Implementing an efficient and scalable Service-Oriented Architecture for the Apertium Rule-Based Machine Translation platform

Pasquale Minervini
Dipartimento di Informatica
Università degli Studi di Bari
Via E. Orabona 4, 70125 Bari, Italy
p.minervini@gmail.com

Abstract

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed services that may be under the control of different ownership domains and implemented using various technology stacks. In some contexts, an organization using an IT infrastructure implementing the SOA paradigm can take a great benefit from the integration, in its business processes, of efficient Machine Translation services to overcome language barriers. This paper describes the architecture and the design patterns used to develop a Machine Translation service that is efficient, scalable and easy to integrate in new and existing business processes; our service relies on Apertium, a Free/Open-Source Rule-Based Machine Translation platform.

1 Introduction

Service Oriented Architecture is an architectural paradigm providing a set of principles of governing concepts used during phases of systems development and integration. In such an architecture, functionalities are packaged as interoperable, loosely coupled services that may be used to build infrastructures enabling those with needs (consumers) and those with capabilities (providers) to interact across different domains of technology and ownership.

Several new trends in the computer industry rely upon SOA as their enabling foundation, including the automation of Business Process Management (BPM) and the multitude of new architecture and design patterns generally referred to as Web 2.0 (O'Reilly, 2005).

In some contexts, an organization using an IT infrastructure implementing the SOA paradigm can take a great benefit from the integration, in its business processes, of an efficient Machine Translation service to overcome language barriers; for instance, it could be integrated in collaborative enviroments where people, who have no language in common, attempt to communicate with each other, or in knowledge extraction processes, where data is not available in a language that can be comprehended by the domain experts or the knowledge extraction tools beign used.

Case Study 1 - UMLS® concepts identification in non-English medical documents: *MetaMap (Aronson, 2001) is an application that allows mapping text to UMLS® Metathesaurus®*[1] *concepts, which have proved to be useful for many applications, including decision support systems, management of patient records, information retrieval and data mining within the biomedical domain.*

Currently MetaMap is only available for English free text, which makes it difficult the use of UMLS® Metathesaurus® to represent concepts

[1] The UMLS® Metathesaurus® (Schuyler et al., 1993) provides a representation of biomedical knowledge consisting of concepts classified by semantic type and both hierarchical and non-hierarchical relationships among the concepts.

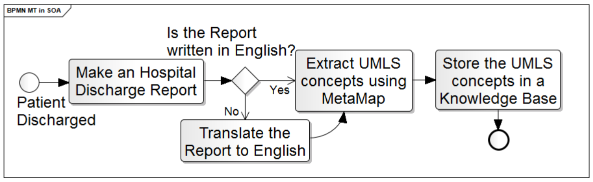
BPMN MT in SOA
Patient Discharged
Make an Hospital Discharge Report
Is the Report written in English?
Yes
No
Translate the Report to English
Extract UMLS concepts using MetaMap
Store the UMLS concepts in a Knowledge Base

Figure 1: Representation of a business process in which a clinical document, if written in a language different than English, is fist translated to English and then processed using MetaMap to extract UMLS® concepts.

from biomedical texts written in languages other than English. A possible way to overcome this limitation consists in using Machine Translation techniques (possibly with dictionaries, translation rules, lexical selection techniques etc. specific for the biomedical domain) to translate the free text from its original language to English, and then process it as in Figure 1. This approach is also discussed in Carrero et al. (2008).

Case Study 2 - Supporting creation of user-generated content: *Wikipedia is an online, multilingual, volunteer-edited encyclopedia. "There are currently 262 language editions of Wikipedia; of these, 24 have over 100,000 articles and 81 have over 1,000 articles" (Wikipedia, 2009). Although access to technology is also an important factor, the number of available articles in a particular language's Wikipedia corresponds somewhat to the number of available speakers of that language.*

In many cases, closely related languages are mutually intelligible (Tyers et al., 2009), and even a prototype Machine Translation system can produce accurate translations (Armentano-Oller and Forcada, 2006). This seems to be the case with Nynorsk and Bokmål (Unhammer et al., 2009), where users of the Nynorsk Wikipedia have made contributions to the system's lexicon, to assist in their translation of articles from the larger Bokmål Wikipedia to the Nynorsk Wikipedia.

However, the current use of Machine Translation on the Wikipedias of marginalized languages is a somewhat error-prone process, where the original text is manually copied from the source Wikipedia, translated off-line, and pasted as a new article to the target Wikipedia. By providing an efficient, easily integrated service, we hope to remove some of the accidental errors inherent to this process.

In addition, the service's logging facilities may be used to improve Machine Translation quality, by incorporating user feedback (Chin and Rosart, 2008), in the form of corrections to the translated text. Small corrections to Wikipedia articles have been used in the construction of error corpora (Miłkowski, 2008), which can then be used to augment translation rules, or in the creation of statistical post-correction systems (Dugast et al., 2007).

We realized a prototype Machine Translation service relying on Apertium (Armentano-Oller et al., 2005), a Free/Open-Source Rule-Based Machine Translation platform being developed with funding from the Spanish government and the government of Catalonia at the Universitat d'Alacant (University of Alicante), for its translation capabilities, and on N-Gram Based Text Categorization (Cavnar and Trenkle, 1994) for language recognition. Our decision to prefer a Rule-Based Machine Translation system to a Statistical or an Example-based Machine Translation system was motivated by the following reasons:

- Statistical Machine Translation systems tend to produce text that appears more "natural" than that produced by Rule-Based ones, with the result that "fluency" can outweight "fidelity", but a natural and fluent translation is not necessairly completely faithful to the original text;

- In Rule-Based Machine Translation systems, linguistic knowledge can be encoded explicitly in the form of linguistic data, so that both humans and automatic systems can process it (this feature can be a great benefit when using domain-specific linguistic knowledge);

- Rule-Based Machine Translation systems tend to produce more "mechanical" translations, so their errors tend also to be more evident;

- Experts who have designed a Rule-Based Machine Translation system find it much easier to diagnose and repair sources of translation errors, like wrong rules in modules or wrong entries in dictionaries.

Efficiency and scalability are also a critical for our service since, especially in collaborative enviroments, it should be able to sustain an heavy load of traffic. In this paper, we will describe what techniques and design patterns we used to implement an efficient and scalable machine translation service, and we will compare it to other existing systems.

2 Service's APIs

Our service provides the following capabilities:

Translation - for automatic translation of free text from a source language to a destination language;

Language Recognition - for automatic language guessing of free text;

In SOA, interoperability between services is achieved by using standard languages for the description of service interfaces and the communications among services. A widely accepted technique for implementing SOA consists in making use of Web Services (Erl, 2005); a Web Service is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." (Brown and Haas, 2004).

Alternative standards to SOAP are XML-RPC (Winer, 1999), a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism, and Representational State Transfer (REST) (Fielding, 2000), a style of software architecture for distributed hypermedia systems such as the World Wide Web.

Our service provides XML-RPC, SOAP and REST interfaces to the Translation and Language Recognition functionalities. All the interfaces follow the schema outlined in Table 1 and Table 2 to expose, respectively, the Translation and the Language Detection functionalities; those can be subsumed by the following methods:

Translate	
parameters	**Text**
	Source Language
	Destination Language
returns	**Translation**
	Detected Source Language

Table 1: Parameters and return value(s) for the Translate method.

Detect	
parameters	**Text**
returns	**Detected Language**

Table 2: Parameters and return value(s) for the Detect method.

Translate: receives three parameters called **Text**, **Source Language** and **Destination Language** containing, respectively the text to be translated, the source language and the destination language, and returns a **Translation** value containing the translated text; if the source language is omitted, then language recognition is used to guess it, and the guessed language is returned in the **Detected Source Language** value.

Detect: receives three parameters called **Text** containing free text, and returns a **Detected Language** value containing the language used by the text.

In both methods, languages are represented by their ISO 639-1 (ISO:639-1, 2002) code.

3 Service's Internal Architecture

Generally speaking, a translation process can be considered as the following sequence of steps:

- **Decoding** the meaning of the source text;
- **Re-Encoding** this meaning in the destination language.

Behind this apparently simple procedure, there is a complex cognitive operation: to decode the meaning of some free text in its entirety, the translator needs to interpret and analyse all the features present in the text, which requires in-depth knowledge of *grammar*, *semantics*, *syntax*, *idioms* etc. of both source and destination languages and the culture of their speakers.

For this reason, according to Arnold et al. (1993), Rule-Based Machine Translation systems make use, during the translation process, of *intermediate representations* (such as an *interlingua*) trying to capture the “meaning” of the original sentence in order to generate the correct translation. A Rule-Base Machine Translation system, according to the nature of its intermediary representation, is described as *Interlingual* if the original text is transformed into an interlingua (i.e. an abstract language-independent representation), or *Transfer-based* if the intermediate representation has some dependences on the language pair involved.

In general, Rule-Based Machine Translation systems apply a set of *linguistic rules* during the translation process, which are defined as correspondances between the structure of the source language and the structure of the destination language. Therefore, the translation is obtained by analysing the source text for morphology and syntax (and eventually semantics) to create an intermediate representation, and then by using bilingual dictionaries and grammatical rules.

Specifically, Apertium is a Transfer-based Machine Translation system which uses Finite-State Transducers (Forcada et al., 1999) for lexical processing, Hidden Markov Models for Part-of-Speech tagging and Finite-State-Based Chunking for structural transfer. Its translation engine consists of an *assembly line*, composed of the following modules:

Formatters - which handle format-specific information with respect to text to be translated;

Morphological Analyser - which tokenizes the text in *surface forms* and delivers, for each surface form, one or more *lexical forms* consisting of lemma, lexical category and informations about morphological inflection;

Part-of-Speech Tagger - which chooses one of the analyses of an ambiguous word, according to its context;

Lexical Transfer Module - which reads each lexical form of the surface form and delivers the corrisponding destination language lexical form;

Structural Transfer Module - which detects and processes patterns of words that need special processing due to grammatical divergences between two languages;

Morphological Generator - that, from a lexical form in the destination language, generates a suitably inflected surface form;

Post-Generator - that performs some orthographic operations in the destination language such as contractions;

In general, those modules often rely on resources whose acquisition and release can be computationally expensive, since they may require access to external data sources and further elaborations; for example:

- Production rule systems relying on *RETE*-like pattern matching algorithms (Forgy, 1982), during the acquisition of a set of rules, build a network of nodes to reduce the time complexity of many-to-many pattern matching operations;

- Resources as bilingual dictionaries are often mapped to hash tables during their acquisition, to reduce the computational complexity of lookup operations;

- Modules dedicated to handling format-related informations can make use of regular expression that often need to compiled into, for example, the corrisponding Finite State Automatas, before they can be effectively used.

To prevent the frequent acquisition and release of the aforementioned resources, our service makes use of the *Pooling Pattern* (Kircher and Jain, 2004), in which multiple instances of one type of resource are managed in a pool. This pool of resources allows for reuse when resource clients release resources they no longer need: released resources are put back into the pool and

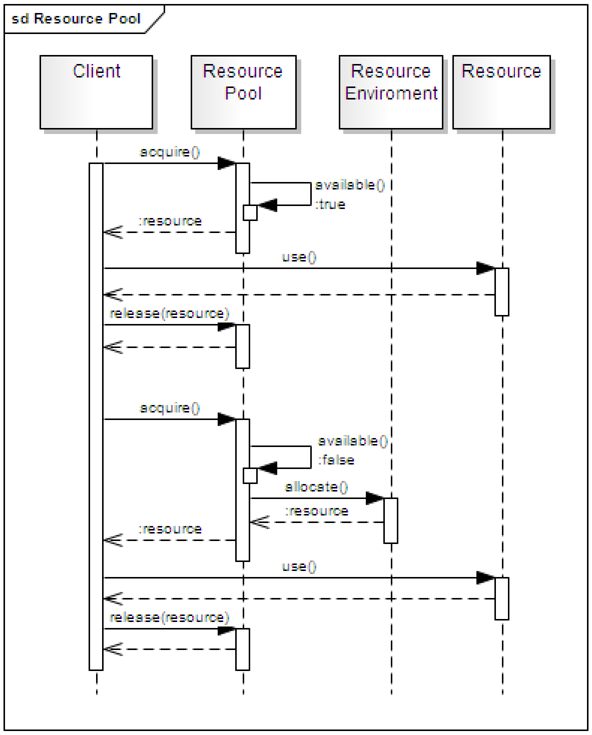

Figure 2: Sequence Diagram describing how acquisition and release of resources works in a system implementing the Pooling Pattern: recycled objects are managed in a pool of resources, which allows pool clients to acquire them, and release them back to the pool when they are no longer needed.

made available to resource clients needing them, as shown in Figure 2.

To improve efficiency, the resource pool can eagerly acquire a number of resources after its creation; then, if demand exceeds the number of available resources in pool, more resources can be lazily acquired.

There are various valid approaches to free unused resources, like those consisting of monitoring the use of a resource and controlling its lifecycle by using strategies such as “Least Recently Used” or “Least Frequently Used”, or introducing a *lease* for every resource that specifies a time duration for which a resource can remain in the pool.

Relying on a resource pool results in the following improvements for our Rule-Based Machine Translation service:

Performance - by preventing repetitious acquisition and release of resources;

Predictability - because direct acquisition of a resource from the resource enviroment can lead to unpredictable results;

Stability - because repetitious acquisition and release of resource can increase the risk of system instability (for example, repetitious acquisition and release of memory can lead to fragmentation problems);

Scalability - since resources can be used also in different types of translation tasks, avoiding the allocation of a complete set of resources for each different translation task (for example, translation tasks on different language pairs using different dictionaries can make use of the same transfer rules and lexical selectors);

4 Results

To evaluate the efficiency of our service, which we will refer to as `apertium-service`, we compared the time it requires to compute and answer to a translation request from the Spanish language to the English language with the time required by the following systems:

- `apertium`, a console application implemented as a part of the Apertium project;
- `moses` (Koehn et al., 2007), a Open Source Statistical Machine Translation system;
- `moses-service`, a service relying on `moses`.

The translation model used by `moses` and `moses-service` has been trained on the well-known Europarl (Koehn, 2002) corpus by using SRILM (Stolcke, 2002), a toolkit for building and applying statistical language models. Language models have been then compiled into binary format using IRSTLM (Federico et al., 2008), and trained to minimize the error rate on a set of sentences from the same corpus.

All the experiments were run on a server with 4 2GHz Dual-Core AMD Opteron^{TM} processors and 4GB of main memory, using the GNU/Linux operating system. Both `apertium-service` and `moses-service` were accepting translation requests in XML-RPC format, and the free text used for timing all the systems was also taken

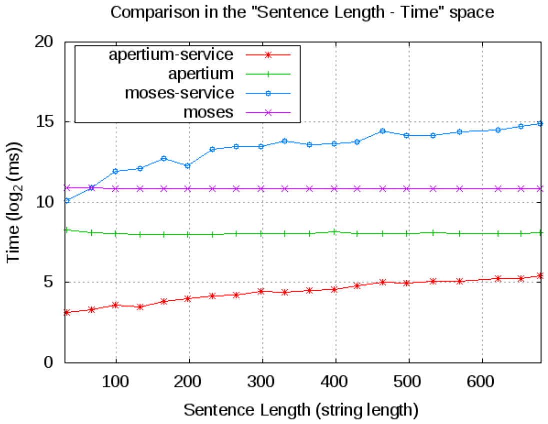

Figure 3: Comparison in the “Sentence Length - Time” space between `apertium`, `apertium-service`, `moses` and `moses-service`; measurements are in $log_2(string\ length)$ for the Sentence Length dimension and in $log_2(ms)$ for the Time dimension.

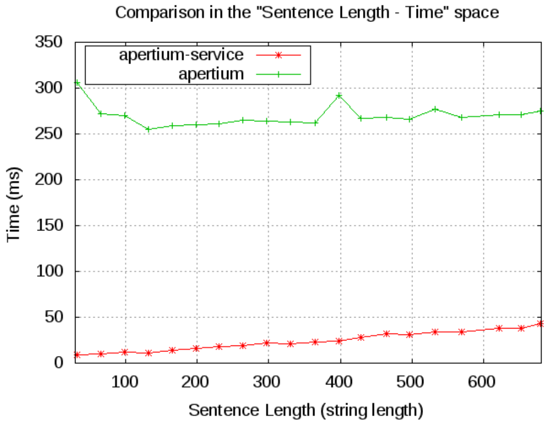

Figure 4: Comparison in the “Sentence Length - Time” space between `apertium` and `apertium-service`; measurements are in $string\ length$ for the Sentence Length dimension and in ms for the Time dimension.

from Europarl corpus. Figure 3 shows the time required to translate increasingly longer sentences for all systems (values in the Time dimension are shown on a logarithmic scale), and Figure 4 only for `apertium-service` and `apertium`.

Scalability for `apertium` and `apertium-service` have been evaluated by calculating the average time required by the two systems to answer to 1024 translations requests sequentially sent by a variable number of clients; the requests consisted to translating the longest sentence from the Europarl evaluation corpus (679 characters) from the Spanish language to the English language. Figure 5 shows the results of this comparison.

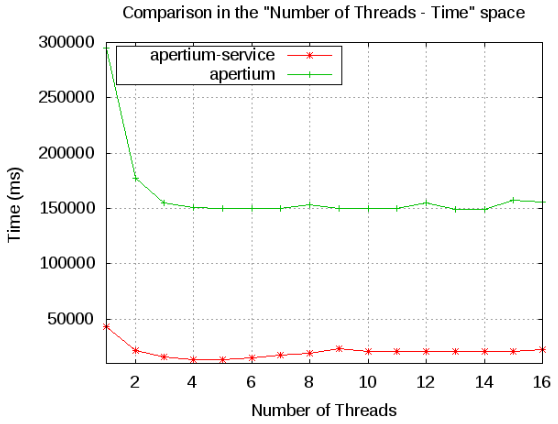

Figure 5: Comparison in the “Number of Threads- Time” space between `apertium` and `apertium-service`.

5 Conclusions and Future Work

We presented `apertium-service`, a Rule-Based Machine Translation service based on Apertium, a Free/Open-Source Rule-Based Machine Translation platform. It was shown to be competitive, in efficiency and scalability, with other systems based on various Machine Translation paradigms.

Source code for our service is released under the GNU General Public License and its available on the Apertium SVN repository at the following address: `http://apertium.svn.sourceforge.net/svnroot/apertium/trunk/apertium-service`.

Acknowledgements

Development for this project was funded as part of the Google Summer of Code[2] programme. Additionally, many thanks go to Jimmy O’Regan, Francis Tyers and others involved in The Apertium Project, for their constant help.

References

Armentano-Oller, C., Corbí-Bellot, A. M., Forcada, M. L., Ginestí-Rosell, M., Bonev, B., Ortiz-Rojas, S., Pérez-Ortiz, J. A.,

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

Ramírez-Sánchez, G., and Sánchez-Martínez, F. (2005). An open-source shallow-transfer machine translation toolbox: consequences of its release and availability. In *OSMaTran: Open-Source Machine Translation, A workshop at Machine Translation Summit X*, pages 23–30.

Armentano-Oller, C. and Forcada, M. L. (2006). Open-source machine translation between small languages: Catalan and aranese occitan. In *Strategies for developing machine translation for minority languages (5th SALTMIL workshop on Minority Languages)*.

Arnold, D., Balkan, L., Meijer, S., Humphreys, R., and Sadler, L. (1993). *Machine Translation: an Introductory Guide*. Blackwells-NCC, London.

Aronson, A. R. (2001). Effective mapping of biomedical text to the umls metathesaurus: the metamap program. *Proc AMIA Symp*, pages 17–21.

Brown, A. and Haas, H. (2004). Web services glossary. World Wide Web Consortium, Note NOTE-ws-gloss-20040211.

Carrero, F. M., Cortizo, J. C., Gómez, J. M., and de Buenaga, M. (2008). In the development of a spanish metamap. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1465–1466, New York, NY, USA. ACM.

Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

Chin, J. and Rosart, D. (2008). Machine translation feedback.

Dugast, L., Senellart, J., Simard, M., and Koehn, P. (2007). Statistical post-edition on systran rule-based translation system. In *ACL Workshop on Statistical Machine Translation*.

Erl, T. (2005). *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR.

Federico, M., Bertoldi, N., and Cettolo, M. (2008). Irstlm: an open source toolkit for handling large scale language models. In *Interspeech 2008*, pages 1618–1621. ISCA.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

Forcada, M., Garrido-Alenda, A., Iturraspe, A., Montserrat-Buendia, S., and Pastor, H. (1999). *A compiler for morphological analysers and generators based on finite-state transducers*. CONGRESO DE LA SOCIEDAD ESPAÑOLA PARA PROCESAMIENTO DEL LENGUAJE NATURAL, Lleida.

Forgy, C. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligences*, 19(1):17–37.

ISO:639-1 (2002). Iso 639-1:2002 – codes for the representation of names of languages – part 1: Alpha-2 code.

Kircher, M. and Jain, P. (2004). *Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management*. Wiley.

Koehn, P. (2002). Europarl: A multilingual corpus for evaluation of machine translation. Draft.

Koehn, P., Hoang, H., Mayne, A. B., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. pages 177–180.

Miłkowski, M. (2008). Automated building of error corpora of polish. In Lewandowska-Tomaszczyk, B., editor, *Corpus Linguistics, Computer Tools, and Applications — State of the Art. PALC 2007*, pages 631–639. Internationaler Verlag der Wissenschaften.

O'Reilly, T. (2005). What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.

Schuyler, P. L., Hole, W. T., Tuttle, M. S., and Sherertz, D. D. (1993). The umls metathesaurus: representing different views of biomedical concepts. *Bull Med Libr Assoc*, 81(2):217–222.

Stolcke, A. (2002). Srilm – an extensible language modeling toolkit.

Tyers, F. M., Wiechetek, L., and Trosterud, T. (2009). Developing prototypes for machine translation between two sámi languages. In *Proceedings of the 13th Annual Conference of the European Association of Machine Translation, EAMT09*.

Unhammer, K., Trosterud, T., and Tyers, F. (2009). Reuse of free resources in machine translation between nynorsk and bokmål.

Wikipedia (2009). Wikipedia — wikipedia, the free encyclopedia. [Online; accessed 28-September-2009].

Winer, D. (1999). XML/RPC specification. Technical report, Userland Software.