi is a

[1]Source code available from SVN repository `http://apertium.svn.sourceforge.net/svnroot/apertium/trunk/apertium-sme-nob` under the GNU General Public License.

highly inflected, agglutinative language, whereas Norwegian morphology is comparatively simple.

Most sme speakers in Norway understand nob, while most nob speakers do not understand sme. The languages are completely unrelated, and the linguistic distance is great, making it hard to achieve high quality MT results. For a nob→ sme system to be widely useful, the quality would have to be good enough that it could be used for text production (post-editing). On the other hand, a sme→ nob gisting-quality system (ie. assimilation system) can be useful for the large group of nob speakers who do not understand sme. Thus we chose to focus on the sme→ nob direction first.

We do not know of other machine translation (MT) systems between sme and any Indo-European language, although Tyers et al. (2009) describe a prototype system between North Sámi and Lule Sámi.

2.2 Design

2.2.1 The Apertium Pipeline

This language pair is based on the Apertium MT platform (Forcada et al., 2011, Zubizarreta et al., 2009). Apertium provides a highly modular, shallow-transfer pipeline MT engine, as well as data for language pairs. Both the engine and the data for all language pairs (about 30 released pairs as of now) are licensed under the GPL.[2]

Apertium language pairs are set up as Unix pipelines, where the typical pipeline consists of:

- deformatting (hiding formatting/markup from the engine),
- source-language (SL) morphological analysis with a finite state transducer (FST),
- disambiguation using a Hidden Markov Model (HMM) and/or Constraint Grammar (CG),
- lexical transfer (word-translation on the disambiguated source),
- one or more levels of finite-state based structural transfer (reordering, and changes to morphological features),
- target-language (TL) generation with an FST
- reformatting (letting format information be shown again)

See Figure 2.1 below for an overview of the modules used in this particular language pair. Most Apertium language pairs use the Apertium lttoolbox FST package for analysis and generation. The lttoolbox dictionaries are written in XML, where one dictionary may be compiled both to an analyser and a generator. The sme→ nob pair uses lttoolbox for nob generation and the translation dictionary, while the sme analyser is written in the Xerox lexc/twol formats (Beesley and Karttunen, 2003); the reason for this is explained in

[2] http://www.fsf.org/licensing/licenses/gpl.html

Section 2.2.2. Both systems allow generalising over classes using paradigms/continuation lexicons, but differ in other features. We use the FOSS package Helsinki Finite State Tools, HFST (Linden et al., 2011)[3] to compile and run the analyser (see Section 2.2.2).

The morphological analysis gives us ambiguous output with no syntactic information. For morphological (e.g. part-of-speech) disambiguation, syntactic annotation/disambiguation and lexical selection[4], we use Constraint Grammar (Karlsson, 1990)[5]. Morphological disambiguation and syntax are run as one CG module, the output of which is unambiguous both morphologically (one analysis per form) and syntactically (each form/analysis is annotated with exactly one syntactic tag, e.g. `<@SUBJ>`).

The first CG module[6] is directly followed by a lexical selection CG module, which may add subscripts to lemmas in certain contexts in order to select a different lexical translation.

To make this more concrete, the morphological analysis of the sentence *Mus lea biebmu vuoššat* "I have food to boil" is

```
^Mus/mun<Pron><Pers><Sg1><Loc>$ ^lea/leat<V><IV><Ind><Prs><Sg3>$
^biebmu/biebmat<V><TV><Imprt><Du1>/biebmu<N><Sg><Nom>$
^vuoššat/vuoššat<V><TV><Ind><Prs><Pl1>
/vuoššat<V><TV><Ind><Prs><Sg2> /vuoššat<V><TV><Inf>$,
```

read as

```
^form/lemma1<tags1>/lemma2<tags2>$.
```

The disambiguator removes the imperative "feed" reading of *biebmu* "food" by i.a. checking for the lack of left-hand clause boundaries or conjunctions. The finite readings of *vuoššat* "boil" are removed since there is a left-hand nominal with an unambiguous finite verb to its left. Then the readings have syntactic tags appended, e.g. *vuoššat* gets `<@←ADVL>` since it's an infinitive with a non-abstract nominative to the right.[7] Then the lexical selection module runs, the only change it makes is adding a subscript `:1` to *leat* "have/be" in order to select the "have" reading. Its output is

```
^Mun<Pron><Pers><Sg1><Loc><@HAB>$
^leat:1<V><IV><Ind><Prs><Sg3><@+FMAINV>$
^biebmu<N><Sg><Nom><@← SPRED>$
^vuoššat<V><TV><Inf><@← ADVL>$.
```

Lexical selection is followed by pretransfer (minor format changes in preparation of transfer) and then a four-stage chunking transfer. The first stage module first handles lexical transfer using the translation dictionary, and then performs chunking[8] based on patterns of morphological and syntactic tags (more on structural transfer in Section 2.3.6).

Output from the last transfer module is fed to morphological generation with the lttoolbox-based `nob` generator.

[3] `http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst/`

[4] Like Word Sense Disambiguation, but restricted to senses that have differing translations.

[5] Using the FOSS package `VISL CG-3`, `http://beta.visl.sdu.dk/cg3.html`

[6] If the disambiguation rules leaves any ambiguity, that CG only prints the first analysis. We may later train an HMM to get rid of leftover ambiguity, this would go between the two CG modules.

[7] The actual rules have several other context conditions.

[8] Newer Apertium language pairs have lexical transfer as a separate module before chunking. This is the plan for `sme→ nob` too, as it would allow matching on both SL and TL patterns, but the possibility was only recently added to the transfer engine.

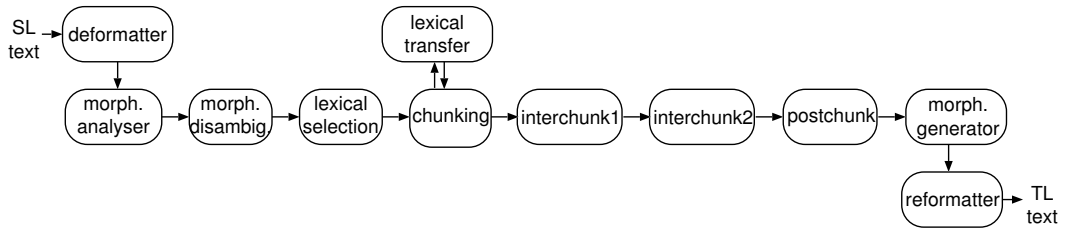


Figure 2.1: The Apertium pipeline architecture for `sme`→ `nob`.

2.2.2 HFST

One novel feature of apertium-sme-nob is the HFST-based analyser. HFST makes it possible to compile lexicons and morphologies originally written for the closed-source Xerox Finite State Tools using FOSS tools, and run them with Apertium-compatible output formats (Pirinen and Tyers, 2011). As with most Xerox-based analysers, the `sme` lexicon and morphology are written in **lexc** and compiled into an FST, onto which **twol** rules are composed which define the morphophonology. HFST analysers are slower at compiling and processing than lttoolbox, but certain morphological phenomena, especially non-concatenative phenomena (e.g. `sme` consonant gradation) are impossible—or at least very difficult—to describe in lttoolbox. Since North Sámi is quite morphologically complex, a pure lttoolbox analyser would be hard to maintain.

2.3 Development

This section describes how the language pair was developed.

2.3.1 Resources

We re-used several FOSS resources in creating this language pair. The `nob` generator came from `apertium-nn-nb` (Unhammer and Trosterud, 2009), while most of the `sme` resources came from the Divvun and Giellatekno Sámi language technology projects[9], including the lexicon/morphology and disambiguator/syntax CG. Although we altered our copies from the originals, we continually merged in the changes that were made in the "upstream" versions (ie. the ones maintaned by Divvun/Giellatekno).

The lexical selection CG and the transfer rules were written from scratch. The translational dictionary was originally based on various word-lists from Giellatekno but expanded throughout development.

2.3.2 Analysis and derivational morphology

The morphological analyser was not originally made for machine translation, and we made several modifications, from minor tag format changes, to restricting derivational morphology

[9]See `http://divvun.no` and `http://giellatekno.uit.no`.

and removing root forms without translations. Our modifications are all done automatically using scripts, letting us easily keep the analyser up-to-date with the upstream version.

The upstream analyser contains many lemmas and readings that are not in our translation dictionary. These often lead to transfer errors that can affect the surrounding context, and can suppress the choice of forms that do have translations. As an example of the latter, the form *vuovdi* (**salesperson**) gets a reading both as the underived noun, and as a derivation of the verb *vuovdit* (**to sell**); if both were in the analyser, but only the verb were in the translation dictionary, the disambiguator might still choose the noun, and we would end up with an untranslated word where we could have had a translation. Transfer errors in surrounding context occur with untrimmed analysers since the translation dictionary is also used to translate morphological features; e.g. the `nob` noun gender is necessarily specified per entry in the translation dictionary, and the transfer rules may insert gender-agreeing determiners based on the tags output from the translation dictionary. Writing heuristic exceptions for every possible tag omission would be more work than simply adding more good translations to the dictionary.

We "trim" the analyser down to those forms which are in the translation dictionary. To do this, we use a script which analyses the lexc source files with the translation dictionary, and outputs only those entries which have translation analyses.[10]

The original analyser defines quite a lot of rules for derivational processes. Derivational morphology expands the coverage of the analyser without having to add new root forms (lexicalisation), but also makes transfer much harder to deal with, as well as often giving very odd-sounding translations. To give an example of the latter, 'geafivuohta' is an adjective→ noun derivation of 'geafi', meaning 'poor'. Simply carrying over the information that this is an adjective→ noun derivation into the target language dictionary (if that dictionary also defined derivational processes) could give us forms that sound like 'poorness' or 'poority' or 'poordom', whereas giving 'geafivuohta' status as a real lexicalised root form would let us specify that 'geafivuohta' should translate to 'poverty'. If we did not use derivations, 'geafivuohta' would either be lexicalised and translated to 'poverty', or not translated at all.

Derivations also create extra transfer complexity. A causative verb derivation requires transfer rules that turn the causative verb into a periphrastic construction (e.g. 'let NP VERB'). If a derivation changes the part-of-speech from verb to noun, we have to translate the derivation into a certain verb form that looks right in a noun context (e.g. present tense of `nob` verbs will most of the time look like an actor noun).[11] To make this even more complex, even a lexicalised form might require a part-of-speech change in the translation dictionary if there is no word with the same meaning and part-of-speech in the target language. The most natural translation to `nob` of the verb *muittohuvvat* ("become forgetful") would be to an auxiliary + adjective, *bli glemsk*, and this is what the translational dictionary specifies. But *muittohuvvat* is not a dynamically formed derivation, it has a regular entry in the analyser, so we can also form derivations of it. This means that we also have to ensure that transfer works for all possible derivations that the analyser can make, combined with all possible

[10]The untrimmed source files weigh in at about 3.5 MB, trimming this down to 2.6 MB is done by a script in our public repository which runs in < 10 seconds, and is general enough that other lexc-based language pairs can easily use it.

[11]The alternative would be to define, for each `sme` verb, both a noun and a verb translation on the `nob` side of the translation dictionary, but this takes away the whole point of increasing coverage without adding all the root forms.

part-of-speech changes specified in the translational dictionary.

However, since `sme→ nob` is meant for gisting, where an odd-sounding translation is more useful than an untranslated word, and resources for automatically expanding the translation dictionary are scarce, we decided to allow a restricted set of derivations. We define legal derivations by the use of additional twol rules which simply forbid analyses containing certain tags or tag sequences. These twol rules are composed onto the main analyser in Apertium, but not used upstream.

2.3.3 Disambiguation

The CG created by Giellatekno was usable in Apertium with only minor changes to tag naming, requiring very little manual intervention to keep up-to-date. However, we did add Apertium-only rules which remove derivations and compound readings if there are lexicalised readings available, since we want lexically specified translations to override the guesswork done by derivation transfer. Certain discrepancies in the tag set of the analyser still exist though, which may affect disambiguation quality.

2.3.4 Lexical selection

A lexical selection CG was created in order to select between different possible translations that otherwise share the same part-of-speech information. Currently it has only 102 rules covering 52 lemmas, mostly high-frequency ones (although 750 other lemmas of the translation dictionary have at least one alternative translation, and are awaiting rules). This CG particularly depends on valency and semantic sets,[12] e.g. *luohkká* by default translates into *bakke*, "hill", but if we see a context word related to the EDUCATION set, we translate into *klasse*, "(school) class".

2.3.5 Lexical transfer

The open classes of the translation dictionary were initiated with entries from the 9900 lemma dictionary Vuosttaš Digisánit,[13] although many "explanatory" multiword translations had to be removed or simplified[14]. Later on, entries were mostly added manually. Not including lexical selection alternatives, there are currently about 3300 verbs, 1400 adjectives and 14000 common nouns in the translation dictionary.

Unlike with most Apertium language pairs, we did not make an attempt to change the tag set in the analyser to conform with the Apertium standard (apart from minor format differences). The change from e.g. `<N><Prop>` (proper noun) to `<np>` or `<Sg1>` to `<sg><p1>` happens in the translation dictionary, mostly using a **paradigm** definition to generalise over changes for each part of speech. Part of the derivation handling also happens here, e.g. most passive derivations turn into plain passive forms, while verbs derived into actor-nouns are transferred to present tense verbs.

[12] The sets themselves were originally developed by Giellatekno for use in the disambiguator.

[13] GPL and CC, see `http://giellatekno.uit.no/words/dicts/index.eng.html`.

[14] The word *madda* might be translated as "branching part of deer's antlers" in a human-readable dictionary, but in an MT dictionary it has obvious problems – it sounds over-specific and is difficult to wedge into all possible grammatical contexts.

We also add special tags used only as a signal to structural transfer, which are removed before generation. The causative derivation of a word gets a tag which signals structural transfer to create a periphrastic 'let'-construction, but we also add the same tag to all forms if the root itself is a lexicalised causative. E.g. *čálihit* "let write" is a lexicalised causative with the lemma *čállit*; since it is lexicalised, the infinitive is simply tagged `<V><TV><Inf>`; since there's no good lexicalised translation to `nob`, we translate this lemma (and thus all its forms) to *skrive* "write" along with the tag `<caus>` in the translational dictionary. Structural transfer removes `<caus>` and outputs *la* "let" (putting the main verb after the causee), and the `nob` generator never sees any `<caus>` tag. If *čálihit* were analysed as a dynamic derivation of *čállit* "write", the lemma would be *čállit*, while the tag sequence would be `<V><TV><Der_h><V><TV><Inf>` (the "h-derivation" `<Der_h>` is a causative derivation). In that case we wouldn't mark the lemma (ie. all forms) with `<caus>`, but a **paradigm** for tag translation would add `<caus>` only if the tag sequence contained `<Der_h>`.

Verbs are also tagged in the translational dictionary according to the most likely animacy of the agent[15] as a signal to structural transfer; `sme` often omits subject pronouns (**pro-drop**), so when translating to `nob` and inserting a pronoun we need to know whether the inserted pronoun should be animate or not.

2.3.6 Structural transfer

Our structural transfer is divided into four stages, with different responsibilities:

1. Chunking, 63 rules: noun phrases turn into larger chunks, prepositions are output based on case information, verb auxiliaries and adverbs are output based on verb modality, voice and derivation tags.
2. Interchunk 1, 26 rules: simple anaphora resolution (based on most recent subject gender), merging coordinated noun phrase chunks, moving postpositions before noun phrases.
3. Interchunk 2, 39 rules: major word order changes, inserting dropped pronouns, inserting adverbs to indicate verb modality, correcting noun phrase definiteness using verb information (e.g. subjects of duals are definite).
4. Postchunk, 29 rules: inserting articles/determiners and the infinitive marker, tag cleanup in preparation of generation.

Wherever generalisations are possible, we use **macros** (e.g. for tranferring agreement information), so rules tend to be fairly short. A lot of work went into structural transfer compared to what is typical of Apertium language pairs between more related languages; e.g. the translator for the closely-related pair Bokmål→ Nynorsk (Unhammer and Trosterud, 2009), is quite mature and achieves post-edit quality translations with 2745 lines of structural transfer code and 107 lines of tag transfer paradigms, whereas the corresponding numbers for `sme`→ `nob` are 9556 and 1002[16].

[15] Currently just manual tagging, a corpus-based method should be possible with the use of semantic CG sets like HUMAN.

[16] Lines of code of course does not correspond one-to-one with amount of work, but since the same people did the bulk of the work, the numbers should be fairly comparable with each other. All numbers are from SVN revision 38590.

Corpus	tokens	coverage	ambig. rate	coverage w/o deriv	ambig.rate w/o deriv
laws	51706	94.68%	2.65	86.02%	2.32
wiki	19942	77.52%	2.36	74.56%	2.19
news	1020250	94.72%	2.59	90.96%	2.34

Table 2.1: Naïve coverage on several corpora.

2.3.7 Generation

The generator was re-used from the language pair `apertium-nn-nb` with very few changes: We added some root forms to the lexicon, and added a tag to distinguish synthetic from analytic adjectives (a change which might later be useful in improving `apertium-nn-nb`).

2.4 Evaluation

The naïve coverage[17] of the analyser is shown in Table 2.1 for legal text (laws), the `sme` Wikipedia (wiki) and a corpus of `sme` news articles. All forms that pass through the analyser, will also pass through the translation dictionary, transfer rules and generator, so this shows the coverage of the other dictionaries (in the `sme`→ `nob` direction) as well. Since derivations are not specified in the translation dictionary, we show coverage with and without derivation-only analyses counted. The table also shows the ambiguity rate (amount of analyses per known word) with and without derivations counted.[18]

The Wikipedia corpus seems to have very low coverage, but looking over the unknown words, it seems that many of them are in Finnish, English or Norwegian (the rest are mostly proper names). The Sámi Wikipedia is also written by non-natives, 12.5% of its words are not recognised even by Giellatekno's non-normative analyser, as opposed to only 3.5 % for a larger, 6.1m reference corpus. The lower coverage for Wikipedia is thus to be expected.

In the rest of this section we evaluate the practical performance of the system using several methods. First we do a word-error rate test, which shows how well the system would perform in a post-editing/dissemination setting, then a set of tests meant to find out how well the system performs in a gisting/assimilation setting. All tests were run on revision 37177 of `apertium-sme-nob`[19].

[17] A form is counted as covered if it gets at least one analysis. It might have ambiguity which the analyser does not cover, thus 'naïve'.

[18] Currently, the ambiguity rate is reduced to about 1.04 by the CG disambiguator; in actual translations we set the module to simply choose the first analysis when there is remaining ambiguity.

[19] At which point the repository address was
`http://apertium.svn.sourceforge.net/svnroot/apertium/staging/apertium-sme-nob`

Text	tokens	Unknown	WER
children's	415	5	45.96%
history	435	28	60.32%

Table 2.2: Word error rate on two short texts.

2.4.1 Word Error Rate on Post-Edited text

We did a Word Error Rate test on a short children's story[20] and some paragraphs from a history web page.[21] The results are shown in Table 2.2.[22] The translator obviously struggles with the more complex formulations in the history text, and has a long way to go before being useful for post-editing.

2.4.2 Gisting evaluation

In order to evaluate to what extent the system was able to convey the meaning of the original to human users, we arranged a test containing 3 parts. All the tests were based on sentences from a parallel corpus of non-fiction, the corpus had not been used during development of the MT system. None of the test subjects had any knowledge of `sme`.

The first test, a multiple choice test, presented 10 `sme` sentences drawn from the corpus. For each sentence, the test person also got the MT output, along with 3 alternative hand-written `nob` paraphrases (based on the `sme` sentence set). Only one of the three paraphrases was paraphrasing that `sme` sentence correctly (the other two were written to be similar, but contain factual mistakes), and the subject had to use the MT output as a guide to pick which paraphrase corresponded with the original `sme` sentence. Example (1) shows the test for one of the sentences[23], where the subject could pick one of a., b. or c., the correct choice being b. Since the other paraphrases were kept as close as possible to the meaning of the `sme` sentence, but had changes that crucially altered the semantics (e.g. removing the negation, changing the main content word), an incorrect choice should indicate that the translation was inadequate in providing understanding.

(1) `Original:` Muhto eat diehtán maid mii čáliimet ('But we didn't know what we wrote')
`Translated:` Men vi visst ikke også skrev vi ('But we didn't known also wrote we')
`Pick the right alternative:`
a. Vi visste hva vi skrev ('We knew what we wrote')
b. Vi visste ikke hva vi skrev ('We didn't know what we wrote')
c. Vi visste ikke hva de skrev ('We didn't know what they wrote')

The second test was set up as the first, but instead of the 3 alternatives, the test presented an open question to be answered using the MT as a guide. This we assumed to test understanding as in Test 1, but with more reliance on the meaning of content words.

[20] `https://apertium.svn.sourceforge.net/svnroot/apertium/branches/xupaixkar/rasskaz`
[21] `http://skuvla.info/skolehist/siri97-s.htm`
[22] Our post-edits are available from our public repository.
[23] English glosses were of course not shown; note that we keep un-/mistranslated terms untranslated in the glosses to indicate what the user actually saw.

(2) `Original:` Goappaš riikkain lea nammaduvvon hálddahuslaš gulahallanolmmoš ('There is appointed an administrative contact from both countries')
`Translated:` Goappaš på rikene er det oppnevnt administrativt forstående hverandre seg mennesket ('Goappaš on the countries there is appointed an administrative understanding eachother self person')
`Answer the question:` Hvor kommer kontaktpersonene fra? ('Where do the contact persons come from?')

For both test sets the paraphrases / questions were prepared on the basis of the `sme` sentence, before they were translated by the system, in order not to be influenced by the translated output.

The third test showed a `sme` source sentence, then the MT output of that sentence, followed by the reference translation (5-15 words long) where at least two of the nouns were removed. For each removed noun, we instead showed a randomised, clickable list consisting of the originally removed word, along with a random choice of other nouns[24] and finally a "none seem to fit" choice. The subjects were instructed to click what seemed to be the removed word, using the MT as a guide. Ten consecutive sentences from the same piece of text were shown one at a time. The test should indicate whether the main content/theme (though perhaps not the truth conditions) are sufficiently well understood; the MT output might use the same word used in the answer alternatives, or one quite similar, or the context might be well enough translated that the correct alternative seems obvious. Example (3) below shows one of the test sentences, where two word choices had to be made (in this case both key words happened to translate correctly). All tests were performed using simple HTML forms.

The results of all tests are shown in Table 2.3.

Table 2.3: Results of gisting evaluation, 3 different tests

Type	Multiple	Fill-in	Random
Result	77 %	41 %	75 %
Number of test subjects	10	14	10

The results from the multiple choice and random word tests correlate with each other, whereas the fill-in test seems much worse. Open questions don't allow "correct guesses", and the fill-in test was more vulnerable to holes in the MT output. Four of the 10 test sentences got no or only one correct answers. It seems that what made these sentences so hard to understand, was that the system failed to translate the key word in the sentence. Sentence (3), with `nob` MT output in (4), gives an example.

[24]Nouns were of the same length (±3 characters), pulled from the same 55128 word long legal text, had the same morphological features (gender, definiteness, number) and were never ambiguous with verbs. This questionnaire generator is available from our public repository (subdirectory `gisting-eval/generated`) and should be usable with other translators.

(3) ... Lillemor Isaksen, geas lei gymnása, measta ii
... Lillemor Isaksen, who had secondary.school, almost did.not
fidnen oahpaheaddjibarggu go son lei sámegielat
get teacher.work since she was **saami-speaking**
'... Lillemor Isaksen, who had secondary school, almost did not get any work as a teacher, since she was **saami-speaking**'

(4) ... Lillemor Isaksen, som hadde ∗gymnása, nesten ikke
... Lillemor Isaksen, who had ∗gymnása, almost not
han fått lærerarbeidet da han var samisker
he gotten the.teacher.work since he was **saamier**
'... Lillemor Isaksen, who had secondary school, almost did he not get the teacher work, since he was **a saamier**'

The word *samisker* (here most likely to be interpreted as an agent noun, on a par with **carpenter**) does not exist, and is interpreted by 9 of the informants as "a Saami", instead of the correct "a Saami speaker".[25] Here, the mistranslation of the key word blocked a proper understanding of the sentence, despite the rest of the sentence being translated correctly. The loan word *gymnása* was not recognised (here marked by a star), but understandable as it is a loan from Norwegian (*gymnas*).

2.4.3 Error analysis

The naïve lexical coverage of the test sentences was good, 96.7%, as compared to the coverage measured on our news corpus (91%). With an average sentence length of 14.5 words (as in our test set), the coverage implies one lexical omission in every second sentence. For some words, our analysis didn't include all likely readings (e.g. *muhte* should be ambiguous between a subjunction and a verb, we only had the verb). In other cases, short (but non-compositional) idioms were treated as compositional individual words. A lot of anaphora get the wrong gender, but it's hard to tell how badly this affects comprehension.

For 3 of the 10 fill-in test sentences the key word (the topic of the question to be addressed) was not translated. This illuminates the importance of a good lexical coverage: On average, 95% coverage implies one error for each sentence. Also, the pivotal word in the discourse is likely to carry new meaning, but also be new to the system.

Another challenge is the erroneous insertion of pronouns in pro-drop sentences. This is more of a problem for dissemination than for assimilation, but in certain cases the superfluous pronouns may break the causality chain of the sentence, as in (5), with `nob` MT output in (6):

(5) ... dieđihuvvo ahte dušše sullii 1/3
... is.informed that only approximately 1/3
skuvllageatnegahtton ohppiin bohte oahpahussii
school.duty with.teacher came to.class
'... tells us that only about 1/3 of those of school age with a teacher came to class'

[25] Neither interpretation was in the analyser (trimmed or not); the analysis given was the plural of the language name, and plurals of common-gendered nouns have the same suffix as singular agent nouns in `nob`, while language names are ambiguous with nationality names.

(6) ... meddeles han at bare omtrent 1/3
... is.informed he that only approximately 1/3
∗skuvllageatnegahtton med en lærer kom **de** til undervisning
∗skuvllageatnegahtton with a teacher came **they** to class
'... is he informed that only about 1/3 skuvllageatnegahtton
with a teacher they came to class'

The subject status of (the untranslated) *1/3 skuvllageatnegahtton* is blocked by the insertion of *de* ("they"). Thus, the otherwise probable (and correct) interpretation (only 1/3 of X came to class) is suddenly less likely to be detected, and it is missed by 8 of our informants, several of whom interpret the sentence as describing a situation where the teacher does not show up.

Some grammatical constructions were too complicated for the system, like the sentence

(7) Jos ii lean vejolaš váhnemiid lusa vuolgit, de ...
if not was possible to.parents to.PO travel, then ...
'If it was not possible to travel to the parents, then ...'

The system interpreted this as a 3rd person pro-drop, and translated as follows:

(8) Hvis han ikke hadde til de mulige foreldrene reiser ,..
If he not had to.PR the possible parents travels, ...
'If he had not to the possible parents travels, ...'

but the correct interpretation is one of a formal subject of what in Norwegian would have been a cleft construction.

In itself, the erroneous translation in (8) would probably be understandable, but as part of an causal if-X-then-Y construction, it proved too difficult for half of the informants. What is needed here is thus better handling of the grammatical construction in question.

In the WER tests, we see some errors that are due to our translator over-specifying, e.g. using "the two" as a subject for dual verbs where "they" might be more natural. But for a gisting translator, over-specific translation is a feature, not a bug.

The ambiguity rate after the disambiguation rules have run is quite low, but on the other hand we get many erroneous readings, especially for high-frequency function words (e.g. *maid*, **also**/**what**/**that**). It is also obvious that we need more lexical selection rules; sometimes the translations simply sound non-fluent, but in other cases the meaning is altered or lost. E.g. *lohkat* can mean either **say** (as in "Go, he said") or **count** (as in "I counted to three"), picking the wrong word here severely hurts understanding.

The more complex the text, the more we see problems relating to structural transfer; sometimes we simply do not catch large enough NP chunks (since we only match fixed-length patterns, they turn into two chunks instead of one).

2.5 Discussion and outlook

Currently, the results of the evaluation of the assimilation indicate that the MT output provides some help for non-Sámi speakers in understanding North Sámi, but as the results of the fill-in sentence test showed, users miss important points from isolated sentences at least.

Both transfer rules and lexical selection could be better. There is an experimental Apertium package apertium-lex-tools that we plan to use to automatically create more lexical selection rules. Disambiguation might be improved by training an HMM to run after the rule-based disambiguator, although the ambiguity rate is already well reduced by the rules that are in place.[26]

The naïve coverage is very good on paper, even disregarding derivations, but on the other hand, with a system meant for gisting, one missing word can take away any chance of understanding the sentence.

Acknowledgements

Development was funded by the Norwegian Ministry of Government Administration, Reform and Church Affairs. Many thanks to Francis Tyers, Lene Antonsen, Linda Wiechetek, Berit Eskonsipo, and Sjur Moshagen, who also contributed much to the development of this language pair.

[26] An HMM after CG would not help with the analyses that are erroneously removed, only those that are erroneously left ambiguous

Bibliography

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite state morphology*. Stanford, Calif.: CSLI Publications. ISBN 1-57586-433-9.

Forcada, Mikel L., Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation* 25(2):127–144.

Karlsson, F. 1990. Constraint Grammar as a Framework for Parsing Running Text. In H. Karlgren, ed., *Proceedings of the 13th Conference on Computational Linguistics*, vol. 3, pages 168–173. Helsinki, Finland.

Linden, Krister, Miikka Silfverberg, Erik Axelson, Sam Hardwick, and Tommi Pirinen. 2011. *HFST—Framework for Compiling and Applying Morphologies*, vol. Vol. 100 of *Communications in Computer and Information Science*, pages 67–85. ISBN 978-3-642-23137-7.

Pirinen, T.A. and F.M. Tyers. 2011. Compiling and using Apertium morphological dictionaries with HFST. (to appear).

Tyers, F.M., L. Wiechetek, and T. Trosterud. 2009. Developing Prototypes for Machine Translation between Two Sámi Languages. In L. Màrquez and H. Somers, eds., *EAMT-2009*, pages 120–127. Barcelona, Spain: Universitat Politècnica de Catalunya.

Unhammer, Kevin Brubeck and Trond Trosterud. 2009. Reuse of Free Resources in Machine Translation between Nynorsk and Bokmål. In J. A. Pérez-Ortiz, F. Sánchez-Martínez, and F. M. Tyers, eds., *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 35–42. Alicante: Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos.

Zubizarreta, Mikel L. Forcada, Francis M. Tyers, and Gema Ramírez-Sánchez. 2009. The Apertium machine translation platform: five years on. In J. A. Pérez-Ortiz, F. Sánchez-Martínez, and F. M. Tyers, eds., *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 3–10. Departamento de Lenguajes y Sistemas Informáticos, Alicante: Universidad de Alicante.

Chapter 3

Choosing the correct paradigm for unknown words in rule-based machine translation systems

V. M. Sánchez-Cartagena, M. Esplà-Gomis, F. Sánchez-Martínez, J. A. Pérez-Ortiz

Universitat d'Alacant

Abstract

Previous work on an interactive system aimed at helping non-expert users to enlarge the monolingual dictionaries of rule-based machine translation (MT) systems worked by discarding those inflection paradigms that cannot generate a set of inflected word forms validated by the user. This method, however, cannot deal with the common case where a set of different paradigms generate exactly the same set of inflected word forms, although with different inflection information attached. In this paper, we propose the use of an n-gram-based model of lexical categories and inflection information to select a single paradigm in cases where more than one paradigm generates the same set of word forms. Results obtained with a Spanish monolingual dictionary show that the correct paradigm is chosen for around 75% of the unknown words, thus making the resulting system (available under an open-source license) of valuable help to enlarge the monolingual dictionaries used in MT involving non-expert users without technical linguistic knowledge.

3.1 Introduction

Rule-based machine translation (MT) systems heavily depend on explicit linguistic data such as monolingual dictionaries, bilingual dictionaries, grammars, etc. (Hutchins and Somers, 1992). Although the acquisition of these data has usually required the intervention of linguists, development costs could be significantly reduced by involving a broader group of

non-expert users in the enrichment of these MT systems. This group may include, for instance, people using an online translation service who want to improve it by adding an unknown word to the underlying MT system dictionaries, or collaborators recruited through crowdsourcing (Wang et al., 2010) platforms.

In previous works (Esplà-Gomis et al., 2011, Sánchez-Cartagena et al., 2012) we proposed a novel method for enlarging the two monolingual dictionaries and the bilingual dictionary of shallow-transfer rule-based MT systems with the collaboration of non-expert users. In the case of a monolingual dictionary, adding a new entry implies determining the stem of the new word and a suitable inflection *paradigm* among those defined by the MT system for the corresponding language. Paradigms are commonly introduced to group regularities in inflection which are common to a set of words; for instance, the paradigm assigned to many common English verbs, indicates that by adding *-ing* to the stem, the gerund is obtained; by adding *-ed*, the past is obtained; and so on. In our approach, the most appropriate stem/paradigm combination is chosen by means of a sequence of simple yes/no questions whose answer only requires *speaker-level* understanding of the language. Basically, users are asked to validate whether the forms resulting from tentatively assigning different candidate paradigms to the new word to be inserted are correct inflected forms of it. The experiments we performed showed (Esplà-Gomis et al., 2011) that the average number of queries posed to the users for a Spanish monolingual dictionary was 5.2, which is reasonably small considering the 56.4 initial compatible paradigms on average.

The whole procedure for adding an unknown word and its translation to all the MT system dictionaries could consequently consist of requesting users a source-language word and its corresponding translation into target language (for instance, *cars* and *coches*, for an English–Spanish MT system). Then, our method could be independently applied to the source-language word and its target-language translation to obtain their inflection paradigms and insert all this information into the monolingual dictionaries. Finally, the corresponding link between both words could be inserted in the bilingual dictionary in a straightforward manner without any additional user interaction. Moreover, we have shown (Sánchez-Cartagena et al., 2012) that when the source-language word has been already inserted, the system is able to more accurately predict the right target-language paradigm by exploiting the correlations between paradigms in both languages, thus reducing significantly the number of queries posed to the user. Note that, although when the source language and the target language are not closely related the correlation between paradigms is not very strong, previous experiments performed with the English–Spanish language pair (Sánchez-Cartagena et al., 2012) have shown that the source-language part of speech is still useful to reduce the number of queries posed when inserting the target-language word.

Our proposal provided a complete framework for dictionary enlargement, but it still lacked a critical component to discriminate between paradigms providing the same set of inflected word forms. It turns out that by only asking users whether a set of word forms are correct forms of the word to be inserted, our system frequently ends up with more than one feasible stem/paradigm solution and, since all of them generate the same set of inflected word forms, no additional query can be posed to the user in order to discriminate between them. For example, in the case of Spanish, the inflected word forms for many adjectives such as *alto* (*tall* in English) are *alt/o* (masculine, singular), *alt/a* (feminine, singular), *alt/os*

(masculine, plural), *alt/as* (feminine, plural);[1] therefore, dictionaries contain a paradigm for adjectives with suffixes {*-o,-a,-os,-as*} to which the stem *alt-*, among others, will be assigned. Additionally, the inflected word forms for many nouns such as *gato* (*cat* in English) are *gat/o* (masculine, singular), *gat/a* (feminine, singular), *gat/os* (masculine, plural), *gat/as* (feminine, plural); consequently, dictionaries contain a paradigm for nouns with suffixes {*-o,-a,-os,-as*} to which stems such as *gat-* will be assigned. As can be seen, in the case of adding an unknown word such as the noun *perro* (*dog* in English), which inflects as *gato*, no yes/no question may be presented to the user to discriminate between the two paradigms (which are equivalent for the interactive method) given the stem *perr-*.

Note that the problem shows similarities to that of part-of-speech tagging (Manning and Schütze, 1999) and it can be addressed through similar approaches, but in our case we also need to disambiguate between equivalent paradigms involving the same lexical category. For instance, the Spanish inflected forms for nouns such as *abeja* (*bee* in English) are *abeja/*ϵ[2] (feminine, singular), *abeja/s* (feminine, plural); and the inflected word forms for the noun *abismo* (*abyss* in English) comprise *abismo/*ϵ (masculine, singular), *abismo/s* (masculine, plural). Therefore, these two words are assigned to equivalent paradigms, both for the noun lexical category with suffixes {ϵ,*-s*}, but with different inflection information (gender). A new noun such as *taza* (*cup* in English) would in principle fit into both paradigms.

Although in this paper we give a fully automatic solution to the multiple step/paradigm issue, an interactive approach could also be followed. Users may be asked to validate some sentences in which the word to be classified would contain the inflecion information from each paradigm. For instance, one possible strategy for eliciting the gender of *taza* would be to ask the user to validate the sentences *el taza* and *la taza*, being *el* a masculine determiner and *la* a feminine determiner.

Our automatic solution is obtained by introducing an n-gram-based model of lexical categories and inflection information which is used to automatically choose the right stem/paradigm combination: nouns belonging to the same paradigm as *abeja* will be usually preceded by a feminine determiner in a corpus, whereas nouns to be assigned to the same paradigm as *abismo* will be frequently preceded by a masculine determiner.

The model is trained with a monolingual corpus where every word is replaced by its morphological analysis comprising lexical category and inflection information. The Java code for the resulting system is available under the free/open-source GNU General Public License[3] and may be downloaded from `https://apertium.svn.sourceforge.net/svnroot/apertium/branches/dictionary-enlargement`.

In the experiments we have used the free/open-source rule-based MT system Apertium (Forcada et al., 2011), which is being currently used to build MT systems for a large variety of language pairs. In the case of the Spanish monolingual dictionary used in the Spanish–Catalan Apertium MT system, 81.1% of the words would be assigned by the original method to more than one equivalent paradigm; as a result, giving a solution to the multiple paradigm issue is critical.

The rest of the paper is organised as follows. Section 3.2 discusses other works related to our proposal. Section 3.3 introduces some concepts about monolingual dictionaries which

[1] In this paper, we use the slash character to separate the stem of a word from the suffix of one of its possible inflections.

[2] Symbol ϵ denotes the empty string.

[3] `http://www.gnu.org/licenses/gpl.html`

will be used in the rest of the paper. An overview of the previous method for dictionary enlargement is presented in section 3.4, followed by the description of our new improvement for discriminating between paradigms generating the same inflected forms in section 3.5. Section 3.6 discusses our experimental setting. Then, the results obtained are presented and discussed in section 3.7. Finally, some concluding remarks are presented in section 3.8.

3.2 Related work

Two of the more prominent works related to the elicitation of knowledge for building or improving MT systems are those by Font-Llitjós (2007) and McShane et al. (2002). The former proposes a strategy for improving both transfer rules and dictionaries by analysing the postediting process performed by a non-expert user through a special interface. McShane et al. (2002) design a complex framework to elicit linguistic knowledge from informants who are not trained linguists and use this information to build MT systems which translate into English; their system provides users with a lot of information about different linguistic phenomena to ease the elicitation task.

Unlike the Avenue formalism used in the work by Font-Llitjós (2007), the MT system we are using is a *pure* transfer-based one in the sense that a single translation is generated and no language model is used to score a set of possible candidate translations; therefore, we are interested in a single correct solution and assume that an incorrect paradigm cannot be assigned to a new word. Unlike the works by McShane et al. (2002) or Bartusková and Sedlácek (2002), we want to relieve users of acquiring linguistic skills.

3.3 Monolingual dictionaries in rule-based MT systems

As already pointed out, monolingual dictionaries have two types of data: *paradigms*, that group regularities in inflection, and *word entries*, represented by a stem and a paradigm. The *stem* is the part of a word that is common to all its inflected variants. Paradigms make easier the management of dictionaries in two ways: by reducing the quantity of information that needs to be stored, and by simplifying revision and validation thanks to the explicit encoding of regularities in the dictionary. Once the most frequent paradigms in a dictionary are defined, entering a new word is generally limited to writing the stem and choosing an inflection paradigm. In this work we assume that all the paradigms for the words in the language are already included in the dictionary.

Let $P = \{p_i\}$ be the set of paradigms in a monolingual dictionary. Each paradigm p_i defines a set F_i of pairs (f_{ij}, m_{ij}), where f_{ij} is a suffix[4] which is appended to stems to build new *inflected word forms* (IWFs), and m_{ij} is the corresponding morphological information.

Given a *stem/paradigm* pair c composed of a stem t and a paradigm p_i, the *expansion* $I(t, p_i)$ is the set of possible IWFs resulting from appending each of the suffixes in F_i to t. For instance, an English dictionary may contain the stem *want-* assigned to a paradigm

[4]Although our approach focuses on languages generating word forms by adding suffixes to the stems of words (for example, Romance languages), it could be easily adapted to inflectional languages based on different ways of adding morphemes as long as this kind of inflection is encoded in paradigms; note that a data structure different from a suffix tree (see section 3.4) may be needed.

with suffixes[5] $F_i = \{\epsilon$,*-s*, *-ed*, *-ing*$\}$ (ϵ denotes the empty string); the expansion $I(\text{want}, p_i)$ consists of the set of IWFs *want*, *wants*, *wanted* and *wanting*. We also define a *candidate stem* t as an element of $\Pr(w)$, the set of possible prefixes of a particular IWF w.

3.4 Original method

As our new proposal is a refinement over our previous method (Esplà-Gomis et al., 2011) for adding new entries to the monolingual dictionaries of an MT system, a brief description of it follows before presenting the main contribution of this paper in section 3.5.

Given a new IWF w to be added to a monolingual dictionary, our objective is to find both the candidate stem $t \in \Pr(w)$ and the paradigm p_i which expand to the largest possible set of IWFs which are correct forms of w. To that end, our method performs these three tasks: paradigm detection, paradigm scoring, and user interaction.

Paradigm detection. To detect the set of paradigms which may produce the IWF w and their corresponding stems we use a *generalised suffix tree* (McCreight, 1976) containing all the possible suffixes included in the paradigms in P. A list L is built containing all the candidate stem/paradigm pairs compatible with the IWF to be added (candidate paradigms, CPs). We will denote each of these candidates as c_n.

The following example illustrates this stage of our method. Consider a simple dictionary with only four paradigms: p_1, with F_1=$\{f_{11}$=ϵ, f_{12}=*-s*$\}$; p_2, with F_2=$\{f_{21}$=*-y*, f_{22}=*-ies*$\}$; p_3, with F_3=$\{f_{31}$=*-y*, f_{32}=*-ies*, f_{33}=*-ied*, f_{34}=*-ying*$\}$; and p_4, with F_4=$\{f_{41}$=*-a*, f_{42}=*-um*$\}$. Let's assume that a user wants to add the new IWF w=*policies* (actually, the noun *policy*) to the dictionary. The candidate stem/paradigm pairs which will be obtained after this stage are: c_1=*policies*/p_1; c_2=*policie*/p_1; c_3=*polic*/p_2; and c_4=*polic*/p_3.

Paradigm scoring. Once L is obtained, a *confidence score* is computed for each CP $c_n \in L$ using a large monolingual corpus C. Candidates producing a set of IWFs which occur more frequently in the corpus get higher scores.

Following our example, the IWFs for the different candidates would be: $I(c_1)$=$\{$*policies*, *policiess*$\}$; $I(c_2)$=$\{$*policie*, *policies*$\}$; $I(c_3)$=$\{$*policy*, *policies*$\}$; and $I(c_4)$=$\{$*policy*, *policies*, *policied*, *policying*$\}$. Using a large English corpus, IWFs *policies* and *policy* will be easily found, and the rest of them (*policie*, *policiess*, *policied* and *policying*) probably will not. Therefore, c_3 would obtain the highest score.

User interaction. Finally, the best candidate is chosen from L by querying the user about a reduced set of the IWFs for some of the CPs $c_n \in L$. In this way, when an IWF w' is accepted by the user, all $c_n \in L$ for which $w' \notin I(c_n)$ are removed from L; otherwise, all $c_n \in L$ for which $w' \in I(c_n)$ are removed from L.

In order to try to maximize the number of IWFs discarded in each query and, consequently, minimize the amount of yes/no questions, our system firstly sorts L in descending order using the confidence score previously computed. Then, users are asked to confirm whether the IWF from the first CP in L which exists in the minimum number of other CPs

[5] We omit the morphological information contained in F_i and show only the suffixes.

in L is a correct form of w. This process is repeated until only one candidate or a set of equivalent paradigms remain in L.

3.5 Improvement to the method

The original method presented in the previous section has an important limitation: the system frequently ends up with more than one stem/paradigm proposal. All these final candidates generate the same set of inflected word forms, although with some variation in the lexical category or in the inflection information, and no additional query can be posed to the user in order to discriminate between them (see the introduction for some examples in Spanish).

As an empirical evidence of the importance of that limitation, we found that when trying to find the most appropriate paradigm for a representative set[6] of the words already inserted in the Spanish monolingual dictionary of the Apertium Spanish–Catalan[7] language pair, 81.1% of the entries would be assigned more than one stem/paradigm pair after users answered correctly all the queries posed by the original system described in section 3.4.

Each of the different sets of equivalent CPs which can be assigned to one or more entries in the monolingual dictionary constitute a *paradigm class*. Table 3.1 shows, for each of the 6 paradigm classes with most words in the Spanish dictionary, the number of entries which would be assigned to them after the original method, the number of different CPs in the final list, and an example word for every CP.

Our hypothesis is that a probabilistic model of lexical categories and inflection information could prove very useful to find the correct paradigm in the paradigm class. For instance, as already commented in the introduction, nouns belonging to the same paradigm as *abeja* will be usually preceded by a feminine determiner in a corpus, whereas nouns to be assigned to the same paradigm as *abismo* will be frequently preceded by a masculine determiner.

Consequently, we propose to train an n-gram model (Manning and Schütze, 1999) upon a monolingual corpus where every word has been replaced by its morphological analysis. In the case of the Apertium platform used in our experiments, the monolingual dictionary and the part-of-speech tagger are used to convert each inflected word form in the monolingual corpus (for instance, *abismos*), into a *lexical form* consisting of lemma, lexical category, and inflection information (*abismo, noun, masculine, plural*); lemmas will be discarded for the purpose of our work.

The n-gram model is then used as showed in algorithm 1: for one[8] of the paradigms p_i in the paradigm class the list of all possible inflected word forms $\{w_j\}$ for the new word is obtained (function *InflectedWordForms*). Each of the word forms w_j is then sought in a monolingual corpus (function *FindSentencesContaining*), and every sentence containing w_j is morphologically analysed to obtain the lexical category and inflection information of all its words (function *ObtainLexicalForms*), except for w_j; for the occurrence of w_j in the sentence, all the possible analysis according to the different paradigms p_i in the paradigm

[6] See section 3.6 for details about how this set was obtained.

[7] Revision 33900 in the Apertium SVN trunk `https://apertium.svn.sourceforge.net/svnroot/apertium/trunk/apertium-es-ca`.

[8] Note that since all the paradigms in the paradigm class are equivalent in the sense that they generate exactly the same set of IWFs, any of them could be chosen here.

Table 3.1: Top 6 paradigm classes for the Spanish monolingual dictionary. Examples of inflections for the different paradigms contained in each class are given together with the total number of paradigms (# P), and the number of words (# W) in the dictionary assignable to the class. The last two columns show results described in section 3.7. Confidence intervals were estimated with 95% statistical significance with a *t-test*.

Word examples	# P	# W	Success (%)	Baseline (%)
atletismo, Suecia, adiós, afueras, ...	32	11 513	56.3 ± 1.2	40.5 ± 1.2
accionista, abeja, abismo, clarisa, abundante	5	11 507	82.9 ± 0.8	46.6 ± 1.0
abogado, cuánto, absoluto, mío, otro, todo	6	3 281	72.3 ± 1.7	78.2 ± 1.6
abdominal, abril, accesibilidad, albañil	4	2 256	87.8 ± 1.5	37.3 ± 2.2
acción, aluvión, marrón, peatón	4	2 014	96.6 ± 0.9	87.7 ± 1.6
abrumadora, señora	2	571	73.9 ± 4.0	53.1 ± 5.0

class are tested one after the other and the perplexity per word (actually, perplexity per *token*; function PPW) of the sentence (Manning and Schütze, 1999) is computed according to the n-gram model of lexical inflection information. The paradigm containing the inflection information which makes the sentence obtain the smallest total perplexity per token is considered the winner. The process is repeated for every sentence in the corpus containing one of the forms w_j and the paradigm which is found winner more frequently is selected by the algorithm as the correct one. Note that an ordered list of all the paradigms in the paradigm class could also be obtained by following this procedure. A sorted list could be useful in scenarios where a user is requested to validate the associated paradigm before finally adding the new entry to the dictionary; in this case, if the first candidate is not valid, then the user will move to the second one and so on; ideally, very few paradigms would need to be tested before getting to the correct one.

3.6 Experimental settings

Since the addition by non-expert users of new entries to monolingual dictionaries has already been evaluated (Esplà-Gomis et al., 2011), our experimental set-up is focused on studying the impact of our lexical model in the selection of the correct paradigm when more than one stem/paradigm candidate exist after querying the user. The evaluation can be carried out automatically by focusing on the entries already included in the dictionary which would obtain more than one CP with the original method described in section 3.4. Since those entries already have the correct paradigm assigned, it is not necessary to pose the yes/no

Algorithm 1 Steps carried out to choose the paradigm whose part of speech and inflection information best fit the new word nw. The function *Init* initialises to zero the amount of sentences in which each paradigm from *paradigm_ class* is the best one. Note that in function *ObtainLexicalForms* the occurrence of w_j is initially marked as an unknown word, since it does not appear in the dictionary.

```
function BestParadigm( nw, list paradigm_class, corpus,ngram_model)
    list iwfs ← InflectedWordForms(nw, paradigm_class)
    map winner_paradigms ← ∅
    Init(winner_paradigms, paradigm_class)
    for all w_j ∈ iwfs do
        list occurrences ← FindSentencesContaining(w_j, corpus)
        for all occurence ∈ occurrences do
            lex_occurrence ← ObtainLexicalForms(occurrence)
            perplexity_per_word ← ∞
            best_paradigm ← null
            for all p_i ∈ paradigm_class do
                lexical_word_form ← LexForm(p_i, w_j)
                lex_occurence_replaced ← lex_occurence.Replace(w_j, lexical_word_form)
                sample_perplexity ← PPW(lex_occurence_replaced, ngram_model)
                if sample_perplexity <= perplexity_per_word then
                    perplexity_per_word ← example_perplexity
                    best_paradigm ← p_i
                end if
            end for
            winner_paradigms[best_paradigm] = winner_paradigms[best_paradigm] + 1
        end for
    end for
    return arg max_p winning_paradigms[p]
end function
```

questions to users in order to have them labelled.

We have used the Apertium Spanish–Catalan[9] language pair, and a combination of sentences from a Spanish Wikipedia dump[10] and the Spanish version of OpenSubtitles corpus (Tiedemann, 2009) as the monolingual corpus to train the n-gram model and to search for sentences containing the inflected word forms w_j in the paradigm class (see section 3.5). The n-gram model used in the experiments is a trigram model trained with the open-source toolkit IRSTLM (Federico et al., 2008) using Witten-Bell smoothing and without pruning singleton n-grams.

Our test set contains all the word entries assigned to paradigms corresponding to open part-of-speech categories which have at least two dictionary entries assigned to them. For each word in the test set, its corresponding paradigm class is obtained by checking all the possible pairs stem-paradigm generating the same IWF set than the correct stem/paradigm pair. Our new approach is then used to select one of the paradigms which is, after that, compared to the correct one according to the dictionary. As a baseline, a simple model which selects the paradigm in the class with the largest number of entries assigned to it in the monolingual dictionary is also considered. It is worth noting that the comparison is not totally fair, since the baseline uses knowledge about the number of words assigned to each paradigm in the dictionary, which is not available for our approach.

3.7 Results

Table 3.1 shows the results (two last columns) for the 6 most frequent paradigm classes among the 26 different paradigm classes which were found in the Spanish dictionary. These classes include 97.0% of the entries which can be assigned more than one candidate paradigm by the original method. Paradigm classes contain between two and six paradigms, except for one of them, which comprises 32 paradigms; this large class corresponds to paradigms containing only the suffix ϵ (which is assigned to words with one single inflected form, such as proper nouns). The results obtained by our approach clearly overcome the results obtained by the baseline, except for the third class. It is worth noting that our approach can only deal with words for which any occurrence of their inflected word forms appear in the corpus. Therefore, success rate was computed only for these words both for the baseline and for our approach.

With regard to the overall results involving all the 32 104 entries assigned to the 26 different paradigm classes and fulfilling the conditions enumerated in section 3.6, the average success rate was $75.7\% \pm 0.6$, whereas the baseline method attains $51.2\% \pm 0.7$. Confidence intervals were estimated with 95% statistical significance with a *t-test*. Figure 3.1 shows the performance of our system for all these words. It can be seen that for paradigm classes with sizes up to 6, our method selects the correct paradigm as first option for more than 70% of the words. In addition, the percentage of cases in which the correct paradigm is between the two better scored candidates is above 90%. The only exception is the case of the paradigm class containing 32 candidate paradigms; in this case, the results are not so good due to the fact that the lexical model is harder to estimate. Results for paradigm classes of size 3 are

[9] Revision 33900 in the Apertium SVN trunk `https://apertium.svn.sourceforge.net/svnroot/apertium/trunk/apertium-es-ca`.

[10] `http://dumps.wikimedia.org/eswiki/20110114/eswiki-20111208-pages-articles.xml.bz2`.

also worse than the rest, although they are not reliable, since only two words were used to obtain the results. The information represented in the histogram shows that our method is not only useful to choose the best candidate paradigm, but also to sort the paradigm candidates in scenarios as the one depicted in the end of section 3.5.

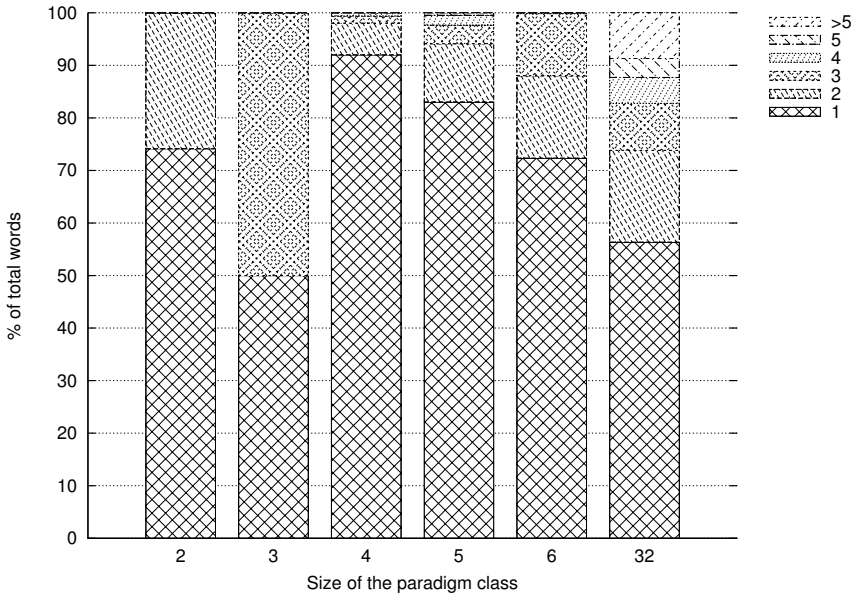


Figure 3.1: Position of the correct paradigm in the ordered list provided by our method depending on the size of the paradigm class. The number of words assignable to each class is 742, 2, 3 685, 9 289, 2 541, and 5 969, respectively. Note that only two words were found in the test set for paradigm classes with three paradigms, so results of the second bar of the histogram are not very reliable.

3.8 Concluding remarks

Our previous work on enlarging monolingual dictionaries of rule-based MT systems by non-expert users has been extended with an n-gram model of lexical category and inflection information to tackle the common case of paradigm classes including more than one paradigm. Results significantly improve those of the baseline and show that the extended system can be used to successfully obtain the right paradigm for most new words; even in those cases where the inferred paradigm is wrong, our system may prove useful as it provides an ordered list of candidates which may help users validating the new entries to quickly arrive to the correct paradigm. We plan to extend our approach to other languages and explore the use of a hidden Markov model (Manning and Schütze, 1999) instead of an n-gram language model. We also plan to detect situations in which a word may be correctly added to more than one paradigm by studying the values of the perplexities of each option.

Acknowledgements

This work has been partially funded by Spanish Ministerio de Ciencia e Innovación through project TIN2009-14009-C02-01, by Generalitat Valenciana through grant ACIF/2010/174 from VALi+d programme, and by Universitat d'Alacant through project GRE11-20.

Bibliography

Bartusková, D. and R. Sedlácek. 2002. Tools for semi-automatic assignment of Czech nouns to declination patterns. In *Proceedings of the 5th International Conference on Text, Speech and Dialogue*, pages 159–164.

Esplà-Gomis, M., V. M. Sánchez-Cartagena, and J. A. Pérez-Ortiz. 2011. Enlarging monolingual dictionaries for machine translation with active learning and non-expert users. In *Proceedings of Recent Advances in Natural Language Processing*, pages 339–346.

Federico, M., N. Bertoldi, and M. Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*.

Font-Llitjós, A. 2007. *Automatic improvement of machine translation systems*. Ph.D. thesis, Carnegie Mellon University.

Forcada, M.L., M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J.A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F.M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation* 25(2):127–144.

Hutchins, W. J. and H. L. Somers. 1992. *An introduction to machine translation*. Academic Press, London.

Manning, C. D. and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

McCreight, E.M. 1976. A space-economical suffix tree construction algorithm. *Journal of the Association for Computing Machinery* 23:262–272.

McShane, M., S. Nirenburg, J. Cowie, and R. Zacharski. 2002. Embedding knowledge elicitation and MT systems within a single architecture. *Machine Translation* 17:271–305.

Sánchez-Cartagena, V. M., M. Esplà-Gomis, and J. A. Pérez-Ortiz. 2012. Source-language dictionaries help non-expert users to enlarge target-language dictionaries for machine translation. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, eds., *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Tiedemann, J. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, vol. V, pages 237–248. John Benjamins.

Wang, A., C. D. V. Hoang, and M. Kan. 2010. Perspectives on crowdsourcing annotations for natural language processing. Tech. rep., School of Computing, National University of Singapore.

Chapter 4

An Open-Source Toolkit for Integrating Shallow-Transfer Rules into Phrase-Based Statistical Machine Translation

V. M. Sánchez-Cartagena, F. Sánchez-Martínez, J. A. Pérez-Ortiz
Universitat d'Alacant

Abstract

In this paper, we present an open-source toolkit to enrich a phrase-based statistical machine translation system (Moses) with phrase pairs generated from the linguistic resources of a shallow-transfer rule-based machine translation system (Apertium). A system built with this toolkit was not outperformed by any other participant in the shared translation task of the Sixth Workshop on Statistical Machine Translation (WMT 11) for the Spanish–English language pair.

4.1 Introduction

Statistical machine translation (SMT) (Koehn, 2010) systems are very attractive because they may be built with little human effort when enough monolingual and bilingual corpora are available. However, bilingual corpora are not always easy to harvest, and they may not even exist for some language pairs. On the contrary, rule-based machine translation systems (RBMT) (Hutchins and Somers, 1992) may be built without any parallel corpus; however, they need an explicit representation of linguistic information, whose coding by human experts requires a considerable amount of time and economic resources. When both parallel corpora and linguistic information exist, a hybrid approach may be followed in order to make the most of such resources.

In this paper, we present the free/open-source Rule2Phrase Toolkit, which implements a recently developed hybrid approach (Sánchez-Cartagena et al., 2011, Sánchez-Cartagena et al., 2011a,b) to enrich a phrase-based (Koehn et al., 2003) SMT system with resources from shallow-transfer RBMT; this toolkit is designed to work with the Apertium (Forcada et al., 2011) RBMT platform and the Moses (Koehn et al., 2007) phrase-based SMT system. The Rule2Phrase Toolkit, which is described for the first time in this paper, permits the creation of a set of phrase pairs which encode the knowledge present in the Apertium linguistic resources, and implements different strategies to integrate them in the translation models built with Moses.

Different experiments have been performed previously to validate the hybrid approach using this toolkit. Experiments carried out with small training corpora confirmed its effectiveness (Sánchez-Cartagena et al., 2011a). Experiments performed with bigger training corpora showed that Apertium data is very useful to improve the translation of general domain texts when systems are trained on in-domain corpora (Sánchez-Cartagena et al., 2011b). In addition, a system built under this hybrid philosophy (Sánchez-Cartagena et al., 2011) was not outperformed by a statistically significant margin by any other participant in the shared translation task of the Sixth Workshop on Statistical Machine Translation (WMT 11)(Callison-Burch et al., 2011) for the Spanish-English language pair.

The rest of the paper is organised as follows. Next section overviews the MT systems combined by the Rule2Phrase Toolkit, while section 4.3 presents some similar hybridisation approaches. The hybridisation strategy is described in section 4.4; then, section 4.5 describes the design principles, some implementation details and usage examples of the toolkit. The paper ends with some concluding remarks.

4.2 Translation Approaches

4.2.1 Phrase-Based Statistical Machine Translation

The Moses toolkit, as well as other phrase-based statistical machine translation systems (PBSMT) (Koehn, 2010, ch. 5), translates sentences by maximising the translation probability as defined by the log-linear combination of a number of feature functions, whose weights are chosen to optimise translation quality (Och, 2003). A core component of every PBSMT system is the phrase table, which contains bilingual phrase pairs extracted from a bilingual corpus after word alignment (Och and Ney, 2003). The set of translations from which the most probable one is chosen is built by segmenting the source-language (SL) sentence in all possible ways and then combining (possibly with some reordering) the translation of the different source segments according to the phrase table.

4.2.2 Shallow-transfer rule-based machine translation

The RBMT process can be split into three steps: i) analysis of the SL text to build a SL intermediate representation, ii) transfer from that SL intermediate representation to a target-language (TL) representation, and iii) generation of the final translation from the TL intermediate representation.

Shallow-transfer RBMT systems use relatively simple intermediate representations based

on lexical forms consisting of lemma, part of speech and morphological inflection information of the words in the input sentence, and apply simple shallow-transfer rules that operate on sequences of lexical forms (no full parsing is performed). Apertium, the shallow-transfer RBMT platform our toolkit is designed to work with, splits the transfer step into structural and lexical transfer. The lexical transfer is performed by using a bilingual dictionary which, for each SL lexical form, always provides the same TL lexical form (no lexical selection is performed). Note that multi-word expressions (such as *on the other hand*, which acts as a single adverb) may be analysed to (or generated from) a single lexical form.

Structural transfer is performed by applying a set of rules in a left-to-right, longest-match fashion; rules are applied to sequences of words and prevent word-for-word translation in those cases in which this would result in an incorrect translation. The structural transfer may be split into three levels in order to facilitate the writing of rules, although, for the sake of simplicity, in this paper only a single-level transfer is taken into account.[1]

The SL intermediate representation is obtained by analysing the SL text with a SL monolingual dictionary and a part-of-speech tagger. A *pretransfer* module then splits those lexical forms, such as verbs with enclitic pronouns and contractions, that will be processed as separate units by the transfer module. The final translation is generated from the TL intermediate representation with a TL monolingual dictionary.

Suppose that the Catalan sentence *La deterioració del senyal* (*the deterioration of the signal* in English) is to be translated into Spanish by Apertium. First, it is analysed as:

```
el<det><def><f><sg> deterioració<n><m><sg>
de<pr>+el<det><def><m><sg>  senyal<n><m><sg>
```

which splits the sentence into four lexical forms: a feminine plural definite determiner (*la*), a noun in feminine singular (*deterioració*), the preposition *de*, joint with a masculine plural definite determiner (*el*), and a noun in masculine singular (*senyal*).
The *pretransfer* module then splits the joint lexical form:

```
el<det><def><f><sg> deterioració<n><f><sg> de<pr>
el<det><def><m><sg> senyal<n><m><sg>
```

After that, the transfer is executed. It performs the lexical transfer and applies the first-level rules of the structural transfer in parallel. On the one hand, the lexical transfer gives as a result:

```
el<det><def><f><sg> deterioro<n><m><sg> de<pr>
el<det><def><m><sg> señal<n><f><sg>
```

On the other hand, a first-level Apertium structural transfer rule is triggered, twice in this case. This rule matches a determiner followed by a noun and makes the determiner to agree in gender and number with the noun. As a result, the final TL lexical forms are obtained:

```
el<det><def><m><sg> deterioro<n><m><sg> de<pr>
el<det><def><f><sg> señal<n><f><sg>
```

Finally, the translation into TL is generated from the TL lexical forms: *El deterioro de la señal*.

[1] Although it facilitates the writing of long rules by linguists, Apertium multi-level transfer has the same expressive power than single-level transfer. However, it is important to remark that the Rule2Phrase Toolkit is also able to work with multi-level transfer rules.

4.3 Related work

Although we are not aware of any other approach which explicitly reuses both structural transfer rules and bilingual dictionaries of a RBMT system in order to improve a SMT one, as does the Rule2Phrase Toolkit, some similar approaches exist.

Bilingual dictionaries are the most reused resource from RBMT. They have been added to SMT systems since its early days (Brown et al., 1993). One of the simplest strategies, which has already been put into practice with the Apertium bilingual dictionaries (Tyers, 2009), consists of adding the dictionary entries directly to the parallel corpus. In addition to the obvious increase in lexical coverage, Schwenk et al. (2009) state that the quality of the alignments obtained is also improved when the words in the bilingual dictionary appear in other sentences of the parallel corpus. However, it is not guaranteed that, following this strategy, multi-word expressions from the bilingual dictionary that appear in the SL sentences are translated as such because they may be split into smaller units by the phrase-extraction algorithm. Other approaches go beyond simply adding a dictionary to the parallel corpus. For instance, Popovic and Ney (2006) propose combining that strategy with the use of hand-crafted rules to reorder the SL sentences to match the structure of the TL.

Although RBMT transfer rules have also been reused in hybrid systems, they have been mostly used implicitly as part of a complete RBMT engine. For instance, Dugast et al. (2008) show how a PBSMT system can be bootstrapped using only monolingual data and an RBMT engine; RBMT and PBSMT systems can also be combined in a serial fashion (Dugast et al., 2007). Another remarkable study (Eisele et al., 2008) presents a strategy based on the augmentation of the phrase table to include information provided by an RBMT system. In this approach, the sentences to be translated by the hybrid system are first translated with an RBMT system and a small phrase table is obtained from the resulting parallel corpus. Phrase pairs are extracted following the usual procedure (Koehn, 2010, sec. 5.2.3) which generates the set of all possible phrase pairs that are consistent with the word alignments. In order to obtain reliable word alignments, they are computed using an alignment model previously built from a large parallel corpus. Finally, the RBMT-generated phrase table is directly added to the original one. On the contrary, our approach directly generates phrase pairs which match either an entry in the bilingual dictionary or a structural transfer rule; thus preventing them from being split into smaller phrase pairs even if they would be consistent with the word alignments. In addition, our approach does not require a large parallel corpus from which to learn an alignment model.

All the approaches described above share a feature: the main system is a statistical one and it is enriched (or even built) with resources from RBMT. However, there are other ways of combining RBMT and SMT. For instance, in statistical post-edition (Simard et al., 2007) the output of an RBMT system is coupled to a SMT decoder which tries to correct the errors made by the RBMT engine. A SMT system may also be enriched with other resources, such as phrases from a example-based machine translation system (Dandapat et al., 2010).

4.4 Conceptual Background

As already mentioned, the Apertium structural transfer module detects sequences of lexical forms which need to be processed together to prevent wrong word-for-word translations.

Therefore, adding to the phrase table of a PBSMT system all the bilingual phrase pairs which either match one of these sequences of lexical forms in the structural transfer or an entry in the bilingual dictionary suffices to incorporate all the linguistic information provided by Apertium. In this section, the generation of these phrase pairs and three different methods to score them are presented; additional implementation details for the Rule2Phrase Toolkit can be found in section 4.5.

4.4.1 Phrase Pair Generation

As described in section 4.5.2, generating bilingual phrase pairs from the bilingual dictionary involves a straightforward combination of the data in the bilingual and monolingual dictionaries.

When generating phrase pairs from the structural transfer rules, the amount of generated pairs is an important issue. Consider, for instance, the rule which is triggered by a determiner followed by a noun and an adjective. Generating all the possible phrase pairs matching this rule would involve combining all the determiners in the dictionary with all the nouns and all the adjectives, producing many meaningless phrases, such as the Spanish *el niño inalámbrico* (*the wireless boy* in English) and making the approach computationally infeasible due to the large number of resulting combinations. In the experiments carried out to evaluate the hybrid approach, this issue was solved by generating only phrase pairs whose source side occurs in the test and tuning sets.

Let the Catalan sentence *El senyal roig*, similar to the example in section 4.2.2, be one of the sentences to be translated. If, in addition to the rule fired by a determiner plus a noun presented in the previous example, there is another rule which matches a determiner followed by a noun and an adjective, the SL sequences *El senyal*, and *El senyal roig* are used to generate bilingual phrase pairs because both match a first-level rule; note that the SL word sequence *El senyal* is used twice because it is covered by two first-level rules.

4.4.2 Scoring the New Phrase Pairs

The Moses PBSMT system attaches 5 scores to every phrase pair in the translation table: source-to-target and target-to-source phrase translation probabilities and lexical weightings, and phrase penalty. The phrase translation probabilities and lexical weightings of the phrase pairs generated from Apertium may be calculated in three different ways which we describe next (computation of the phrase penalty is trivial). As in previous experiments (Sánchez-Cartagena et al., 2011b) neither of the three strategies clearly outperformed the others, the three approaches are implemented by the toolkit.

Augmenting the Training Corpus (*corpus-rules*). The simplest approach involves appending the Apertium-generated phrase pairs to the training corpus and running the usual PBSMT training algorithm. This improves the alignments obtained from the original training corpus and enriches both the phrase table and the reordering model. However, the phrase extraction algorithm (Koehn, 2010, sec. 5.2.3) may split the resulting bilingual phrase pairs into smaller units which may cause multi-word expressions not to be translated in the same way as they appear in the Apertium bilingual dictionary.

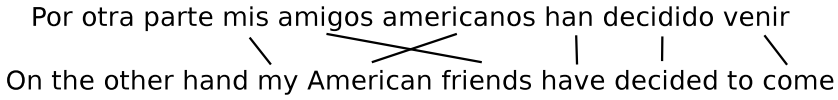


Figure 4.1: Example of word alignment obtained by tracing back the operations performed by Apertium when translating from Spanish to English the sentence *Por otra parte mis amigos americanos han decidido venir*. Note that *por otra parte* is analysed by Apertium as a multi-word expression whose words are left unaligned for convenience (see section 4.4.2).

Expanding the Phrase Table (*phrase-rules*). Apertium-generated phrase pairs are added to the list of corpus-extracted phrase pairs; then, the phrase translation probabilities are calculated by relative frequency as it is usually done (Koehn, 2010, sec. 5.2.5). As they are added only once, if one of them happens to share its source side with many other corpus-extracted phrase pairs, or even with a very frequent, single one, the RBMT-generated phrase pair will receive lower scores, which penalises its use. To alleviate this without adding the same phrase pair an arbitrary amount of times, an additional boolean score to flag Apertium-generated phrase pairs can be introduced. [2]

To calculate the lexical weightings (Koehn, 2010, sec. 5.3.3) of the RBMT-generated phrase pairs, a probabilistic bilingual dictionary and the alignments between the words in the source side and those in the target side are needed. These word alignments are obtained by tracing back the operations carried out in the different steps of Apertium (see section 4.5.2). Only those words which are neither split nor joined with other words by the RBMT engine are included in the alignments; thus, multi-word expressions are left unaligned. This is done for convenience, so that multi-word expressions are assigned a lexical weighting of 1.0. Figure 4.1 shows the alignment between the words of a sentence in Spanish and its English translation with Apertium which would be obtained with this strategy. Regarding the probabilistic bilingual dictionary, it is usually computed from the word alignments extracted from the training corpus. Our approach also takes advantage from the Apertium bilingual dictionary to obtain a richer probabilistic bilingual dictionary.

Combining Both Approaches (*pc-rules*). The two previous approaches may be combined by appending the RBMT bilingual phrase pairs to both the training corpus and the phrase table. Following this strategy, the list of phrase pairs from which the phrase table is built will contain each Apertium-generated pair twice, but each sub-phrase identified by the phrase-extraction algorithm only once. Therefore, phrase pairs extracted from Apertium which have been split will be present in the phrase table, but they will have lower scores than those which have not been split. In addition, as in the *corpus-rules* approach, the alignment model is built from a bigger corpus, and so is the reordering model.

[2] Phrase pairs generated from Apertium which are also extracted from the corpus are flagged as Apertium-generated too.

4.5 Description of the Toolkit

4.5.1 Overview

The Rule2Phrase Toolkit implements the hybridisation strategy described above by easily allowing its users to perform these two main steps:

1. Generate a list of phrase pairs and the alignments between their words from the Apertium linguistic resources (see section 4.4.1).

2. Integrate the resulting Apertium-generated phrases in a PBSMT system built with Moses following the three strategies previously presented (see section 4.4.2).

As trying to generate all the SL phrases which match a transfer rule would result in an excessive amount of meaningless phrases (see section 4.4.1), a mechanism to filter them and generate only sentences which are likely to appear in the texts the hybrid system will have to translate is needed. In the experiments performed to validate this hybrid approach (Sánchez-Cartagena et al., 2011, Sánchez-Cartagena et al., 2011a,b) this objective was accomplished by simply generating only phrases present in the tuning and test corpora. However, a different solution is needed when the resulting hybrid system is to be used in a real scenario, where the texts to be translated are not known a priori.

Our toolkit is able to deal with both scenarios by performing a n-gram based filtering. A list of allowed n-grams must be provided when generating the phrase pairs from the transfer rules, so that those SL sequences containing exclusively n-grams from the allowed list are generated. Therefore, if the test corpus is provided, the list of allowed n-grams is extracted from it. If not, one can simply use the most probable n-grams of a source-language model. This second strategy is partially implemented, and still requires a systematic evaluation.

Regarding the integration of the Apertium-generated corpus in the models of PBSMT system, our toolkit provides a wrapper over the Moses training scripts, which facilitates the integration the Apertium-generated corpus in the PBSMT models following any of the three strategies defined in section 4.4.2.

4.5.2 Design Principles

The design of the two modules of our toolkit and their interaction with Apertium and Moses are discussed in this section.

Phrase Generation Module

The generation of the Apertium-generated phrases from the dictionaries is straightforward. All the SL surface forms recognised by Apertium and their corresponding lexical forms are obtained from the SL monolingual dictionary; then, these SL lexical forms are translated using the bilingual dictionary; finally, their TL surface forms are generated using the TL monolingual dictionary.

The generation of phrase pairs from the Apertium shallow-transfer rules, which is summarised in figure 4.2, is performed as follows. Firstly, all the SL lexical form sequences (extracted from the SL monolingual dictionary) which match a first-level transfer rule and

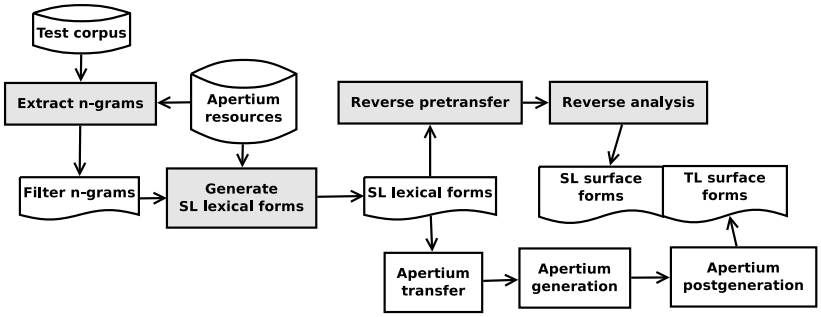


Figure 4.2: Steps carried out by the Rule2Phrase Toolkit to generate a set of phrase pairs from the Apertium transfer rules (grey-shadowed boxes).

whose subsequences are present in the list of allowed n-grams are generated. Note that transfer rules are applied to the output of the *pretransfer* module, which means that, at this step, lexical forms which would have been split by the *pretransfer* module (such as contractions and verbs plus enclitic pronouns) must appear as independent lexical forms. Therefore, when extracting the n-grams from the test corpus (see section 4.5.1), it must be analysed and passed through the *pretransfer* module first.

Then, for each SL lexical form sequence, two processes are carried out in order to obtain, respectively, the TL and the SL side of the resulting bilingual phrase pair.

In the first process, each SL lexical form sequence is passed through the rest of the Apertium pipeline to obtain a TL surface form sequence. The alignments between the input and output sequences of lexical forms of each module are also computed. For instance, consider the following SL lexical form sequence obtained when generating phrase pairs from Apertium for translating from Catalan to Spanish and that a rule matching the preposition *de* plus a determiner followed by a noun is applied:

```
de<pr> el<det><def><m><sg> senyal<n><m><sg>
```

The transfer module produces the following TL lexical forms:

```
de<pr> el<det><def><f><sg> señal<n><f><sg> (1-1 2-2 3-3)
```

Alignments are represented as pairs of numbers $i - j$, where i is the position of the SL word aligned with the TL word at position j. Finally, the generation module produces the following TL surface forms:

```
de la señal (1-1 2-2 3-3)
```

In the second process, the SL surface forms are obtained by firstly passing each SL lexical form sequence through a new module which joins the words split by the pretransfer:

```
de<pr>+el<det><def><m><sg> senyal<n><m><sg> (1-1 1-2 2-3)
```

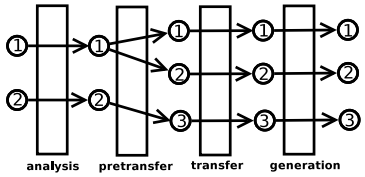


Figure 4.3: Alignments obtained from the different Apertium modules when translating the sentence *Del senyal* from Catalan to Spanish.

And then by using the SL monolingual dictionary to obtain the corresponding surface forms:

```
del senyal (1-1 2-2)
```

Finally, the end-to-end alignments between SL and TL surface forms are obtained by joining the alignments of each module. Alignments are combined in a transitive manner: if module A output is connected to module B input, word i in the input of module A is aligned with word j in its output, and word j in the input of B aligned with word k in its output, we can state that word i is aligned with word k. Figure 4.3 shows how the final alignments of the running example (*1-1 1-2 2-3*) are calculated. At this point the toolkit permits keeping only the alignments which meet the restrictions defined in section 4.4.2.

Integration Module

Implementing the strategy *corpus-rules*, defined in section 4.4.2, only requires concatenating the Apertium-generated phrases with the original training corpus and running the Moses training script as-is. However, the other two strategies involve adding additional steps to the standard Moses training process.

In particular, the following steps, summarised in figure 4.4, are automatically executed by the Rule2Phrase Toolkit to integrate the Apertium-generated phrases using the strategy *phrase-rules*:

1. Alignments of the original training corpus are obtained using the Moses toolkit.
2. The probabilistic bilingual dictionary is obtained from the concatenation of the original training corpus and the subset of the Apertium-generated phrases obtained from the dictionaries.
3. Phrase pairs are extracted from the original training corpus.
4. Apertium-generated phrase pairs are appended to the list of corpus-extracted phrase pairs.
5. Phrase pairs are scored to obtain the phrase table.
6. The boolean score which flags Apertium-generated phrase pairs (see section 4.4.2) is added to the phrase table.

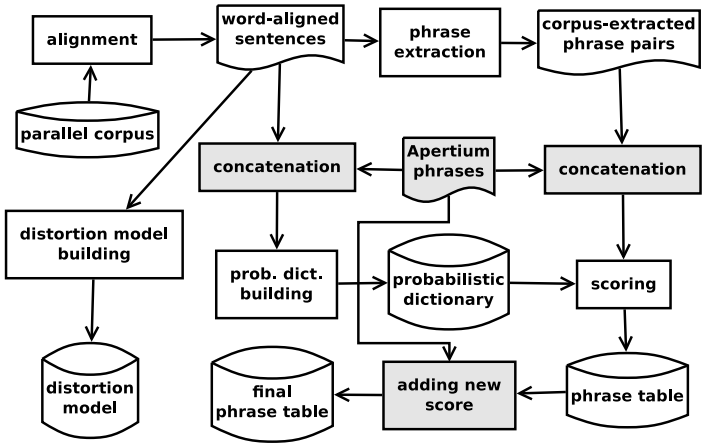


Figure 4.4: Steps carried out by the Rule2Phrase Toolkit to integrate a set of phrase pairs extracted from Apertium directly into the phrase table of a Moses system (grey-shadowed boxes).

7. The remaining standard Moses training pipeline is executed.

The *pc-rules* strategy involves a few less steps:

1. Phrase pairs are extracted from the concatenation of the original training corpus and the Apertium-generated phrases.
2. Apertium-generated phrase pairs are appended to the list of corpus-extracted phrase pairs.
3. Phrase pairs are scored to obtain the phrase table.
4. The boolean score which flags Apertium-generated phrase pairs (see section 4.4.2) is added to the phrase table.
5. The remaining standard Moses training pipeline is executed.

4.5.3 Implementation Details

The different steps of the actions carried out by the toolkit are encoded in a GNU Make Makefile in order to avoid executing some of them when it is not necessary. It is wrapped by a Python script which simplifies the parameter processing, and the main modules are written in Java and Python. Some UNIX utilities such as *sort* and *uniq* are used too.

The strategy to obtain the alignments varies across the different Apertium modules. Obtaining them from the analysis, generation and *pretransfer* modules is relatively straightforward as they keep word order and only split or join words in some cases. Therefore, maintaining a list of multi-word units suffices to obtain the alignments of the words processed by them.

On the contrary, obtaining the word alignments associated with the operations carried out by the transfer module is a more complex task, since transfer may add, delete and reorder words. In order to keep track of these operations, it has been modified to append to its output some extra information, including which transfer rules have been applied and, for each rule, which input SL word corresponds to each output TL word. When generating the phrase pairs from the Apertium linguistic resources, the toolkit executes this modified transfer module and parses its output to obtain the alignments.

4.5.4 Using the Toolkit

The Rule2Phrase Toolkit is licensed under the GNU GPL v3 free software license.[3] It has been tested under GNU/Linux, although it should work under other operating systems, as long as GNU Make and some other UNIX utilities are available for them.

Assuming that Apertium and Moses are already present in the system, installing our toolkit only involves unpacking the binary distribution and patching and recompiling Apertium for obtaining alignment information, a task for which convenient scripts are provided.

Once Apertium has been patched, generating the phrase pairs from it is as easy as typing:

```
$ python rule2Phrase.py --extract-n-grams --test TEST\_CORPUS
                        --output NGRAMS\_DIRECTORY --sl SL --tl TL
```

to extract the n-grams from the test corpus and:

```
$ python rule2Phrase.py --gen-phrases --n-grams NGRAMS\_DIRECTORY
                       --output NEWPHRASES\_DIRECTORY --sl SL --tl TL
```

to get the actual Apertium-generated phrases and their alignments. It is assumed that Apertium is installed under the standard prefix (`/usr/local`), but different installation directories may be chosen.[4]

Regarding the integration of the Apertium-generated phrases in the Moses PBSMT system, the toolkit provides a command for each of the steps described in section 4.5.2 which are not part of the standard Moses training procedure.[5] In addition, the enriched PBSMT system may be built with a single command. For instance, for the *pc-rules* strategy:

```
$ python rule2Phrase.py --buildSMT phrase-rules --synth-phrases
                                  NEWPHRASES\_DIRECTORY --sl SL --tl TL
```

4.6 Concluding Remarks

In this paper, we have presented an open-source toolkit which permits the enrichment of the PBSMT system Moses with phrase pairs generated from the linguistic resources of the shallow-transfer RBMT system Apertium. The hybridisation strategy implemented by the toolkit has already been evaluated with different experiments, which showed that it is very effective when the training corpus is small (Sánchez-Cartagena et al., 2011a)

[3] The toolkit can be downloaded from `http://www.dlsi.ua.es/~vmsanchez/Rule2Phrase.tar.gz`

[4] Run `python rule2Phrase.py -help` for a list of available options.

[5] The integration of Apertium phrase pairs into Moses has been tested with Moses revision 3739.

or the systems are trained on in-domain corpora (Sánchez-Cartagena et al., 2011b) and the texts to be translated are from a general (news) domain. A system built under this hybrid philosophy (Sánchez-Cartagena et al., 2011) was not outperformed by a statistically significant margin by any other participant of the shared translation task from the Sixth Workshop on Statistical Machine Translation (WMT 11)(Callison-Burch et al., 2011) for the Spanish–English language pair. We release the toolkit with the hope that it will be useful to other MT practitioners.

Acknowledgements

This work has been partially funded by Spanish Ministerio de Ciencia e Innovación through project TIN2009-14009-C02-01, by Generalitat Valenciana through grant ACIF/2010/174 from VALi+d programme, and by Universitat d'Alacant through project GRE11-20.

Bibliography

Brown, P. F., S. A. D. Pietra, V. J. D. Pietra, M. J. Goldsmith, J. Hajic, R. L. Mercer, and S. Mohanty. 1993. But dictionaries are data too. In *Proceedings of the workshop on Human Language Technology*, pages 202–205. ISBN 1-55860-324-7.

Callison-Burch, C., P. Koehn, C. Monz, and O. Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64.

Dandapat, S., M. L. Forcada, D. Groves, S. Penkale, and A. Way. 2010. *OpenMaTrEx: A Free/Open-Source Marker-Driven Example-Based Machine Translation System*, pages 121–126. Berlin: Heidelberg: Springer. ISBN 978-3-642-14769-2.

Dugast, L., J. Senellart, and P. Koehn. 2007. Statistical post-editing on SYSTRAN's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223.

Dugast, L., J. Senellart, and P. Koehn. 2008. Can we Relearn an RBMT System? In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 175–178.

Eisele, A., C. Federmann, H. Saint-Amand, M. Jellinghaus, T. Herrmann, and Y. Chen. 2008. Using Moses to integrate multiple rule-based machine translation engines into a hybrid system. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 179–182.

Forcada, M.L., M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J.A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F.M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation* 25(2):127–144. Special Issue: Free/Open-Source Machine Translation.

Hutchins, W. J. and H. L. Somers. 1992. *An introduction to machine translation*, vol. 362. Academic Press New York.

Koehn, P. 2010. *Statistical Machine Translation*. Cambridge University Press.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, C. Shen, W.and Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Koehn, P., F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 48–54. Edmonton, Canada.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.

Och, F. J. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29:19–51.

Popovic, M. and H. Ney. 2006. Statistical machine translation with a small amount of bilingual training data. In *LREC workshop on Minority Languages*, pages 25–29.

Sánchez-Cartagena, V. M., F. Sánchez-Martínez, and J. A. Pérez-Ortiz. 2011a. Enriching a statistical machine translation system trained on small parallel corpora with rule-based bilingual phrases. In *Proceedings of Recent Advances in Natural Language Processing*, pages 90–96. Hissar, Bulgaria.

Sánchez-Cartagena, V. M., F. Sánchez-Martínez, and J. A. Pérez-Ortiz. 2011b. Integrating shallow-transfer rules into phrase-based statistical machine translation. In *Proceedings of the XIII Machine Translation Summit*, pages 562–569. Xiamen, China.

Sánchez-Cartagena, V. M., F. Sánchez-Martínez, and J. A. Pérez-Ortiz. 2011. The universitat d'alacant hybrid machine translation system for wmt 2011. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 457–463. Edinburgh, Scotland: Association for Computational Linguistics.

Schwenk, H., S. Abdul-Rauf, L. Barrault, and J. Senellart. 2009. SMT and SPE machine translation systems for WMT'09. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 130–134.

Simard, M., N. Ueffing, P. Isabelle, and R. Kuhn. 2007. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206. Prague, Czech Republic.

Tyers, F. M. 2009. Rule-based augmentation of training data in Breton-French statistical machine translation. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 213–217.

Chapter 5

A rule-based machine translation system from Serbo-Croatian to Macedonian

Hrvoje Peradin, Francis Tyers
University of Zagreb, Universitat d'Alacant

Abstract

This paper describes the development of a one-way machine translation system from Serbo-Croatian to Macedonian on the Apertium platform. Details of resources and development methods are given, as well as an evaluation, and general directives for future work.

5.1 Introduction

The modern Macedonian language was standardised in 1944. and is the official language of the Republic of Macedonia.

Serbo-Croatian is a term that encompasses four standard languages (Bosnian, Croatian, Montenegrin and Serbian) based on the *neoshtokavian* dialect. The standardisation of the language started in the 19th century, as an attempt to unify the literary and linguistic traditions of the south Slavic area. The standard remained pluricentric until the dissolution of Yugoslavia. Due to the large similarities between the standards we have decided to group them into one module, with a common mode for analysis. Having in mind future work we have added separate modes for generation of Bosnian, Croatian and Serbian.[1]

Serbo-Croatian and Macedonian are largely mutually intelligible, however despite their close relation the differences in morphology create difficulties in translation. For this reason the system is currently mono-directional (sh→mk). The direction was chosen since it is

[1]The standardisation process of Montenegrin is under way, and we are awaiting the outcome to implement a separate mode.

easier to construct a system translating from a more detailed morphology (e.g. Macedonian "во куќа" can be translated both as "u kući" [in the house.LOC] and "u kuću"[into the house.ACC]).

Other systems currently supporting the languages are notably Google Translate[2] and Systran.[3]

The language pair `apertium-sh-mk`[4] is available under GNU GPL.

5.2 Design

The Apertium platform

The Apertium[5] platform follows a modular machine translation model. Morphological analysis of the source text is performed by a letter transducer compiled from a morphological lexicon,[6] and cohorts[7] obtained in this manner go through a disambiguation process. Disambiguated readings proceed to a bilingual dictionary also performed by a letter transducer and through a two-level syntactic transfer, which performs word reordering, deletions, insertions, and basic syntactic chunking. The final module is a letter transducer that generates surface forms in the target language from the bilingual transfer output.

Constraint Grammar

The disambiguation in this language pair is performed by a Constraint Grammar (CG) module[8]. CG is a paradigm that uses hand-written rules to reduce the problem of linguistic ambiguity. A series of context-dependent rules are applied to a stream of tokens and readings for a given surface form are excluded, selected or assigned additional tags.

5.3 Development

Resources

Although some resources for morphological analysis of Serbian and Croatian exist (Vitas and Krstev, 2004, Vitas et al., 2003, Agić et al., 2008, Šnajder et al., 2008), to our knowledge there are none freely available for either Serbian, Bosnian or Croatian. Thus the monolingual dictionary for Serbo-Croatian has been developed almost from scratch, with the aid of a Croatian grammar (Barić et al., 1997), and on-line resources such as *Hrvatski jezični portal*,[9]

[2] Supports Croatian, Serbian and Macedonian.

[3] Language pairs Serbian→English, Croatian→English.

[4] `http://wiki.apertium.org/wiki/Serbo-Croatian_and_Macedonian`

[5] `http://wiki.apertium.org/`

[6] A morphological lexicon contains ordered pairs of word surface forms and their lemmatised analyses.

[7] A cohort consists of a surface form and one or more readings containing the lemma of the word and the morphological analysis.

[8] Implemented in the CG3 formalism, using the `vislcg3` compiler, available under GNU GPL. For a detailed reference see: `http://beta.visl.sdu.dk/cg3.html`

[9] `http://hjp.srce.hr`

wiktionaries and Wikipedia, as well as an SETimes corpus[10] (Tyers and Alperen, 2010) and a corpus composed from the Serbian, Bosnian, Croatian and Serbo-Croatian Wikipedias.

Bilingual resources available were also scarce. We used a parallel corpus obtained from SETimes, and a Serbian–Macedonian dictionary.[11]

The morphological analyser/generator for Macedonian was taken from `apertium-mk-bg` (Rangelov, 2011), which is freely available under GNU GPL. For reference on the Macedonian language we used the SEELRC reference grammar[12] and Дигитален речник на македонскиот јазик.[13]

Analysis and generation

The morphological analyser for Serbo-Croatian was written in the XML formalism of *lttoolbox*[14] (Ortiz-Rojas et al., 2005), almost entirely from scratch, with the aim to match the lexicon of the analyser from `apertium-mk-bg`. Since we intended to create a resource for all three standards, a paradigm was assigned to the reflex of the vowel yat[15] to enable analysis of both ekavian and ijekavian dialects (for a more detailed reference on Serbo-Croatian dialects see Brown and Alt, 2004), and the extended metadix format was used to enable separating different standards by analysis and generation modes.

The basic inflectional paradigms were taken from the Croatian grammar (Barić et al., 1997), and further refined according to new entries (e.g. with voice changes not covered by basic declension patterns).

The entries were made mostly manually, with some proper nouns obtained semi-automatically from the Macedonian dictionary.

Disambiguation

As there was no reliable, free training corpus, and the target-language based training of Sánchez-Martínez et al. (2008) only supports 1-stage transfer, we elected to do the disambiguation solely by a Constraint Grammar module, and omit the statistical tagger component standardly used in Apertium language pairs. In case of remaining ambiguity, the system picks the first analysis from the output of the disambiguation module.

The following are examples of disambiguation rules:

- Preposition-based case disambiguation:

  (9) ...u mojoj kući...
  [in.PR.GEN/ACC/LOC] [my.PRN.DAT/LOC] [house.DAT/LOC]
  (in my house)

[10] `http://opus.lingfil.uu.se/SETIMES.php`
[11] `http://rechnik.on.net.mk/`
[12] `http://slaviccenters.duke.edu/projects/grammars`
[13] A digital dictionary of the Macedonian language, `http://www.makedonski.info/`
[14] `http://wiki.apertium.org/wiki/Lttoolbox`
[15] Typically in ekavian it is either a long or short "e", while in ijekavian the long variant is reflected as "ije", and the short as "je".

```
REMOVE Prep + $$Case IF (1 Nominal - $$Case)
```

```
REMOVE Nominal + $$Case IF (NOT -1 Prep + $$Case) (NOT -1 Modifier + $$Case)
```

The first rule cleans a grammatical case from a preposition[16] if it is not followed by a noun, pronoun or adjective in the same case. The second rule, similarly, cleans a case from a noun, pronoun or adjective if it is not preceded directly either by a preposition which governs the case or a modifier (e.g. adjective or demonstrative pronoun) in the same case.[17] Thus the whole phrase is correctly disambiguated as locative.

- Noun phrases:

(10) ...lijepa žena...
[pretty.ADJ.(NT.PL)/(F.SG)] [woman.N.(F.SG)/(F.PL)]
(a pretty woman)

```
REMOVE Modifier + $$GenNum IF (1 Nominal - $$GenNum)
```

```
REMOVE Nominal + $$GenNum IF (-1 Modifier - $$GenNum)
```

These rules operate on noun phrases, and use the gender and number agreement to eliminate grammatically impossible readings. In this example the first rule removes the neuter reading from the adjective, since the noun it agrees with does not have the neuter gender. The adjective is then left only with the singular reading and the second rule proceeds to remove the plural reading from the noun.

- Adverb / adjective ambiguity:

(11) On puno radi.
[he] [full.(ADJ.NT.SG)/(ADV)] [works.VB]
(He works a lot.)

```
SELECT Adverb IF (0 Adverb OR Adjective) (1 Verb)
```

This simple rule resolves a common ambiguity by selecting the adverb reading if the word is followed by a verb.

- Dative / locative ambiguity:

(12) Brod prilazi luci.
[ship] [approaches] [harbour.DAT/LOC]
(The ship is approaching the harbour.)

```
SELECT Dative IF (0 Dative OR Locative) (NOT -1 Prep) (NOT -1 Modifier +
Locative)
```

The cases are orthographically identical, however locative is purely prepositional, so in most cases the ambiguity is easily resolved by selecting dative if the phrase is not preceded by a locative preposition.

[16] The cases the prepositions govern are marked on the analyses of the prepositions.
[17] The $$ prefix signifies unification, i.e. iteration over the set of all grammatical cases.

Lexical transfer

The bilingual lexicon was written using the *lttoolbox* format, and composed mostly manually, with paradigms added to compensate the tag set differences. Translation entries were added according to the lexicon from the Macedonian analyser. Having in mind future work, translations specific solely to Bosnian, Croatian or Serbian standard were grouped in respective sections.

Syntactic transfer

Despite the close relation of the two languages, there are substantial differences in morphology, and structures with analogous functionality are not necessarily morphologically cognate. Therefore we have used a two level syntactic transfer.

The first level performs tag mappings, normalisation (e.g. case to nominative, infinitive to present), rudimentary transformations, and packing of phrases in syntactic chunks.

Examples of transfer rules:

- The future tense:

  (13) Ja ću gledati[18] → Јаз ќе гледам
  [I] [will.CLT.P1.SG] [watch.INF] → [I] [will.CLT] [watch.PRES.P1.SG]
  (I will watch.)

  Serbo-Croatian uses a clitic + infinitive form with a declinable clitic, while Macedonian uses a frozen clitic form, and the person/number is marked on the verb. Thus several rules were written to match occurrences of future tense and transfer the information in translation.

- Clitic reordering:

  (14) Okrenut ću se → Ќе се обрнам
  [turn.INF] [will.CLT.P1.SG] [myself.CLT] →
  [will.CLT] [myself.CLT] [turn.PRES.P1.SG]
  (I will turn myself around)

  The order of clitics in both languages is different, so a series of rules was written to rearrange them.

- Cases as prepositional phrases:

  (15) Let avionom[19] → Летање со авион
  [flight] [by aeroplane.INS] → [flight] [with] [aeroplane]
  (Flying by an aeroplane.)

[18] The encliticised future tense forms (gledat ću / gledaću) are handled equally.
[19] The Croatian normative 'zrakoplov' is also accepted and translated equally.

Table 5.1: Status of `apertium-sh-mk` as of April 11 2011.

Module	Entries / Rules
Serbo-Croatian dictionary	7564
Macedonian dictionary	8672
Bilingual dictionary	9985 (unique)
Transfer rules (1 and 2)	51 + 11
Serbo-Croatian CG	170

While Serbo-Croatian has seven morphological cases, Macedonian has completely replaced its declension system with analytic, prepositional and clitic constructions. The second level of transfer replaces simple noun and adjective phrases with prepositional constructions.

- Inference of definiteness:

  (16) U sastavu Vojske Srbije → Во составот на Српската војска[20]
  [in] [composition] [of Serbian Army] → [in] [composition.DEF] [of Serbian Army]
  (In the composition of the Serbian Army)

  The definite article in Macedonian has no analogy in Serbo-Croatian (except to some extent the definiteness of adjectives). This transfer rule infers definiteness for a common noun preceding a proper noun in genitive.

- A clear definiteness transfer:

  (17) Lijep dan → Ував ден
  [lovely.IND] [day] → [lovely.IND] [day]
  (A lovely day)
  Lijepi dan → Убавиот ден
  [lovely.DEF] [day] → [lovely.DEF] [day]
  (The lovely day)

  For a class of adjectives in Serbo-Croatian definiteness can be distinctly marked. In such cases it can be directly used in translation.

Status

The current status of the language pair is given in Table 5.1.

5.4 Evaluation

This section presents an evaluation of the system performance, with coverage measured on two corpora, and a quantitative analysis.

[20]The article in Macedonian attaches to the first constituent of the noun phrase.

Table 5.2: Coverage

Corpus	Coverage	Std. dev.
Wikipedia (sh+bs+sr+hr)	73.12%	0.36
SETimes (sr + hr)	82.64%	0.38

Coverage

The data for coverage of the Serbo-Croatian analyser is given in Table 5.2. Coverage is naive, it means that for any given form in the source language at least one analysis has been given. The analyser has been tested on a combined Wikipedia corpus, and on a corpus of Serbian and Croatian SETimes articles. The corpora was divided in four parts and average coverage calculated.

Quantitative evaluation

Quantitative evaluation has been performed on four articles from SETimes. The articles were translated by Apertium, and post-edited by a human translator.

The first two articles were selected with nearly full coverage to get an idea of how disambiguation and transfer rules work in ideal circumstances, while the remaining two provide an assessment of the system's practical quality.

The word error rate (WER) and the position-independent error rate (PER) were calculated by the number of changes the human editor needed to make. Results are given in Table 5.3.

Table 5.3: Quantitative evaluation

Article	OOV[1]	Words	WER	PER	Translit.[2]
`setimes.pilots.txt`	0.4%	454	29.9%	20.5%	97.5%
`setimes.tablice.txt`	0.4%	470	48.1%	34.6%	85.2%
`setimes.klupa.txt`	18.1%	480	60.4%	46.8%	82.7%
`setimes.povijest.txt`	14.2%	529	53.4%	40.5%	84.8%

[1] Out of vocabulary words

[2] Baseline WER, obtained by transliteration of the source text

Common problems

Although CG rules successfully rule out quite a lot of grammatically impossible analyses, the number of rules for this language pair is quite low, so disambiguation is not always correct.

Another obvious source of errors are unknown words, which typically disrupt the flow of disambiguation, especially when they occur inside noun phrases.

The definite article is quite difficult to infer. Though in limited cases it can be transferred from definite adjectives, or guessed from specific context, there is e.g. no straightforward way to mark a subject previously introduced in discourse as definite.

Serbo-Croatian cases do not translate consistently to prepositional constructions. A notable example is the partitive vs. possessive genitive. The phrase "čaša vode" can be

translated as "чаша вода" ("a glass of water") or "чашата на вода" (the water's glass).

Both languages have a very free word order of the main constituents. For instance, an adjective can agree with a noun arbitrarily far to the left:

(18) *Vožnja*.N.FEM zrakoplovom ... bila je *odlučujuća*.ADJ.FEM → *Возење*.N.NEUT со авионот ... беше *решавачка*.ADJ.FEM

[The airplane *ride* ... was *decisive*]

If the noun changes gender in translation, the adjective is not matched to it, and retains the source language gender.

5.5 Discussion

This paper presented the design and an evaluation of a language pair for the Apertium platform. It is the first rule-based MT system between Serbo-Croatian[21] and Macedonian, and the morphological analyser and CG module are currently only such open-source resources for the languages.

The system was dubbed by a native speaker as overall fine, there are obvious errors, but the output is legible and easily post-editable.

A significant part of the problems is typical for a system in such an early phase of development. The morphological lexicons for both languages are small, and the same remark can be made for the number of disambiguation rules.

Some ambiguities that arise in analysis of the source language are difficult or impossible to resolve in a simple rule-based manner, which suggests that the system should be combined with machine learning and statistical methods.

In terms of future work the essential task is to increase coverage, to enable working with larger corpora, and to improve the disambiguation rules, which make a significant contribution to translation quality.

Acknowledgements

The development of this language pair was funded as a part of the Google Summer of Code.[22] Many thanks to Jimmy O'Regan, Kevin Unhammer and other Apertium contributors for their help on Apertium, to Tino Didriksen for his help on CG, to Tihomir Rangelov for his advice in the initial phase of this work, and to Dime Mitrevski for his input on the Macedonian language.

[21] It is to our knowledge the first MT system supporting Bosnian.

[22] `http://code.google.com/soc/`

Bibliography

Agić, Ž., M. Tadić, and Z. Dovedan. 2008. Improving part-of-speech tagging accuracy for Croatian by morphological analysis. *Informatica* 32(4):445–451.

Barić, E., M. Lončarić, D. Malić, S. Pavešić, M. Peti, V. Zečević, M. Znika, et al. 1997. *Hrvatska gramatika*. Zagreb: Školska knjiga.

Brown, W. and T. Alt. 2004. *A handbook of Bosnian, Serbian, and Croatian*. SEELRC.

Forcada, M.L., M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J.A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F.M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation* pages 1–18.

Ortiz-Rojas, S.O., M.L. Forcada, and G.R. Sánchez. 2005. Construcción y minimización eficiente de transductores de letras a partir de diccionarios con paradigmas. *Procesamiento de Lenguaje Natural* 35:51–57.

Rangelov, T. 2011. Rule-based machine translation between Bulgarian and Macedonian. In *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation (2011: Barcelona)*.

Sánchez-Martínez, F., J.A. Pérez-Ortiz, and M.L. Forcada. 2008. Using target-language information to train part-of-speech taggers for machine translation. *Machine Translation* 22(1):29–66.

Tyers, F. and M.S. Alperen. 2010. South-East European times: A parallel corpus of Balkan languages. In *Forthcoming in the proceedings of the LREC workshop on "Exploitation of multilingual resources and tools for Central and (South) Eastern European Languages*.

Vitas, D. and C. Krstev. 2004. Intex and Slavonic morphology. *INTEX pour la linguistique et le traitement automatique des langues, Presses Universitaires de Franche-Comté* pages 19–33.

Vitas, D., G. Pavlović-Lažetić, C. Krstev, L. Popović, and I. Obradović. 2003. Processing Serbian written texts: An overview of resources and basic tools. In *Workshop on Balkan Language Resources and Tools*, vol. 21, pages 97–104.

Šnajder, J., Bojana B. Dalbelo Bašić, and M. Tadić. 2008. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing and Management* 44(5):1720–1731.

Chapter 6

Deep evaluation of hybrid architectures: Use of different metrics in MERT weight optimization

Cristina España-Bonet, Gorka Labaka, Arantza Díaz de Ilarraza, Lluís Màrquez, Kepa Sarasola

UPC Barcelona, University of the Basque Country

The process of developing hybrid MT systems is usually guided by an evaluation method used to compare different combinations of basic subsystems. This work presents a deep evaluation experiment of a hybrid architecture, which combines rule-based and statistical translation approaches. Differences between the results obtained from automatic and human evaluations corroborate the inappropriateness of pure lexical automatic evaluation metrics to compare the outputs of systems that use very different translation approaches. An examination of sentences with controversial results suggested that linguistic well-formedness should be considered in the evaluation of output translations. Following this idea, we have experimented with a new simple automatic evaluation metric, which combines lexical and PoS information. This measure showed higher agreement with human assessments than BLEU in a previous study (Labaka et al., 2011). In this paper we have extended its usage throughout the system development cycle, focusing on its ability to improve parameter optimization.

Results are not totally conclusive. Manual evaluation reflects a slight improvement, compared to BLEU, when using the proposed measure in system optimization. However, the improvement is too small to draw any clear conclusion. We believe that we should first focus on integrating more linguistically representative features in the developing of the hybrid system, and then go deeper into the development of automatic evaluation metrics.

6.1 Introduction

The process of developing hybrid MT systems is guided by the evaluation method used to compare outputs of different combinations of basic subsystems. Direct human evaluation is more accurate but unfortunately it is extremely expensive, so automatic metrics have to be used in prototype developing. However, the method should evaluate the outputs of different systems with the same criteria, and these criteria should be as close as possible to human judgment.

It is well known that rule-based and phrase-based statistical machine translation paradigms (RBMT and SMT, respectively) have complementary strengths and weaknesses. First, RBMT systems tend to produce syntactically better translations and deal with long distance dependencies, agreement and constituent reordering in a better way, since they perform the analysis, transfer and generation steps based on syntactic principles. On the bad side, they usually have problems with lexical selection due to a poor handling of word ambiguity. Also, in cases in which the input sentence has an unexpected syntactic structure, the parser may fail and the quality of the translation decrease dramatically. On the other side, phrase-based SMT models usually do a better job with lexical selection and general fluency, since they model lexical choice with distributional criteria and explicit probabilistic language models. However, phrase-based SMT systems usually generate structurally worse translations, since they model translation more locally and have problems with long distance reordering. They also tend to produce very obvious errors, which are annoying for regular users, e.g., lack of gender and number agreement, bad punctuation, etc. Moreover, SMT systems can experience a severe degradation of performance when applied to corpora different from those used for training (*out-of-domain* evaluation).

Because of these complementary virtues and drawbacks several works are being devoted to build hybrid systems with components of both approaches. A classification and a summary of hybrid architectures can be seen in Thurmair (2009). The case we present here is within the philosophy of those systems where the RBMT system leads the translation and the SMT system provides complementary information. Following this line, Habash et al. (2009) enrich the dictionary of a RBMT system with phrases from an SMT system. Federmann et al. (2010) use the translations obtained with a RBMT system and substitute selected noun phrases by their SMT counterparts. Globally, their results improve the individual systems when the hybrid system is applied to translate into languages with a richer morphology than the source.

Regarding the evaluation of the final system and its components, still nowadays, the BLEU metric (Papineni et al., 2002) is the most used metric in MT, but several doubts have arisen around it (Melamed et al., 2003, Callison-Burch et al., 2006, Koehn and Monz, 2006). In addition to the fact that it is extremely difficult to interpret what is being expressed in BLEU (Melamed et al., 2003), improving its value neither guarantees an improvement in the translation quality (Callison-Burch et al., 2006) nor offers such high correlation with human judgment as was believed (Koehn and Monz, 2006).

In the last few years, several new evaluation metrics have been suggested to consider a higher level of linguistic information (Liu and Gildea, 2005, Popović and Ney, 2007, Chan and Ng, 2008), and different methods of metric combination have been tested. Due to its simplicity, we decided to use the idea presented by Giménez and Màrquez (2008), where a set of simple metrics are combined by means of the arithmetic mean.

This work presents a deep evaluation experiment of a hybrid architecture that tries to get the best of both worlds, rule-based and statistical. The results obtained corroborated the known doubts about BLEU. And suggests that the further development of the hybrid system should be guided by a linguistically more informed metric that should be able to capture the syntactic correctness of the rule-based translation, which is preferred by human assessors.

In the next section of this paper we describe the hybrid system. Section 6.3 presents the evaluation experiments: the corpora used in them, and the results of the automatic and manual evaluations. Finally, the last section is devoted to conclusions and future work.

6.2 The hybrid system, SMatxinT

'Statistical Matxin Translator', SMatxinT in short, is a hybrid system controlled by the RBMT translator and enriched with a wide variety of SMT translation options (España-Bonet et al., 2011).

6.2.1 Individual systems

The two individual systems combined in SMatxinT are a rule-based Spanish–Basque system called Matxin (Mayor et al., 2011) and a standard phrase-based statistical MT system based on Moses which works at the morpheme level allowing to deal with the rich morphology of Basque (Labaka, 2010).

Matxin is an open-source RBMT engine, whose main goal is to translate from Spanish into Basque using the traditional transfer model. Matxin consists of three main components: (i) analysis of the source language into a dependency tree structure; (ii) transfer from the source language dependency tree to a target language dependency structure; and (iii) generation of the output translation from the target dependency structure.

The engine reuses several open tools and it is based on an unique XML format for the flow between the different modules, which makes easier the interaction among different developers of tools and resources. The result is an open source software which can be downloaded from matxin.sourceforge.net, and it has an on-line demo[1] available since 2006.

For the statistical system, words are split into several morphemes by using a Basque morphological analyzer/lemmatizer, aiming at reducing the sparseness produced by the agglutinative nature of Basque and the small amount of parallel corpora. Adapting the baseline system to work at the morpheme level mainly consists of training the decoder on the segmented text. The SMT system trained on segmented words generates a sequence of morphemes. So, in order to obtain the final Basque text from the segmented output, a word-generation post-process is applied.

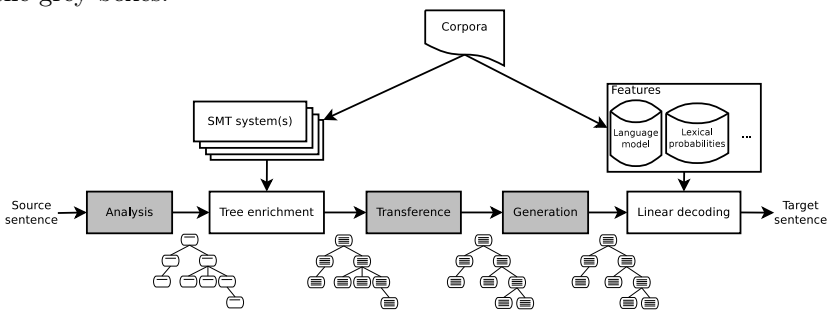
State-of-the-art tools are used in this case. GIZA++ toolkit (Och, 2003) is used for the alignments, SRILM toolkit (Stolcke, 2002) for the language model and the Moses Decoder (Koehn et al., 2007). We used a log-linear functions: phrase translation probabilities (in both directions), word-based translation probabilities (lexicon model, in both directions), a phrase length penalty and the target language model. The language model is a simple 3-gram language model with modified Kneser-Ney smoothing. We also used a lexical reordering

[1] `http://www.opentrad.com`

model ('msd-bidirectional-fe' training option). Parameter optimization was done following the usual practice, i.e., Minimum-Error-Rate Training (Och, 2003), however, the metric used for the optimization is not only BLEU, but it depends on the system as it will be seen.

6.2.2 Hybridisation

Figure 6.1: General architecture of SMatxinT. The RBMT modules which guide the MT process are the grey boxes.



The initial analysis of the source sentence is done by Matxin. It produces a dependency parse tree, where the boundaries of each syntactic phrase are marked. In order to add hybrid functionality two new modules are introduced to the RBMT architecture (Figure 6.1): the tree enrichment module, which incorporates SMT additional translations to each phrase of the syntactic tree; and a monotonous decoding module, which is responsible for generating the final translation by selecting among RBMT and SMT partial translation candidates from the enriched tree.

The tree enrichment module introduces two types of translations for the syntactic constituents given by Matxin: 1) the SMT translation(s) of every phrase, and 2) the SMT translation(s) of the entire subtree containing that phrase. For example, the analysis of the text fragment "*afirmó el consejero de interior*" (said the Secretary of interior) gives two phrases: the head "*afirmó*" (said) and its children "*el consejero de interior*" (the Secretary of interior). The full rule-based translation is "*Barne Sailburua baieztatu zuen*" and the full SMT translation is "*esan zuen herrizaingo sailburuak*". SMatxinT considers these two phrases for the translation of the full sentence, but also the SMT translations of their constituents ("*esan zuen*" and "*herrizaingo sailburuak*"). However, short phrases may have a wrong SMT translation because of a lack of context. So, to overcome this problem SMatxinT also uses the translation of a phrase extracted from a longer SMT translation ("*herrizaingo sailburuak*" in the previous example). So, in order to translate "*afirmó el consejero de interior*" the system has produced 5 distinct phrases, a number that can be increased by considering the n-best lists.

After tree enrichment, the transfer and generation steps of the RBMT system are carried out in a usual way, and a final monotonous decoder chooses among the options. A key aspect for the performance of the system is the election of the features for this decoding. The results we present here are obtained with a set of eleven features. Three of them are usually used as standard SMT features (language model, word penalty and phrase penalty). We also include

four features to show the origin of the phrase and the consensus among systems (a counter indicating how many different systems generated the phrase, two binary features indicating whether the phrase comes from the SMT/RBMT system or not, and the number of source words covered by the phrase generated by both individual systems simultaneously). Finally, we use the lexical probabilities in both directions in two forms: a similar approach to IBM-1 probabilities modified to take unknown alignments into account and a lexical probability inferred from the RBMT dictionary. We refer the reader to España-Bonet et al. (2011) for further details.

6.3 Experiments

The language pair used at evaluation is dictated by the rule-based system and, in this case, Matxin works with the Spanish-to-Basque translation. Basque and Spanish are two languages with very different morphology and syntax.

In previous experiments we evaluated all systems by means of both automatic an manual evaluations (Labaka et al., 2011). Those results corroborated the already known inadequacy of the metrics that measure only the lexical matching for comparing systems that use so different translation paradigms. This kind of metrics are biased in favor of the SMT, as it happens in our evaluation, where the statistical system achieved the best results in the in-domain evaluation, even when it generated the worst translations according to the manual assessment.

To address these limitations of the metrics that are only based on lexical matching, we defined a metric that seeks to check the syntactic correctness, calculating the same expressions but at the PoS level and combining it with lexical BLEU through the arithmetic mean. This metric, which is able to assess the syntactic correctness, has shown a higher level of agreement with human assessments both at document and sentence level.

But evaluation metrics are not only used for comparing different systems, those metrics are also used to guide the development of the systems. Thus, being aware of the problems of BLEU to identify many of the good translations generated by the RBMT system, we used linguistically informed metrics not only on the evaluation, but also in MERT optimization of the linear decoder. So, in addition to individual systems, we will evaluate three different hybrid systems, depending on the metric used in optimization (BLEU, METEOR and BLEU_c, a new defined metric according to Eq. 6.1).

6.3.1 Bilingual and monolingual corpora

The corpus built to train the SMT system consists of four subsets: (1) six reference books translated manually by the translation service of the University of the Basque Country (EHUBooks); (2) a collection of 1,036 articles published in Spanish and Basque by the Consumer Eroski magazine[2] (Consumer); (3) translation memories mostly using administrative language developed by Elhuyar[3] (ElhuyarTM); and (4) a translation memory including short descriptions of TV programmes (EuskaltelTB). All together they made up a corpus of 8 mil-

[2] `http://revista.consumer.es`
[3] `http://www.elhuyar.org/`

lion words in Spanish and 6 mil-ion words in Basque. Table 6.1 shows some statistics on the corpora, giving some figures about the number of sentences and tokens.

Table 6.1: Statistics on the bilingual collection of parallel corpora.

		sentences	**tokens**
EHUBooks	Spanish	39,583	1,036,605
	Basque		794,284
Consumer	Spanish	61,104	1,347,831
	Basque		1,060,695
ElhuyarTM	Spanish	186,003	3,160,494
	Basque		2,291,388
EuskaltelTB	Spanish	222,070	3,078,079
	Basque		2,405,287
Total	Spanish	491,853	7,966,419
	Basque		6,062,911

The training corpus is then basically made up of administrative documents and descriptions of TV programs. For development and testing we extracted some administrative data for the *in-domain* evaluation and we selected a collection of news for the *out-of-domain* study, totaling three sets:

Elhuyardevel and *Elhuyartest*: 1,500 segments each, extracted from the administrative documents.

NEWStest: 1,000 sentences collected from Spanish newspapers with two references.

Additionally, we collected a 21 million word monolingual corpus, which together with the Basque side of the parallel bilingual corpora, builds up a 28 million word corpus. This monolingual corpus is also heterogeneous, and includes text from two sources: the Basque Corpus of Science and Technology (ZT corpus[4]) and articles published by Berria newspaper (Berria corpus).

6.3.2 Automatic Evaluation

In order to perform the automatic evaluation of the translations we use a subset of lexical metrics available in the Asiya evaluation package (Giménez and Màrquez, 2010). Tables 6.2 and 6.3 show the BLEU, TER and METEOR scores for the in-domain test set (Elhuyartest) and the out-of-domain one (NEWStest) respectively[5]. Besides, the tables include the score given by the combination of metrics for the two individual systems (Matxin and SMT) and three hybrid systems SMatxinT that have been optimized against these different metrics. Results of Google Translate[6] are given as control system.

In Labaka et al. (2011) it was shown that a simple combination of n-gram matching metrics at different linguistic levels, such as words and PoS, is more correlated with human

[4] www.ztcorpusa.net/

[5] Figures do not exactly match the ones presented in previous work, since we correct some capitalization errors.

[6] http://translate.google.com/

assessments than just the lexical match. Therefore, we use this new metric, BLEU_c, not only to evaluate the translations but also to optimize the system.

$$\text{BLEU}_c = (\text{BLEU} + \text{BLEU}_{PoS})/2 \tag{6.1}$$

Table 6.2: Automatic evaluation of the in-domain test set, Elhuyartest, for the individual and hybrid systems.

		BLEU	**METEOR**	**TER**	**BLEU$_c$**
Ind. systems	**Matxin**	6.07	27.20	83.49	19.65
	SMT	**16.50**	37.49	70.39	27.64
Control	**Google**	8.19	28.02	78.43	20.73
SMatxinT	**BLEU**	16.09	**38.24**	69.92	**27.95**
	BLEU$_c$	15.36	**38.24**	70.78	27.33
	METEOR	15.87	37.77	**67.77**	27.53

Table 6.3: Automatic evaluation of the out-of-domain test set, NEWStest, for the individual and hybrid systems.

		BLEU	**METEOR**	**TER**	**BLEU$_c$**
Ind. systems	**Matxin**	12.67	36.10	69.16	31.98
	SMT	15.84	37.70	66.52	31.01
Control	**Google**	12.36	32.57	70.44	29.08
SMatxinT	**BLEU**	16.61	39.24	64.50	32.77
	BLEU$_c$	**17.11**	**39.94**	63.84	**33.39**
	METEOR	16.76	39.30	**62.83**	32.50

According to all the automatic metrics Matxin is the worst system both for in-domain and out-of-domain data. The statistical system is worse than the hybrid models for out-of-domain data and shows a similar performance in the in-domain test set. In this case, the BLEU score achieved by SMatxinT is slightly worse than the scores obtained by the single SMT system, but better according to the rest of metrics. The distinct behavior between metrics and the small differences do not allow us to define a clear preference between statistical and hybrid systems. On the contrary, on the out-domain corpora (NEWStest), SMatxinT consistently achieves better scores than any other system.

The use of different metrics in the MERT optimization does not significantly affect the final evaluation. The systems that have been optimized with respect to different metrics obtained very similar results and, when these differences exists, they are not consistent between different evaluation test set or metrics.

In the in-domain evaluation, although the differences are small, the hybrid system optimized on BLEU gets the best results according to BLEU, METEOR and BLEUc. In contrast, the TER metrics assigns the best score to the hybrid system that is optimized on METEOR. It is worth noting that the optimizations on BLEUc and METEOR does not improve results by those metrics.

In the out-domain corpus, although the differences remain small, the results are more stable. In this test set, the hybrid system that achieves the best evaluation is the one optimized on BLEUc, improving the results obtained by the BLEU optimization according to all evaluation metrics. In this corpus, as in the in-domain one, the system optimized on

METEOR achieves results particularly high in the TER metric, which makes if to be the best system according to this metric.

Based on these results, one could state that the low in-domain performance of Matxin penalizes the hybrid system, preventing it to overcome the single SMT system. But, in the out-domain test set, where the scores of Matxin were not so far from the rest of the systems, our hybridization technique was able to combine the best of both systems obtaining the best translation.

6.3.3 Human Evaluation

As in previous works, we contrast those automatic results with a manual evaluation carried out on 100 sentences randomly chosen from the in-domain test set (Elhuyartest) and another 100 sentences chosen from the out-domain test set (NEWStest). The human evaluators are asked to order the 5 translation provided (both individual systems and three different optimizations of SMatxinT). Human evaluators are allowed to determine that various translations are equally good. Depending on how many draws there are, the ranking scope can vary for 1 to 5 (when there is not any draw) to 1 to 1 (when all systems are considered equal). So, we normalized all rankings to the 0-1 scope (where 0 is the best system and 1 is the worst in all cases).

Table 6.4 shows the original and normalized average rankings obtained by each system. According those results, in the in-domain test set Matxin obtains the best ranking, but differences to the three SMatxinT instances are not significant. Those systems that use linguistically motivated metrics (METEOR and BLEU$_c$) in MERT obtain slightly better results than the instance optimized over BLEU. The SMT system, in turn, obtains the worst ranking. On the other hand, in the out-domain evaluation the differences are bigger: Matxin, the rule-based system, clearly outperforms the hybrid systems and these ones outperform the statistical system. The differences between different optimizations of SMatxinT are not significant.

Table 6.4: Real and normalized mean of the ranking manually assigned to each system.

		Elhuyartest		**NEWStest**	
		ranking	**norm.**	**ranking**	**norm.**
Ind. systems	**Matxin**	2.070	0.396	1.705	0.275
	SMT	2.510	0.532	2.605	0.625
SMatxinT	**BLEU**	2.165	0.423	2.210	0.485
	BLEU$_c$	2.085	0.399	2.110	0.445
	METEOR	2.095	0.403	2.125	0.470

Each sentence, 100 in each test set, has been assessed by two evaluators. Agreement between evaluators is difficult to check, as qualitatively small changes between them can produce multiple single changes in the precedence numbers in the ranking. For example, between the following two rankings

Matxin 1, BLEU 2, BLEU$_c$ 2, METEOR 2, SMT 3
Matxin 1, BLEU 2, BLEU$_c$ 3, METEOR 3, SMT 4

three precedence numbers are changed, but there is only a single qualitative difference (in

the second ranking the system trained with BLEU is better than those trained with BLEUc and METEOR).

To make the rankings more comparable we discretized the assigned ranking into 4 possible values: *best*, *intermediate*, *worst* and *all-draw*. The *best* and *worst* values mean that the system has been asserted as the best or the worst system. The *intermediate* value is assigned to other systems. In the cases that all systems are assigned to the same rank the *all-draw* value is assigned.

Table 6.5 shows the times that both evaluators assigned the same discrete ranking. Between brackets, the times that each evaluator assigns this ranking is shown. In some cases, the agreement is high, as when Matxin is claimed as the best out-domain system, *47(51+64)*. But generally the agreement is not very high.

Table 6.5: Discrete ranking results. Figures correspond to agreement of both evaluators, between brackets each evaluator's figures.

	Elhuyartest			
	best	intermediate	worst	all-draw
Matxin	24 (34+42)	9 (26+19)	20 (38+32)	0 (2+7)
SMT	9 (22+23)	7 (31+23)	30 (45+47)	0 (2+7)
BLEU	8 (27+19)	22 (52+43)	8 (19+31)	0 (2+7)
BLEU$_c$	12 (27+18)	29 (55+45)	7 (16+30)	0 (2+7)
METEOR	6 (28+19)	24 (54+47)	6 (16+27)	0 (2+7)
	NEWStest			
	best	intermediate	worst	all-draw
Matxin	47 (51+64)	4 (22+12)	10 (25+19)	0 (2+5)
SMT	7 (20+11)	6 (21+25)	41 (57+59)	0 (2+5)
BLEU	11 (28+15)	27 (44+43)	21 (26+37)	0 (2+5)
BLEU$_c$	12 (27+17)	28 (50+44)	15 (21+34)	0 (2+5)
METEOR	11 (26+16)	26 (46+42)	18 (26+37)	0 (2+5)

These results further demonstrate the equality of the systems, thickened by the lack of agreement between evaluators. In addition, it also shows some interesting results, as the fact that even in-domain the RBMT system produces more sentences tagged as the best translation. But the system also generates a high number of sentences labeled as the worst translation. So, in the overall assessment it fails to distance itself from the hybrid systems (which produce less 'best' translations, but also less 'worst' translations).

6.4 Conclusions

In this work we present an in-depth evaluation of SMatxinT, a hybrid system that is controlled by the RBMT translator and enriched with a wide variety of SMT translation options. The results of the human evaluation, where the translation of all the individual systems was ranked, established that Matxin, the RBMT system, achieved the best performance followed by SMatxinT, while the SMT system generated the worst translations.

Those results, very far from what the automatic metrics show, corroborate the already known inadequacy of the metrics that measure only the lexical matching for comparing

systems that use so different translation paradigms. This kind of metrics is biased in favor of the SMT, as it happens in our evaluation, where the statistical system achieves the best results in the in-domain evaluation, even when it generates the worst translations according to the manual assessment.

To address these limitations of the metrics that are only based on lexical matching, we defined a metric that seeks to ensure the syntactic correctness, combining lexical BLEU with PoS matching information. At the time of combining these metrics, we opted for simplicity and we used the arithmetic mean of BLEU in words and PoS. This method, despite its simplicity, has already shown its suitability before. Our combined metric is simple and able to maintain a higher correlation with manual evaluation than the usual lexical metrics, while ensures the lexical matching.

But evaluation metrics are not only used for comparing different systems, those metrics are also used to guide the optimization of the systems. In practical terms, in our hybrid architecture, we used those metrics to identify the features that are able to differentiate the best translation proposed by different approaches. Thus, being aware of the problems of BLEU to identify many of the good translations generated by the RBMT system, we used linguistically informed metrics not only on the evaluation, but also in MERT optimization of the linear decoder. So, in addition to individual systems, we evaluate three different hybrid systems, depending on the metric used in optimization. According to the results achieved, the use of different metrics in optimization has low impact in translation quality. Although the use of BLEU_c in optimization slightly improves the results achieved by manual evaluation, this improvement is too small to draw clear conclusions.

We consider that the minimal differences that exist between different optimizations are due to the lack of linguistic features at monotonous decoding. Current 11 features are mainly devoted to characterize the origin system of a given phrase and the probabilities for the lexical translation. In MERT optimization, the evaluation metrics are only used to find out which of the features present in the decoding are the most useful at generating the final translation. So, if there are no features which depend on the PoS in our case, or on higher level information such as the type of chunk, they may not be informative enough to strengthen the metric. In this case, optimization has little room for improvement.

Given these results, the need to provide more in-depth linguistic information to the evaluation metrics is undeniable. But, since we carry out our research in translation into Basque, we have at our disposal few linguistic tools, much less than for languages like English. Future work should first focus on integrating more representative linguistic features in the hybrid system which allow a qualitative leap in the translations quality. Then the small improvements reported here could be confirmed or ruled out.

Acknowledgments

This research has been partially funded by the Spanish Ministry of Education and Science (OpenMT-2, TIN2009-14675-C03) and the EC Seventh Framework Programme under grant agreement numbers 247914 (MOLTO project, FP7-ICT-2009-4-247914) and 247762 (FAUST project, FP7-ICT-2009-4-247762).

Bibliography

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the International Conference of European Chapter of the Association for Computational Linguistics (EACL)*, pages 249–256.

Chan, Yee Seng and Hwee Tou Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of ACL-08: HLT*, pages 55–62. Columbus, Ohio: ACL.

España-Bonet, Cristina, Gorka Labaka, Arantza Díaz de Ilarraza, Lluis Màrquez, and Kepa Sarasola. 2011. Hybrid machine translation guided by a rule–based system. In *Proceedings MT Summit XIII*. Xiamen, China.

Federmann, C., A. Eisele, Y. Chen, S. Hunsicker, J. Xu, and H. Uszkoreit. 2010. Further experiments with shallow hybrid mt systems. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 77–81. Uppsala, Sweden: ACL.

Giménez, Jesús and Lluís Màrquez. 2008. A smorgasbord of features for automatic MT evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 195–198. Columbus, Ohio: ACL.

Giménez, Jesús and Lluís Màrquez. 2010. Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. *The Prague Bulletin of Mathematical Linguistics* (94):77–86.

Habash, Nizar, Bonnie Dorr, and Christof Monz. 2009. Symbolic-to-statistical hybridization: extending generation-heavy machine translation. *Machine Translation* 23:23–63.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Prague, Czech Republic.

Koehn, Philipp and Christof Monz. 2006. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the 1st Workshop on Statistical Machine Translation*, pages 102–121.

Labaka, Gorka. 2010. *EUSMT: Incorporating Linguistic Information into SMT for a Morphologically Rich Language. Its use in SMT-RBMT-EBMT hybridation*. Ph.D. thesis, University of the Basque Country.

Labaka, Gorka, Arantza Díaz de Ilarraza, Cristina España-Bonet, Lluis Màrquez, and Kepa Sarasola. 2011. Deep evaluation of hybrid architectures: simple metrics correlated with human judgements. In *Proceedings of International Workshop on Using Linguistic Information for Hybrid Machine Translation LIHMT*. Barcelona.

Liu, Ding and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 25–32.

Mayor, Aingeru, Iñaki Alegria, Arantza Díaz de Ilarraza, Gorka Labaka, Mikel Lersundi, and Kepa Sarasola. 2011. Matxin, an open-source rule-based machine translation system for basque. *Machine Translation* 25:53–82.

Melamed, I. Dan, Ryan Green, and Joseph P. Turian. 2003. Precision and Recall of Machine Translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 61–63. Morristown, NJ, USA: ACL.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Popović, Maja and Hermann Ney. 2007. Word error rates: decomposition over pos classes and applications for error analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 48–55. Stroudsburg, PA, USA: ACL.

Stolcke, A. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference of Spoken Language Processing*, vol. 2, pages 901–904.

Thurmair, G. 2009. Comparing different architectures of hybrid machine translation systems. In *Proceddings of MT Summit XII*.