Structured lexical data: how to make them widely available, useful and reasonably protected? A practical example with a trilingual dictionary

Mathieu Lafourcade - UTMK-USM - 11800 Penang - Malaysia / GETA-CLIPS-IMAG - 38041 Grenoble - France –
mathieu.lafourcade@imag.fr or lafourca@cs.usm.my

Abstract

We are studying under which constraints structured lexical data can be made, at the same time, widely available to the general public (freely or not), electronically supported, published and reasonably protected from piracy? A three facet approach – with dictionary tools, web servers and e-mail servers – seems to be effective. We illustrate our views with Alex, a generic dictionary tool, which is used with a French-English-Malay dictionary. The very distinction between output, logical and coding formats is made. Storage is based on the latter and output formats are dynamically generated on the fly at request times – making the tool usable in many configurations. Keeping the data structured is necessary to make them usable also by automated processes and to allow dynamic filtering.

Introduction

In the framework of the development of a French-English-Malay Dictionary (FEM – the producing methodology of which has been discussed in [2]), we try to address the question of making the produced lexical data widely available to the public. Although, the first goal of the project is to publish a paper dictionary (due summer 96), it appears that some other forms of distribution and presentation could be valuable. As the dictionary contractors want to keep their work proprietary while expecting a large diffusion, we have to cope with the following dilemma: how can we, at the same time, widely distribute and reasonably protect structured lexical data?

The analysis and implementation results presented here are twofold. Firstly, we identified three distribution modes that could be implemented simultaneously at a low cost. They take the form of a resident dictionary tool, a world-wide-web (or web for short) service and an e-mail service. Secondly, the problem of how to structure and

represent entries had to be tackled to keep the manipulation convenient (reduced data size, readability, version management, etc.). The proposed solution is based on a strong distinction between encoding, logical and formatting levels.

This paper is organized as follows. First, we present the objectives and constraints we identified regarding the outcome of the linguistic production of the FEM project. Then, we present three distribution models that could respond to what we identified as the needs and desires of end-users but also of the computational linguistics community. The third and last part explains our methodology and the kind of format retained to make our models a reality. We actually implemented and experimented the solutions we propose.

Constraints and desires

Beside its printed declination, we investigated some other distribution and exploitation means for the FEM. The advent of the Internet seems to offer some good opportunities for making our data widely available, but concerns have been expressed on several points: usefulness, protection and production cost.

Making data available is meaningless if they are not in a useful format. Most of the time, designing a good format and converting the data to it, is an unanticipated expenditure. The question of copyright is also an obstacle that arises much before the purely technical difficulties (see [7] for that question).

The visual appearance (opposed to the conveyed informative contents) of the data may be crucial for making them palatable to the general public. The question is in fact not only to make the data available but mainly to make people willingly use it. For these

reasons, we think the data layout proposed to the end-user is one of the main factors of success or failure of such an enterprise. But it is very difficult to forecast which kind of formatting could be "felt" by end-users as exploitable. It may depend on the task undergone, on established standards or tools available, on the user intentions, culture, etc. A presentation close to what can be found in a paper dictionary might be desirable but it can become intricate with complex data. Visual clues can help locate information (see [3]); this becomes especially critical with multilingual dictionaries. For automated processes, an explicit tagged format is more appropriate.

In fact, we would like to freely "give access" to the dictionary without "giving up" the control over its source. The legal context can be covered by copyrights, but some technical adjustments are still needed to give real protection to such a work. The dictionary should not be accessible as a whole, but merely through requests concerning one (or several) entry. Even if one entry has links to the next or previous ones as parts of its information, fetching the complete dictionary will definitely prove a painful task (as difficult as to scanning a paper dictionary). This scheme is not foolproof to hackers, but it is inconvenient enough to rebuke most users.

In an academic context, making data freely available is viable only through low cost solutions. We have to make the distinction between costs for producer (the academics and/or the researchers and linguists) and costs for the end-user. The process of formatting the data for end-users should be fast, painless and not resource demanding. Similarly, the user will not make use of (or even fetch) the data, if that gobbles up the resources of his/her own personal computer (disk space, memory, or network access time). While free of charge, the acceptance of the dictionary will be greatly improved if it is easy to manipulate. The main relevant factor is a good ratio between

compactness of the data and length of the processing time.

Three distribution models and a common tool

It is possible to distribute data in an encrypted (protected) form by distributing a free "reader". The data are located on the user computer and a dictionary tool (the reader) allows browsing among distributed dictionaries. The user can create and modify personal dictionaries, handle multiple dictionaries, copy and paste the displayed information in other applications, etc. We implemented such a tool – called Alex.

The FEM dictionary has been made accessible on the Web. The main advantages over resident tools are the transparent updates of the dictionary contents and the reduced resources needed on a local personal computer. However, one has to own an Internet connection. Moreover, the hypertext nature of Web pages can be the occasion to offer some extended features compared to paper dictionaries (which are similar to the one found in resident dictionary tools), among which access to previous or next entries, dynamic filtering and look up by patterns.

The Web approach is well adapted to end-users but (1) people having a Web access are still a minority compared with people having an e-mail account, and (2) we also would like to make our dictionary useful to automated processes. For example, e-mail access to large linguistic resources can allow regular update requests of small local linguistic databases. If the task does not require real time, communication by e-mail presents many advantages. The mail request format – which should stick to one (or several) format – can define the nature of information looked for much more precisely than what an end-user would accept to specify).

Alex is a simple dictionary tool with two main features – (1) a high level of scriptability (declined on MacOS with AppleScript) and (2) built-in extension facilities – allowing to make it the core of Web and e-mail servers. As handling several

versions of the database or pre-transcribing its contents into several formats are not viable solutions for implementation or exploitation, Alex is used as a unique engine, which operates on a unique database (one per dictionary) and produces multiple representations.

Coding format vs. Logical format vs. Output format

We have designed a mechanism that permits to produce on the fly any kind of output (or external) formats from a logical format. The chosen format is at the same time compact and adequate for fast processing.

As coding directly the logical format was too space costly for our purposes, we defined a coding (or internal) format in which the data are actually stored. Processing a request for an entry is then executed in three steps: retrieving the entry, translating the coding format into the logical format, and translating the logical format into one output format.

The logical format for one entry has been intentionally made simple. An entry kind indicator (symbol), is followed by an open list of field names (symbols) and values (strings) pairs: $(n_i, v_i)^*$. The ordering of the pairs in the list is relevant and several pairs with the same symbol can be contiguous. For example, the logical format for the entry "aimer" (love) is given below.

```
(:fem-entry (:entry
"aimer") (:Pronunciation_French "/E-ME-
/") (:French_Category "v.tr.")
(:English_Equivalent "like")
(:Malay_Equivalent
"menyukai") (:Malay_Equivalent
"menyayangi")
(:Gloss_In_French "(apprécier)")
(:English_Equivalent "like")
(:Malay_Equivalent "menggemari")
(:Malay_Equivalent "menyenangi")
(:Malay_Equivalent
"menyukai") (:Gloss_In_French
"(d'amour)") (:English_Equivalent "love")
(:Malay_Equivalent "mencintai") ...)
```

Figure 1. Part of logical format for "aimer".

In fact, the choice of the exact kind of the logical format is somewhat arbitrary as long as we keep the structure of the entry. The point to keep in mind is that the adequacy of

the format depends on the kind of processing intended. The one we adopted fits reasonably well for most of the processes we are dealing with. But sometimes small details can have a big impact on processing costs. For example, the fact that we do not factorize a sequence of several pairs with the same field name, $(n, v_1)(n, v_2)...$ as a list composed of the field name followed by the values, $(n, v_1, v_2, ...)$ is relevant. The first solution is slightly less efficient in space, but systematically dealing with pairs leads to a major performance gain in formatting.

We designed and developed a set of useful output formats with their respective producing procedures – all of them are string-based. Some are HTML strings (for Web based requests), others are labeled formats for e-mail based requests. Generally, an output format loses some of the explicit structure of the entry. An example of formatting for the entry "aimer" is given below (actually it is an RTF format – but we "interpreted" it for readability).

aimer /cmcl/, vt menyukai, menyayangi;
(apprécier) menyanangi, menyanangi, menyukai;
(d'amour) mencintai, mengasihi ; ~ **bien** suka juga;
 ~ **mieux** lebih suka; *j'aime mieux lire que regarder*
la télévision, saya lebih suka membaca
 daripada menonton television; ~ **autant** suka lagi;
 j'~**aisque** saya ingin sekiranya.

Figure 2. Formatting of the entry "aimer" as it appears on the paper dictionary (French-Malay only, the English information has been filtered out)

When Alex is used as a standalone dictionary tool, the format presented to the user is similar to the paper dictionary. The fact that we have a full control over the displaying allows us, for example, to investigate the usage of some anti-aliased fonts and softly tinted background for an increased on-line readability. The filtering functions and some aspects of the formatting are customizable by the user.

The approach we have taken for our trilingual dictionary for the Web is to include visual clues to help the user locate the information. Diamond shapes of different colors are referring to different languages (like ♦ and ♦), thus making an extension to

other languages, without losing coherence, relatively easy. Also, the filtered outputs seem to be more intuitive to the user.

The multiple e-mail formats cannot take advantage of styled text or pictures and thus have been made more explicit (and more verbose) by the use of tags. An e-mail request can specify the kind of formatting desired and generally offers a finer tuning than the two solutions above mentioned. We consider, however, that e-mail based requests are primarily aimed at automated processes.

The actual coding in which each dictionary entry is stored has been designed to be as compact as possible while allowing a fast decoding (generation of the logical format). The format can be described as containing a structural part and a textual part. In the structural part, an entry is coded as a vector. This vector does not contain any text but (1) an identifier indicating the field kind and (2) indexes to the textual part. The textual part is a buffer containing the dictionary strings. Basically, when an entry is added each field value is cut into words, which are stored in the buffer in exchange of a location (where the strings begins in the buffer) and a length (allowing to compute where it ends). Such collections of location and length constitute the indexes kept as vectors. Now words are stored twice, and a reverse alphanumeric sort increases the probability of factorization by prefix.

For example, in a first mockup of our French-English-Malay dictionary containing over 8000 entries (about 25% of the whole), the size of the structural part is about 3200 Ko and that of the buffer part is around 450 Ko. These figures are comparable to the size of the dictionary on a plain text file format.

Advantages and drawbacks of multiple formats

The first obvious gain of our solution is the reduction in the space needed for coding our dictionary. Compared to producing in advance several formats – a solution not only painful and error prone but which would also have clobbered the server

resources – a multi-server (Web and e-mail) reduced to one engine and one database per dictionary allows us to save enough resources to handle several dictionaries at the same time. Another very important aspect is the avoidance of the often nightmarish problem of synchronizing several versions of the data.

Filtering is a feature that is naturally derived from the conversion of the structure. Especially with multilingual dictionaries, it is to be expected that users will want to have access to more or less information according to their needs. This flexibility is implemented through our dictionary tool, both on the Web and by e-mail.

Generating output formats on the fly is time consuming compared to retrieving pre-formatted data. But, this is a marginal loss if we consider that the resources, effort and time devoted to the implementation of a new format can be drastically reduced.

Implementation, availability and future work

Alex has been implemented with Macintosh Common Lisp ([1] and [9]) the top of our Dictionary Object Protocol, DOP [5], itself built using a persistent object-oriented database, WOOD [8]. A more detailed account on the architecture and implementation of Alex and its derivations can be found in [4]. Prototype versions are already freely available on an experimental basis.

We are investigating how to actually make a Malay thesaurus based on the same criteria available. The formatting would include references and back-references. We also are looking for dictionaries dealing with more than three languages (adding Thai to our current French-English-Malay, for instance) and some work has already been undertaken with the Arabic transcription of Malay (Jawi).

Conclusion

Once a long term and costly project has produced a large amount of lexical data, it often runs into the questions of making its results available, usable and protected. More

often than not, they remain unused and forgotten. We presented some practical solutions for making multilingual dictionaries (in particular) and lexical data (in general) widely available, reasonably protected from piracy and useful both to the general public and to applications. We have actually implemented our solutions and made several prototypes available through a Web server and an e-mail server.

The solution we presented here is based on a common engine - Alex -, one unique database per dictionary and several formats. A logical format is used as "pivot" between a coding format and several output formats. It has been kept as simple as possible to be both easily understood and efficient for on the dynamic generation of "external representations". The coding format is used for the actual storage and has been designed to be compact enough for fast retrieval but also for efficient transcription into the logical format.

We hope that the framework of this work can inspire some other projects and help reducing the number of lexical treasures that remain unknown and unreachable both to the general public and the (computational) linguistics community.

Acknowledgments

My gratefulness goes to the staff of the UTMK and USM, the Dewan Bahasa dan Pustaka and the French Embassy at Kuala Lumpur. I do not forget the staff of the GETA-CLIPS-IMAG laboratory for supporting this project and the reviewers of this paper, namely H. Blanchon, Ch. Boitet, J. Gaschler and G. Sérasset. Of course, all errors remain mine.

References

- [1] **Digitool Inc., A. C. &. (1989-1995)** *Macintosh Common Lisp*. 3.0.
- [2] **Gaschler, J. and M. Lafourcade (1994)** *Manipulating human-oriented dictionaries with very simple tools*. Proc. COLING-94, August 5-9 1994, Makoto Nagao & ICCL, vol. 1/2, pp 283-286.
- [3] **Kahn, P. (1995)** *Visual Clues for Local and Global Coherence in the WWW*. **38/8**, pp. 67-69.

- [4] **Lafourcade, M. (1995)** *Alex 1.0 - A Generic and Scriptable Dictionary Tool*. Rapport Final, GETA-UJF, septembre 1995, 35p
- [5] **Lafourcade, M. and G. Sérasset (1993)** *DOP (Dictionary Object Protocol)*. GETA-IMAG, Grenoble, Common Lisp Object System (MCL - CLOS), Apple Macintosh, version 2.0.
- [6] **Manfred Thüring, Jörg Hanneman and J. M. Haake (1995)** *Hypermedia and Cognition: Designing for Comprehension*. **38/8**, pp. 57-66.
- [7] **Samuelson, P. (1995)** *Copyright and Digital Libraries*. **38/4**, pp. 15-21.
- [8] **St Clair, B. (1991)** *WOOD: a Persistent Object Database for MCL*. Apple, Available in MCL CD-ROM & FTP (cambridge.apple.com).
- [9] **Steele, G. L., Jr. (1990)** *COMMON LISP. The Language*. Digital Press, 1030 p.