Logic Based Methods for Terminological Assessment

Benoît Robichaud
Observatoire de linguistique Sens-Texte (OLST)
Université de Montréal
benoit.robichaud@umontreal.ca

Abstract

We present a new version of a Graphical User Interface (GUI) called *DiCoInfo Visuel*, mainly based on a graph visualization device and used for exploring and assessing lexical data found in *DiCoInfo*, a specialized e-dictionary of computing and the Internet. This new GUI version takes advantage of the fundamental nature of the *lexical network* encoded in the dictionary: it uses logic based methods from logic programming to explore relations between entries and find pieces of relevant information that may be not accessible by direct searches. The result is a more realistic and useful data coverage shown to end users.

Keywords: electronic dictionary, Graphical User Interface, data visualization, logic programming, assessment tool

1. Introduction

This paper describes the goals, architecture and usability of a new version of a Graphical User Interface (GUI) used to browse and assess encoded data through a subset of selected lexical relations found in DiCoInfo (*Dictionnaire fondamental de l'informatique et de l'Internet*), an online multilingual specialized dictionary of computing and the Internet. The GUI allows searches in the dictionary entry fields and presents results as (directed) graphs. Figure 1 bellow exemplifies the kind of data visualization one can expect to obtain with this sort of approach.

This new version is natively written in Prolog (Sterling & Shapiro 1994) and as such allows declarative and generalized search recursion through the lexical descriptions encoded in field entries of the dictionary. In particular, this new version improves the results in two major ways: first, while performing a reification of the data (Polguère 2009), it is able to show relevant lexical relationships that remained hidden in the preceding version; second, search recursion is used to compute transitive closures on selected (or all) subsets of lexical relations allowing the implementation of other search strategies and layout improvements.

2. Motivation and related work

This GUI project started with the idea that it was possible to improve the visual and communicative value of dictionary contents using a graph visualization device that would enhance and capitalize on the fundamental nature of dictionaries since, in essence, they are *networks*. This is possible firstly by displaying the links between data that appear in field entries of one record, namely the relationships (that exist otherwise) between the *synonyms*, *derivatives* and *related meanings* of a particular term (Figure 2). Secondly, by displaying the links (that exist otherwise) between entries that share particular data in some field entry, namely the relationships between the records labeling a particular

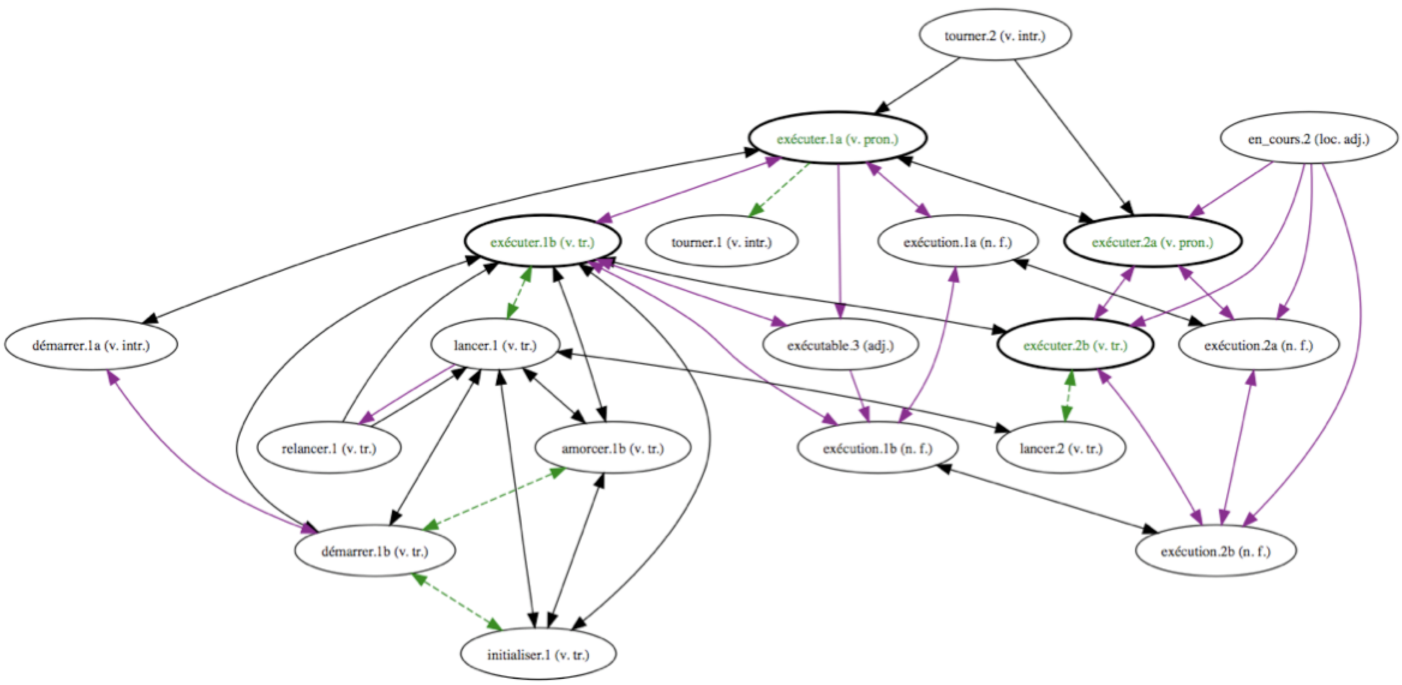

Figure 1: Some of the lexical relations of the polysemous French term *exécuter*

term as a *derivative*, an *opposite* or a *related meaning* (Figure 3).

Explanation - Typical term	Related term
Related Meanings	
≈	tourner $_{1}$
≈	démarrer $_{1a}$
≈	exécuter $_{2a}$
Other Parts of Speech and Derivatives	
Nom	exécution $_{1a}$
Qqn ou qqch. cause que le logiciel s' e.	exécuter $_{1b}$
Un logiciel qui peut être e.	exécutable $_{3}$

Figure 2: List of Related Meanings and Derivatives in the French record of *executer 1a*

Related Meanings

amorcer $_{1b}$: *exécuter* $_{1b}$ (≈)
démarrer $_{1a}$: *excuter* $_{1a}$ (≈)
démarrer $_{1b}$: *exécuter* $_{1b}$ (≈)
exécuter $_{1a}$: *exécuter* $_{2a}$ (≈)
exécuter $_{1b}$: *exécuter* $_{2b}$ (≈)
exécuter $_{2a}$: *exécuter* $_{1a}$ (≈)
exécuter $_{2b}$: *exécuter* $_{1b}$ (≈)
initialiser $_{1}$: *exécuter* $_{1b}$ (≈)
lancer $_{1}$: *exécuter* $_{1b}$ (≈)
lancer $_{2}$: *exécuter* $_{2b}$ (≈)
relancer $_{1}$: *exécuter* $_{1b}$ (≈)
tourner $_{2}$: *exécuter* $_{1a}$ (≈)
tourner $_{2}$: *exécuter* $_{2a}$ (≈)

Figure 3: List of records cataloging the French homographs *executer*

This idea is not new. During the last decades, with the spread of computer use, and the development of more ergonomic GUIs, various innovative means for searching and displaying dictionary contents have been explored and proposed. In particular, a lot of attention has been paid to the network perspective to give end users "a richer experience" (e.g. Jansz et al. 2008; Araúz et al. 2009; Thinkmap's *Visual Thesaurus* 2011; logicalOctopus's *Visuwords* 2011; Vercruysse's *WordVis* 2011) such as Figure 4.

However, it is also well known that this path is fraught with pitfalls: for example, without appropriate display or layout options of some sort, these GUIs may become rapidly complex and confusing and, as a result, users will have trouble to understand a graph with overloaded information.

With a little anticipation on that particular point, this type of interface proves, nonetheless, to be useful for at least two kinds of users: the intended forefront users (such as technical writers and translators) to whom it gives a wider angle on the relations between lexical units and enabling them to better navigate according to their search; and to the terminologists that encode the dictionary records as it allows them to quickly check the consistency between entries, thereby saving time when they review their work (details are given in section 3).

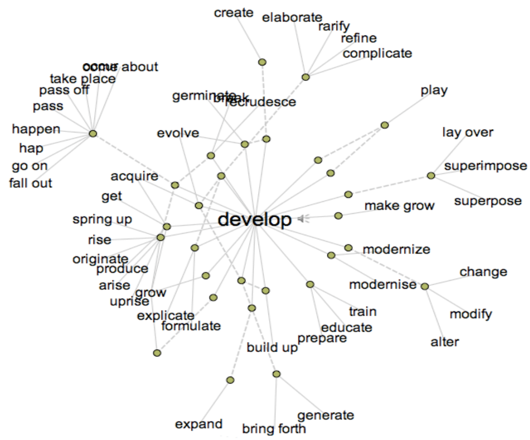

Figure 4: Lexical relations of the English word *develop* from Thinkmap's *Visual Thesaurus*

3. The *DiCoInfo* and the *DiCoInfo Visuel*

DiCoInfo was originally developed as a monolingual tool with the main function of helping end users solve specific knowledge problems associated with this specialized language. From year to year, new languages and functionalities have been added to assist them with tasks such as translation and text production in a second language (see L'Homme et al., 2009). In this section we review specific details related to its technical management and present our interactive graph visualization GUI and its new implementation using logic programming.

3.1 The *DiCoInfo* terminological database [1]

The records of *DiCoInfo* are encoded in XML files stored in an eXist database management system (Meier et al. 2011). Apart from the new graph-based interface presented in Section 2.2, end users can access and browse the terminological database through two main standard Web interfaces.

The first one is a compilation of hyperlinked HTML pages that provides the list of all records in the conventional alphabetical fashion. The second one is a searchable version that mimics a search engine and finds records containing strings (that correspond to substrings, words or terms) in specific field entries such as the usual *headword*, *variants* and *synonyms*, but also in other fields that group different sorts (or *families*) of paradigmatic and syntagmatic lexical relationships.

[1] The *DiCoInfo* can be accessed at: http://olst.ling.umontreal.ca/dicoinfo/

These relationships are formally classified and encoded by means of the *lexical functions* used in the *Explanatory Combinatorial Lexicology* framework (Mel'čuk 1996). Both GUIs are implemented using customary XSLT stylesheets that convert the original XML records and put them together in HTML format (Clark 1999).

It is worth mentioning that the subsets of lexical functions that are used in *DiCoInfo Visuel* were purposely selected for the few preliminary GUI versions. These functions encode paradigmatic relationships, namely *hypernyms*, *synonyms*, *antonyms*, *derivatives* and *related meanings*. The *hyponymic* and *meronymic* relationships, which are also interesting, are not yet incorporated since the data needs to be revised and their drawing polished. Finally, lexical functions encoding syntagmatic relationships are also ignored for the time being as another strategy for displaying them is presently being developed (Jousse et al. 2011).

3.2 The *DiCoInfo Visuel*[2]

The *DiCoInfo Visuel* is an interactive graph visualization device for browsing the *DiCoInfo* database, such as Cholz (2006), Kidd (2009) and *WordVis* (2011) for the WordNet database (Fellbaum 1998).

The *DicoInfo Visuel* project was undertaken for two main reasons:

1. We assumed that many relationships between terms were likely to be better understood if they were first presented graphically rather than immediately listed in tables with textual explanations as in an ordinary dictionary. In terminology, *taxonomies* and *meronymies* are usually presented by means of graphical hierarchies. Other relationships could also lend themselves to this kind of graphical representation.
2. We also wanted to provide the terminologists that create the entries with tools for helping them better assess the descriptions they are updating. For instance, bidirectional relationships such as *synonyms*, *antonyms*, *derivatives* and *related meanings* could be more easily checked by means of a graphical interface.

The architecture of the *DiCoInfo Visuel* is rather simple and can be described as an operational cycle carrying out the following series of tasks: (1) generation of the welcome and result HTML pages, (2) management of the search options, (3) query to the eXist database, (4) analysis and classification of the returned data, and (5) generation of the digraph description.

The originality of this visualization GUI is that it allows for the selection of one or more (families of) lexical relationships described earlier. Combined with the search precision option that offers the possibility of searching data that matches either partially or exactly the string entered, the GUI literally slices the lexical network and draws the nodes and relationships found as an intertwined directed graph such as the one shown in Figure 1. Figure 5 presents the different relationships that may be looked for during searches. The last group of options controls the different strategies that we have already implemented and which will be discussed in the next section.

All graphs presented have a 'tree' shape, as opposed to the 'spring' shape displayed in other graph-based GUIs mentioned earlier. This choice appeared to be a natural one since trees are meaningful and more appropriate for at least two subtypes of taxonomic relationships, namely *hypernyms* and *hyponyms*. The arcs of the drawn digraphs are colored according to the type of the relation they model. Dashed and dotted edges are use to distinguish the different subtypes within a particular family.

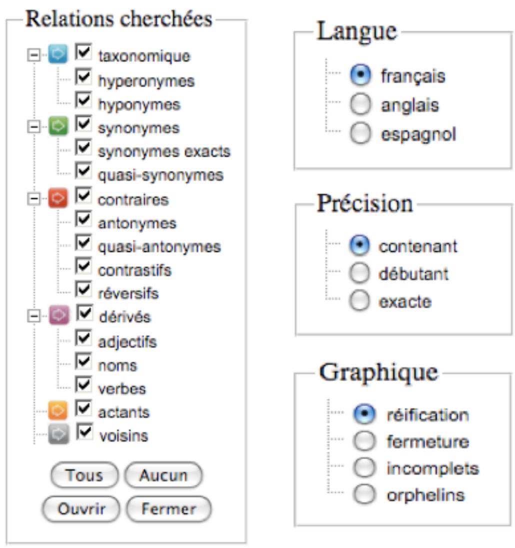

Figure 5: Search options in the *DiCoInfo Visuel*

3.3 The new Prolog version[3]

The first version was written in PHP. The new version of the GUI is all written in Prolog (with the exception of the queries to the eXist database that is still written in the XQuery language). This particular programming language was preferred for several reasons. First, as already mentioned, dictionaries are in essence networks of relational data. Logic programming seemed to be the best choice to implement a GUI that will mostly manipulate only this kind of data structures. Second, as our primary goal in this version was to develop

[2] The first version of *DiCoInfo Visuel* can be accessed at: http://olst.ling.umontreal.ca/dicoinfo/visuel.php

[3] This new Prolog version will be accessible at: http://olst.ling.umontreal.ca:5000/dicoinfo/visuel/search.cgi

computations of analogies and inferences over these relational data, we thought that it would be simpler to state the problems in a declarative fashion and then take advantage of the fact that Prolog APIs already incorporate logical inference engines, rather than having to call an external module to do them.

Other than the rewriting of the PHP programs in Prolog, new functionalities were also implemented. Firstly, the GUI now draws lexical relationships that remained hidden in the preceding version. Figure 6 shows that in the current version, the nodes that do not contain the searched expression (here *disque*, Eng. disc) may also be related to one another (here for example, *CD* and *clé USB*, Eng. pen drive) when they have a relationship that was searched for (here, *related meanings*).

The second improvement concerns the generalization of the transitive closure computation for other relations than that of *hypernymy*. In the previous version, this computation was relegated to the XQuery interpret of the eXist database server. In the current version, the main Prolog program computes transitive closures for all subtypes of relations using the ordinary recursion mechanism a better view of the lexical coverage. This new capabilities will also be exploited for implementing graph-based methods such as the search for hubs and cliques (Bang-Jensen 2009).

It is worth to mention that the GUI now uses a cache mechanism to reduce the access time to the database (downloading XML file contents on the side).

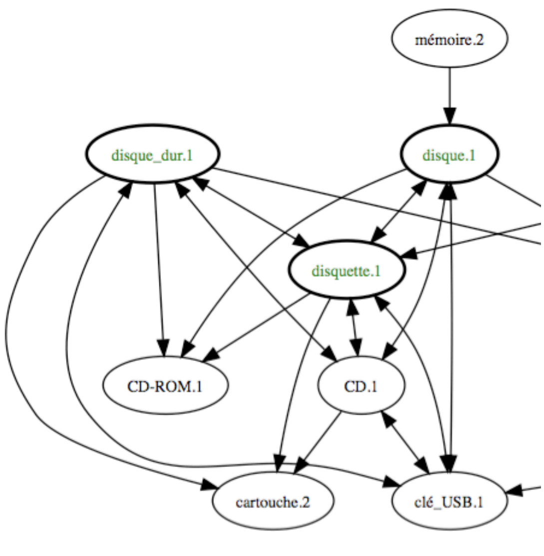

Figure 6: Relations between the participating nodes that do not contain the searched expression *CD* and *clé USB* (Eng. pen drive)

Apart from these new functionalities, two new options have also been implemented to provide support to terminologists. Firstly, as already mentioned, except for taxonomic relationships, all the others are bidirectional and their corresponding digraphs should be *complete* (Bang-Jensen 2009). When users choose the option 'incomplete', the GUI draws for these relationships only the subset of links that do not respect this constraint (Figure 7). Secondly, the GUI is also capable to search for *orphans*, i.e. nodes that match the search expression but do not share any relations with other terms (due for example to a typo or another mistake).

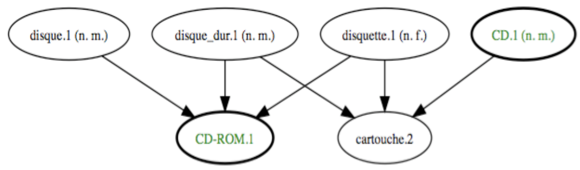

Figure 7: non bidirectional relations found when searching for 'CD'

4. Future work and conclusion

With the new gear describe here, we are ready to challenge the three drawbacks previously noted in Robichaud (2011). First, we will have to find the means to mix 'tree' shapes and 'spring' shapes when a particular node has too many direct daughters and that these span too widely on the horizontal axis of the tree. A clustering strategy base on constraints solving might be the answer. Second, some queries may simply return too many nodes linked by countless vertices and the entire graph becomes itself extremely difficult to interpret, for example, when searching for transitive closure. In this case, a mechanism could be used to alleviate the graph by identifying nodes that (under the circumstances) seem less important. The last drawback noted in Robichaud (2011) lies in the fact that the first version was simply not 'intelligent'. By using Prolog as the main program in this version, it will become easier to explore which kind of analogies and inferences (such as the ones presented in the previous section) could lend themselves to search and present data in useful new ways for end users and terminologists.

In this paper, we presented a prototypical GUI that allows a new method for organizing and visualizing the lexical relationships of dictionary contents. We also briefly presented a new implementation based on logic programming that simplifies data reification and the computation of transitive closure and searches for specific types of subgraphs over parts of a lexical network. We have high hopes that it will also allow to formalize lexical analogies and inferences.

5. References

Araúz et al. (2009). Managing inner and outer overinformation in EcoLexicon: an environmental ontology. In *8ème conférence internationale Terminologie et Intelligence Artificielle*. Toulouse

Bang-Jensen, J. (2009). *Digraphs: theory, algorithms and applications*, London: Springer-Verlag.
Clark, J. (ed.) (1999). *XSL Transformations (XSLT). W3C Recommendation*. Version 1.0http://www.w3.org/TR/xslt
Fellbaum, C. (1998). *WordNet: An electronic lexical database*. Cambridge MA: MIT Press.
Hawick, KA. (1993). *Interactive Graph Algorithm Visualization and the GraViz Prototype*. Technical report, AT&T Bell Laboratories
Jansz, K et al. (2008). *Kirrkirr: software for the exploration of indigenous language dictionaries*. http://nlp.stanford.edu/kirrkirr
Jousse, A.-L.; L'Homme, M.-C.; P. Leroyer & Robichaud B. (2011). Presenting collocates in a dictionary of computing and the Internet according to user needs. In *Proceedings of the 5th International Conference on the Meaning Text Theory*, Barcelona.
Kidd, E. (2009). Visualizing WordNet relationships as graphs. In *Random Hacks* http://www.randomhacks.net/articles/2009/12/29/visualizing-wordnet-relationships-as-graphs
Cholz, K. (2006). First Whack at JavaScript Visual Wordnet. In *kylescholz.com :: blog*
L'Homme, M.-C.; Leroyer P. (2009). Combining the semantics of collocations with situation-driven search paths in specialized dictionaries, In *Terminology,* 15(2), pp. 258-283.
L'Homme, M.-C.; Leroyer, P.; Robichaud B. (2010). Advanced Encoding for Multilingual Access in a Terminological Database – A Matter of Balance, In *Terminology and Knowledge Engineering Conference. Presenting Terminology and Knowledge Engineering Resources Online: Models and Challenges* (TKE 2010), 12-13 August, Dublin.
Meier, W. et al. (2011). *eXist : an Open Source Native XML Database*. http://exist.sourceforge.net
Mel'čuk, I. (1996). Lexical functions: A tool for the description of lexical relations in the lexicon. In Wanner, L. (ed.), *Lexical functions in lexicography and natural language processing*. Amsterdam / Philadelphia: Benjamins, pp. 37–102.
Polguère, A. (2009). Lexical systems: graph models of natural language lexicons. In *Language Resources and Evaluation*, 43, pp. 41-55.
Robichaud, B. (2011). A Graph Visualization Tool for Terminology Discovery and Assessment. In *Proceedings of the 5th International Conference on the Meaning Text Theory*, Barcelona.
Sterling, L.; Shapiro E. (1994). *The Art of Prolog*. MIT Press.
Thinkmap, Inc. (2011). *The Visual Thesaurus*. http://www.visualthesaurus.com
Vercruysse, S. (2011). *WordVis, the Visual Dictionary*. http://wordvis.com/about.html