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

Mathieu Lafourcade - UTMK-USM - 11800Penang - 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 bemade, at the same time, widely available to the general public (freely ornot), electronically supported, published and reasonably protected frompiracy? A three facet approach– with dictionary tools, web servers and e-mail servers –seems to be effective. We illustrate our views with Alex, a genericdictionary tool, which is used with a French-English-Malay dictionary. Thevery distinction between output, logical and coding formats is made. Storage is based onthe latter and output formats are dynamically generated on the fly atrequest times – making the tool usable in many configurations. Keeping the data structuredis necessary to make them usable also by automated processes and to allowdynamic filtering.

Introduction

In the framework of the development of a=46rench-English-Malay Dictionary (FEM – the producing methodology ofwhich has been discussed in [2]), we try to address thequestion of making the produced lexical data widely available to thepublic. Although, the first goal of the project is to publish a paperdictionary (due summer 96), it appears that some other forms ofdistribution and presentation could be valuable. As the dictionary contractors want to keep their work proprietary whileexpecting a large diffusion, we have to cope with the following dilemma:how can we, at the same time, widely distribute and reasonably pro-tectstructured lexical data?

The analysis and implementation results presented here are twofold. Firstly, we identifiedthree distribution modes that could be implemented simultaneously at a lowcost. 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 representententries had to be tackled to keep the manipulation convenient (reduced datasize, readability, version management, etc.). The proposed solution isbased on a strong distinction betweencoding, logical and formatting levels.

This paper is organized as follows. First, we present the objectives andconstraints we identified regarding the outcome of the linguisticproduction of the FEM project. Then, we present three distribution modelsthat could respond to what we identified as the needs and desires of end-users but also of the computationallinguistics community. The third and last part explains our methodology andthe 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 andexploitation means for the FEM. The advent of the Internet seems to offersome good opportunities for making our data widely available, but concernshave been expressed on severalpoints: 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 conver-ting the data to it,is an unanticipated expen-diture. The question of copyright is also an obstacle that arises muchbefore the purely technical difficulties (see [7] for that question).

The visual appearance (opposed to the conveyed informative contents) of thedata may be crucial for making them palatable to the general public. Thequestion isin fact not only to make the data available but mainly to makepeople willingly use it. For these reasons, we think the data layout proposed to the end-user is oneof the main factors of success or failure of such an enterprise. But it isvery difficult to forecast which kind of formatting could be "felt" byend-users as exploitable. It may depend on the task undergone, onestablished standards or tools available, on the user intentions, culture, etc. A presentation close to what can befound in a paper dictionary might be desirable but it can become intricatewith complex data. Visual clues can help locate information (see [3]);this becomes especially critical with multilingual dictionaries. For automated processes, anexplicit 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 coveredby copyrights, but some technical adjustments are still needed to give realprotection to such a work. The dictionary should not be accessible as a whole, but merely through requestsconcerning one (or several) entry. Even if one entry has links to thenext or previous ones as parts of its information, fetching the completedictionary will definitely prove a painful task (as difficult as to scanning apaper dictionary). This scheme is not foolproof to hackers, but it isinconvenient enough to rebuke most users.

In an academic context, making data freely available is viable only throughlow cost solutions. We have to make the distinction between costs forproducer (the academics and/or the researchers and linguists) and costs forthe end-user. The process of formatting the data for end-users should be fast, painless and not resourcedemanding. Similarly, the user will not make use of (or even fetch) thedata, 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 begreatly improved if it is easy to manipulate. The main relevant factor is agood ratio between compactness of the data and length of the processingtime.

Three distribution modelsand a common tool

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

The FEM dictionary has been made accessible on the Web. The main advantagesover resident tools are the transparent updates of the dictionary contentsand 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 tothe one found in resident dictionary tools), among which access to previousor 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 withpeople having an e-mail account, and (2) we also would like to make ourdictionary useful to automated processes. For example, e-mail access tolarge linguistic resources can allow regular update requests of small local linguistic databases. If the task doesnot 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 preciselythan what an end-user would accept to specify).

Alex is a simple dictionary tool with two main features – (1) a highlevel 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 handlingseveral versions of the database or pre-transcribing its contents into several formats are notviable solutions for implementation or exploitation, Alex is used as aunique engine, which operates on a unique database (one per dictionary) and produces multiplerepresentations.

Coding format vs. Logicalformat vs. Output format

We have designed a mechanism that permits toproduce on the fly any kind of output (or external) formats from a logicalformat. The choosen format is at the same time compact and adequate forfast processing.

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

The logical format for one entry has been intentionally made simple. Anentry 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 pairswith the same symbol can be contiguous. For example, the logical format forthe 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 oflogical format for "aimer".

In fact, the choice of the exact kind of the logical format is somewhatarbitrary as long as we keep the structure of the entry. The point to keepin mind is that the adequacy of the format depends on the kind ofprocessing intended. The one we adopted fits reasonably well for most of the processes we are dealing with. Butsometimes small details can have a big impact on processing costs. Forexample, the fact that we do not factorize a sequence of several pairs withthe same field name, $(n, v_1)(n, v_2)\ldots$ as a list composed of the field name followed bythe values, $(n, v_1, v_2, \ldots)$ is relevant. The first solution is slightly less efficient in space, butsystematically dealing with pairs leads to a major performance gain informatting.

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

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

Figure 2. Formating ofthe 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 paperdictionary. The fact that we have a full control over the displaying allowsus, for example, to investigate the usage of some anti-aliased fonts and softly tainted background for an increased on-line readability. Thefiltering functions and some aspects of the formatting are customizable bythe user.

The approach we have taken for our trilingual dictionary for the Web is toinclude visual clues to help the user locate the information. Diamondshapes 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 moreintuitive to the user.

The multiple e-mail formats cannot take advantage of styled text orpictures and thushave been made more explicit (and more verbose) by the use of tags. Ane-mail request can specify the kind of formatting desired and generallyoffers 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 beendesigned to be as compact as possible while allowing a fast decoding(generation of the logical format). The format can be described ascontaining a structural part and a textual part. In the structural part, an entry iscoded as a vector. This vector does not contain any text but (1) anidentifier 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 valueis 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 computewhere it ends). Such collections of location and length constitute the indexes kept as vectors. Nowords are stored twice, and a reverse alphanumeric sort increases theprobability of factorization by prefix.

=46or example, in a first mockup of our French-English-Malay dictionarycontaining over 8000 entries (about 25% of the whole), the size of thestructural part is about 3200 Ko and that of the buffer part is around 450Ko. These figures are comparable to thesize of the dictionary on a plain text file format.

Advantages and drawbacks ofmultiple formats

The first obvious gain of our solution isthe reduction in the space needed for coding our dictionary. Compared toproducing in advance several formats – a solution not only painful and error prone but which would also haveclobbered the server resources – a multi-server (Web and e-mail)reduced toone engine and one database per dictionary allows us to saveenough resources to handle several dictionaries at the same time. Another very importantaspect is the avoidance of the often nightmarish problem of synchronizingseveral versions of the data.

=46iltering is a feature that is naturally derived from the conversion of the structure. Especially with multilingualdictionaries, it is to be expected that users will want to have access tomore or less information according to their needs. This flexibility isimplemented through our dictionary tool, both on the Web and by e-mail.

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

Implementation, availabilityand future work

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

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

Conclusion

Once a long term and costly project hasproduced a large amount of lexical data, it often run into the questions of making its resultsavailable, usable and protected. More often than not, they remain unusedand forgotten. We presented some practical solutions for making multilingual dictionaries (in particular) and lexical data(in general) widely available, reasonably protected from piracy and usefulboth to the general public and to applications. We have actuallyimplemented our solutions and made several prototypes available through a Web server andan e-mail server.

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

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

Acknowledgments

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

References

[1] **Digitool Inc.,A. C. &. (1989-1995)** *Macintosh Common Lisp.* 3.0.

[2] **Gaschler, J. and M.Lafourcade (1994)** *Manipulating human-oriented dictionaries withvery 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 Cohrence in the WWW.* **38**/8, pp. 67-69.

[4] **Lafourcade, M.(1995)** *Alex 1.0 – A Generic and ScriptableDictionary 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), AppleMacintosh, version 2.0.

[6] **Manfred Thüring,Jörg Hanneman and J. M. Haake (1995)** *Hypermedia andCognition: 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,Avalaible in MCL CD-ROM & FTP (cambridge.apple.com).

[9] **Steele, G. L., Jr. (1990)***COMMON LISP. The Language.* Digital Press, 1030 p.