Data Repository Organization and Recuperation Process for Multilingual Lexical Databases[1]

Jérôme Godard, Mathieu Mangeot, Frédéric Andrès

Software Research Division
National Institute of Informatics
Hitotsubashi 2-1-2-1407, Chiyoda-ku
101-8430 Tokyo, Japan
Tel. +81 (0)3 4212 2542
Fax.: +81 (0)3 3556 1916
e-mail : {jerome, mangeot, andres}@nii.ac.jp

Abstract

This paper describes the data management in multilingual lexical databases. Since NLP systems are using lexical data, the amount of work to build them is huge. That is the reason why it is important to use rigorous powerful systems and to be able to get data for a minimal cost. As an open source project, Papillon is reusing existing lexical data and wants to make volunteers collaborate. To make it possible, it combines state of the art concepts in the field of linguistic and computing.

1 Introduction

Many NLP systems are based on lexical data. Creating such kind of data requires an amazing knowledge and a huge amount of work. Furthermore, the existing lexical data has generally been developed for a specific purpose and can't be reused easily in other applications.

In order to overcome these shortcomings, the Papillon project applies some linguistic concepts, tools and methods to develop multipurpose, multilingual lexical data collaboratively on Internet. This data is complete and detailed enough to be finally used either by NLP systems (MT engines for example) or by human users (language learners, translators…).

After presenting the Papillon project and its purpose in section 2, we will describe the management of the data into a multilingual lexical database in section 3. Then, according to that method, we will explain in section 4 how to reuse efficiently existing dictionaries. Finally, section 5 concludes the paper.

2 The Papillon Project

The Papillon Project is a two-year old project born as a French-Japanese cooperation. It is now an international project building a worldwide open multilingual lexical database.

2.1 Motivations

The first motivation of the Papillon Project was to fill the gap in the domain of digital French-Japanese dictionaries and to use very powerful multilingual lexical linguistic concepts. But the real lack of multilingual lexical resources available on Internet proved the need to take other languages in consideration: a lot of free dictionaries are available but very few of them imply more than 2 languages. Moreover, most of these dictionaries include English as one of their languages. Furthermore, the existing dictionaries often lack essential information for beginners or NLP systems. Another point contributing to this lack: the high costs of development of large lexical resources for NLP involves also a high price dissuasive for the end-user.

2.2 Description of a multilingual lexical database

Two different ways are possible to develop a multilingual lexical database, either a bilingual or

[1] This research has been partially supported by National Institute of Informatics (NII) under the leadership framework, by the Multimedia Annotation Consortium, and by the JSPS

an interlingual approach. The most interesting one is quite obviously the interlingual one because it allows to build only once one dictionary for each language (n languages implies n monolingual dictionaries and one interlingual database); in the case of the bilingual method, it is necessary to create a dictionary for each couple (n languages implies $\left(\sum_{i=1}^{n-1} i\right)$, i.e. $\frac{n(n-1)}{2}$ dictionaries). But the interlingual approach requires a strong methodology to become effective (Jérôme Godard, 2001).

Some partners of the Papillon project were involved in researches on the definition of structures and tools to handle multilingual lexical databases. They were looking for an opportunity to apply these researches on real scale lexical data.

2.3 Structure of the entries

Papillon entries monolingual structure is taken from the DiCo database, based on the research of Igor A. Mel'cuk and Alain Polguère, Université de Montréal, Montréal, Canada (Alain Polguère, 2000) and encoded in XML. Dictionaries' entries are semantic-dependent; indeed, each meaning of one word (vocable) represents an entry. These semantical entities are called lexies. They are used to build monolingual dictionaries and to give those databases a lexical approach. Then, the lexies of various monolingual dictionaries need to be linked. This is done via a pivot architecture that connects the lexies ones with the others; this multilingual architecture is based on the relation between the lexies (Gilles Sérasset, 1994) implemented with acceptions called axies.

2.4 Collaborative Development over Internet

Most partners were participating, as computer scientists, to the development of open source products. With the democratisation of Internet access in a lot of countries, came the opportunity to apply the open source principles to the development of a multipurpose, multilingual lexical database.

Cooperation projects for bilingual dictionaries are already going on such as EDICT, a Japanese-English dictionary lead by Jim Breen (2001) for more than 10 years and more recently, SAIKAM, a Japanese-Thai dictionary (Vutichai Ampornaramveth, 2000). Some cooperations have also been launched in order to add Thai (Lexibase), Lao and Vietnamese.

With the Papillon project, the dictionary is extended to a multilingual lexical database. Volunteers will find lexicons developed by others and some tools to complete or correct the Papillon multilingual dictionary. Users will also be able to define their own personal views of the database.

3 Lexical Data Management

3.1 Issues

From our work in the field of lexical data recuperation process, we got some fundamentals principles about storing and manipulating digital dictionaries in order to build a multilingual lexical database.

They are mainly due to the fact that many dictionaries are used. Moreover, those dictionaries evolve from the modification we bring them to make the data relevant to our XML data schema.

There are also many people within the project who have to be able to access quickly and easily the content they are looking for; people who are also evolving, it means some new contributors are continually taking part in the project.

Therefore, we decided to use a strong data repository definition with efficient and effective naming rules.

3.2 Data Repository[2]

The lexical data repository of the Papillon project is divided into 4 subdirectories (Mathieu Mangeot-Lerebours 2001):

- *Administration* contains guidelines and administrative files
- *Hell* (data in original format)
- *Purgatory* (data in XML & UTF-8)
- *Paradise* (data in Papillon format)

An example of that structure is given in the following figure:

[2] NB all the lexical data stored in the repository are free of rights or protected by a GPL-like licence.

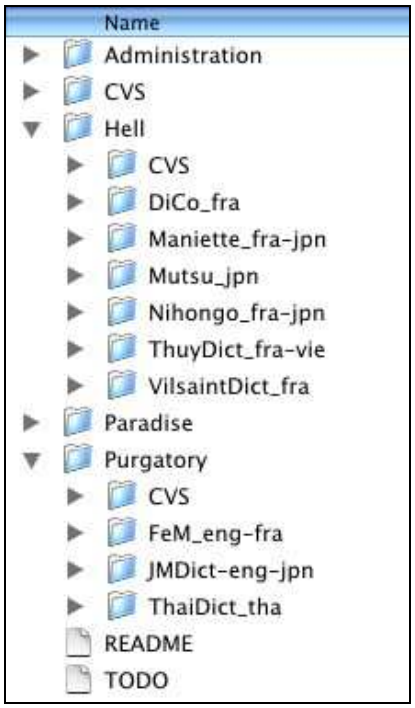

Figure 1. Papillon Data Repository.

An illustration of our goal would be to say that any contributor should be able to understand the data organization after typing "*ls*".

3.2.1 Hell Directory

This directory contains lexical data in their original format. When a dictionary is received, it is first stored there while waiting for an eventual recuperation process. For each dictionary, a metadata file containing all the information available on the dictionary (name, languages covered, creation date, size, authors, domain, etc.) is written. It is then used to evaluate the quality of the dictionary and to begin the recuperation process. Internet users only have access to these dictionaries as whole downloadable files.

3.2.2 Purgatory Directory

The *Purgatory* directory receives the lexical data once the recuperation process is over. This process consists in converting the lexical data from its original format into XML encoded in UTF-8. To perform this task, we use the RECUPDIC methodology described by Haï Doan-Nguyen (1998) with regular expression tools like Perl scripts.

If a dictionary is already encoded in XML, the recuperation process consists in mapping the elements of information into CDM (Common Dictionary Markup) elements and storing the correspondence into the metadata file.

With this information available, Internet users have access to these dictionaries as classical online dictionaries, retrieving individual entries by way of requests on the Papillon web site.

3.2.3 Paradise Directory

The *paradise* directory contains only one dictionary often called the "Papillon dictionary".

This dictionary has a particular DML (Dictionary Markup Language) structure (Mathieu Mangeot-Lerebours, 2001). Internet users have access to individual entries of this dictionary through requests to the Papillon web site. They also can add new entries or correct existing ones online.

Other *purgatory* dictionaries may be integrated into the Papillon dictionary with the help of the CDM elements.

3.3 Data Identification

The names of the files and directories are normalised in order to be able to know exactly what a file contains (with an increasing number of files and files' versions, it becomes quickly complicated to remember what makes a version relevant) and to allow people to navigate easily into the repository.

- Directories

Each dictionary is installed in a directory which name is built as follow, *DirName_la1-la2-la3/*, where:

DirName is the dictionary nickname (eg: FeM). The first letter is a capital. The remainder of the name can be either capital or small letters but always ASCII.

la1, *la2*, *la3*, etc. are the languages present in the dictionary either source or target language. The languages have to be alphabetically ordered. The 3 letter language codes are used for the name of the languages. They are written in small letters eg: *FeM_eng-fra-msa/*;

- Files

In the directories, each name of the file containing lexical data is written as follow: *dictName_sla_tll_tl2-encoding-version.ext*
where:

dictName is the nickname of the dictionary in small letters (eg: fem).

sla is the source language of the volume in 3 letter language code and small letters.

tl1, *tl2* are the target languages of the volume written in alphabetical order in 3 letter language code and small letters.

encoding is the encoding type of the file (ISO-8859-[1-9], ASCII, UTF-8, SJIS, etc). If the file is encoded in XML, it is not necessary to indicate it. Furthermore, the default encoding is ISO-8859-1. Mime type encoding names are used[3].

version is the version number of the file. This number is optional.

ext is the file extension (txt, xml, html, etc.).

For example, the version 1 of a volume of the dictionary FeM with French as source language, English and Malay as target language, encoded in mac roman in text format would be named: *FeM_en-fr-ml/fem_en_fr-macroman-v1.txt*;

- Languages codes

We use the 3 letter language codes of the standard ISO 639/2 T (eg: eng for English, fra for French, jpn for Japanese, etc). We also added new language codes for our own purpose: unl for Universal Network Language and axie for Interlingual Acception. A complete list of 3 letter language codes is available online at the same URL.

3.4 Data Structure

This major innovative structure of the lexical data repository presented in the previous part can be used for any kind of dictionary. Indeed, the precise classification into a logical organization allows building reliable and coherent multilingual lexical databases.

In order to handle heterogeneous structures with the same tools, we defined a subset of DML elements and attributes that are used to identify which part of the different structures represent the same lexical information. This is CDM. This set is in constant evolution. If the same kind of information is found in several dictionaries then a new element representing this piece of information is added to the CDM set. It allows tools to have access to common information in heterogeneous dictionaries by using pointers into the structures of the dictionaries.

[3] http://www-clips.imag.fr/geta/services/dml/dml.xsd

4 Recuperation Process

4.1 Preamble

The aim of Papillon as an open source project is of course to make users (who are able to) become contributors. But as mentioned before, the construction of good quality lexical data requires an incredible amount of work.

Since many digital dictionaries already exist, mono- or bilinguals ones, it is obvious that reusing them is very interesting. Unfortunately, their structure does not allow adding easily new languages.

The part tries to describe generic methods to successfully and efficiently manage data recuperation processes.

4.2 Problems

It became almost immediately obvious that the recuperation process had to be done with a very strong methodology. In fact, each file requires its own treatment. In spite of that, the various operations that are described below are valid for each dictionary but the way to apply them depends on each file. That is why the scripts that are written to modify the data structure must be respectful of the procedure and well documented. That also allows to other people to reuse and to adapt scripts in order to facilitate and to improve their work on other files. Another important issue is to be able to remake well-structured files from original ones in case of bad manipulation.

4.3 Methodology

A recuperation process consists in recuperating lexical data from original format to XML. It also consists in moving the new files from *Hell* directory to *Purgatory* directory.

This process can be very complicated. It was very deeply analyzed in Hai Doan Nguyen's Ph.D. Thesis. There is an existing tool called Hgrammar implemented in LISP. It is a powerful compiler that recuperates lexical data. However, errors remain and the code is not fully valid.

It is possible to recuperate existing lexical data with a language implementing regular expressions. We frequently use perl for this purpose. For each main step, a perl script is used. They are named like this: *StepNumber-StepName.pl*. *StepNumber* is a

number on two digits. The first step is 01. *StepName* is a brief explanation of the step. For example, the first Perl script frequently used is *01-convxmlchars.pl*. The order of the steps is important because it can be used afterwards to re-recuperate a dictionary or to modify the recuperation process.

If the character encoding of the file is not UTF-8, it can be converted using UNIX/LINUX tools like *uniconv*.

During the recuperation process, one important rule has to be observed; people manipulating lexical data in order to modify their structure need to keep the following postulate in mind: always keep all the information present in the original file even if it does not seem useful at a first glance (eg typographical marks, etc).

Finally, when the recuperation process is done, it is vital to ensure the update on the database in a proper way; what we use and recommend to use is cvs with the following procedure: make a cvs add and cvs commit for the new files.

4.4 Case study

We already worked on various existing dictionaries we were allowed to use. Most of the time, those data are contained into text type files, sometimes with several encoding in one file. We describe here guidelines of the recuperation process that are valid for any kind of text file. The aim of that work is to build a generic methodology; then, almost all the operations can be automatically solved with perl scripts[4].

First two steps have to be done if the file is a "*.doc*" one:

- conversion of the XML specific characters:
« < » by « < »
« > » by « > »
« & » by « & »;

- bold and italic characters:
The search of those characters can be easily done using wildcards. Then, they have to be inserted between
« » and « » if bold
« » and « » if bold;

then the .doc file must be saved as a .txt file;

- clean the overlapping of the tags and ;

- replace the characters that present an encoding problem. It is important to point out that this step requires language knowledge. In fact, only people knowing French are able to see if diacritical characters are well represented or not. Then, since a wrong character is identified, it can be automatically replaced by the correct UTF8-encoded character in the whole file. It is important to say that UTF8 does not allow to directly represent all characters. We had problem with a Vietnamese character. In that kind of case, it is possible to build the character with a combination by using existing character codes[5];

- delete empty lines;

- add the header and footer that contain relevant information about the file;

- add the XML tags according to the schema that defines the data structure. This step is quite critical. In fact, most of time or more precisely almost always, the lexical structure of the dictionaries is far from the Papillon schema. We copy the fields that are coherent with the Papillon schema and copy others into "more-info" tags;

- delete spaces besides tags;

- save the file as "*.xml*";

- Here appears a step that needs some observation from the people running those operations. Indeed, the dictionaries always contain some structural errors (even according to there own structure) that create some parsing errors. That is why it is very important to parse the file to check what kinds of error occur. Those errors are most of the time due to missing fields in the original file. The parsing operation can be processed with XML Spy or Netscape 6. The correction of those errors can automatically be done but it implies to understand what causes the errors.

Then, an XML file should be added into the *Purgatory* directory…

[4] NB as said before, the steps order must be respected; mixing the steps could modify the lexical content.

[5] http://www.unicode.org/unicode/faq/char_combmark.html

5 Conclusion

According to its motivations, goals and methodology, the Papillon project is acting in the state of the art within the domain of multilingual lexical database.

As the project progresses, it is confronted with many kinds of problems related to linguistic and computing. Of course, most of those problems are due to NLP complexity. This is why the Papillon project represents a very interesting platform in order to experiment NLP researches in optimal conditions.

We must point out that the benefits of that project will go to the community that will contribute and use it. One very important point in the close future will be to make it attractive to its recipients.

Acknowledgement

We would like to thank NII, GETA, and Kasetsart University/NAIST for their support and help in the Papillon project.

References

Gilles Sérasset. 1994, *Interlingual Lexical Organization for Multilingual Lexical Databases in NADIA*, COLING'94, 5-9 August 1994, Kyoto, Japan, pp. 278-282

Haï Doan-Nguyen. 1998, *Accumulation of Lexical Sets: Acquisition of Dictionary Resources and Production of New Lexical Sets*, 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, Proc. COLING-ACL'98, vol 1/1, Montréal, Canada, 10-14 August 1998, pp 330-335.

Alain Polguère. 2000, *Towards a theoretically-motivated general public dictionary of semantic derivations and collocations for French*, Proceedings of EURALEX'2000, Stuttgart, pp 517-527.

Jérôme Godard, Mathieu Mangeot, Frédéric Andrès. 2001, *Production of Multilingual Lexical Databases*, 1st workshop on Mlabnet, Mlabnet01, 10-12 October 2001, Karuizawa, Japan

Jim W. Breen. 2001, *A WWW Dictionary and Word Translator: Threat or Aid to Language Acquisition?*, in R Gitsaki-Taylor and P Lewis (eds), Proceedings of the 6th Annual International Conference of the Japan Association for Language Teaching Computer-Assisted Language Learning Special Interest Group (JALT-CALL 2001), Gunma, Japan, 26-27 May 2001, 10 pp.

Mathieu Mangeot-Lerebours. 2001, *Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue*, Thèse de nouveau doctorat, Spécialité Informatique, Université Joseph Fourier Grenoble I, 27 September 2001, 280 p.

Vutichai Ampornaramveth, Akiko Aizawa, Keizo Oyama & Tanasee Methapisit. 2000, *An Internet-Based Collaborative Dictionary Development Project: SAIKAM*, First International Symposium on Advanced Informatics, Proc. AdInfo 2000, 9-10 March 2000, NACSIS, Tokyo, Japan, 4 p.