MuLLinG: multilevel linguistic graphs for knowledge extraction

Vincent Archer
Laboratoire I3S (équipe RL), Université de Nice Sophia Antipolis
Sophia Antipolis, France
vincent.archer@unice.fr

Abstract

MuLLinG is a model for knowledge extraction (especially lexical extraction from corpora), based on multilevel graphs. Its aim is to allow large-scale data acquisition, by making it easy to realize automatically, and simple to configure by linguists with limited knowledge in computer programming. In MuLLinG, each new level represents the information in a different manner (more and more abstract). We also introduce several associated operators, written to be as generic as possible. They are independent of what nodes and edges represent, and of the task to achieve. Consequently, they allow the description of a complex extraction process as a succession of simple graph manipulations. Finally, we present an experiment of collocation extraction using MuLLinG model.

1 Introduction

Natural language processing systems often produce low-quality results, because of ambiguities and particular linguistic phenomena. One major reason is the lack of linguistic data needed to detect these phenomena or to solve ambiguities. To fill this lack, new linguistic resources should be produced. It could be done quickly with automatic processes, but quality would be unsatisfactory; on the contrary, manual work by linguists allows precise results, but takes lot of time. To get both rapidity and precision, we must combine machine and human abilities, by giving automatic processing tools to linguists, and allowing them to guide the process. Existing tools are often too centered on a task, and require too much knowledge in computer programming: they are not appropriate for linguists with few knowledge in coding. We should thus develop generic tools.

In this article, we first focus on how to make the resource gathering easier. Then, we introduce *MuLLinG*, our multilevel graph model for linguistic extraction, with several associated operations. Finally, we present an application of that model on collocation extraction.

2 Knowledge extraction

There are several manners to collect resources with automatic processes (machine learning, collaborative interfaces, etc.). We focus here on (linguistic and statistic) extraction of candidates. More precisely, our goal is to facilitate the large-scale production of candidates by extraction.

2.1 Simplify programming

Making a particular extraction task is not easy, as there is often no dedicated tool. It forces to write ad hoc tools (most of the time not unveiled). Moreover, ad hoc tools are not written to be universal. They generally depend on the data model, it is therefore difficult or impossible to use a new resource with a different format (such as an analysis from an other parser). To be really useful, an extraction tool should be *generic* (able to handle different data models) and *easy* to understand and to use. The data model on which the tool rely must be simple, expressive (complex structure should be represented easily), and universal (for monolingual or multilingual corpora, dictionaries, etc.). It should also provide simple generic, task-independent, high-level operations that can be combined to describe a complex task.

We choose to introduce a graph-based model. Graphs are understandable quickly by humans, easy to use in automatic processes, and flexible enough to represent various data types. Using graphs for knowledge extraction is quite classic. They can represent relations between words (produced by dependency analysers from corpora), and be used to produce semantically close terms (Widdows & Dorrow, 2002) or to group similar n-tuples (Hassan et al., 2006). Graphs also can be

Proceedings of the 2010 Workshop on Graph-based Methods for Natural Language Processing, ACL 2010, pages 69–73,
Uppsala, Sweden, 16 July 2010.

generated from dictionaries, and used to produce knowledge bases (Richardson et al., 1998) or proximity information (Gaume et al., 2006).

2.2 Existing graph models

Influenced by "existential graphs" (Peirce, 1931-1935) where relations between elements are represented by nodes, "conceptual graphs" (Sowa, 1976) are bipartite graphs with two node types: concepts and conceptual relations (edges only associate relations and concepts). That relation *materialization* is useful, as it allows to handle easily n-ary relations, without hypergraphs.

Another interesting network is the "lexical system" one (Polguère, 2006), defined as oriented, weighted, unhierarchical and, above all, *heterogeneous*: there is no constraint on what is modelized (it could be terms, meanings, collocations, etc.). It avoids the separation between dictionary-like and network-like lexical databases, and shows the same representation can be used for each kind of data and relation.

Finally, graphs can be *multilevel*, to represent different kinds of information. Links are generally allowed only in a same level or between two adjacent levels, like in "hypertexts" (Agosti and Crestani, 1993) made of three specified levels (documents, terms, concepts), or in Multi-Level Association Graphs (Witschel, 2007) in which there is no constraint on the number of levels. We believe that the use of several levels to represent various content types is pertinent in an extraction process, as it allows to handle both the occurrences of terms, and the terms themselves.

3 MuLLinG model

We introduce *MuLLinG* (Multi-Level Linguistic Graph), our own graph model. Divided in several ordered and distinct levels, it contains two kinds of edges: *intra-level* ones (between nodes from same level) and *inter-level* ones (from a node on level i to a node on level i+1). Intra-level edges are not unique (several edges are allowed between two nodes): every level is a multigraph. On the contrary, a node can be the source of only one inter-level edge; this association means that the target node (on the superior level) is a more global representation of the source node (it defines a hierarchy of precision).

Finally, in order to allow the heterogeneity of represented data, nodes and intra-level edges can carry any attribute (with no limit on kind or number). Figure 1 shows an example of a MuLLinG graph, in which 1st level contains occurrences of words, 2nd level contains lemmas, and 3rd level contains synonymy classes.

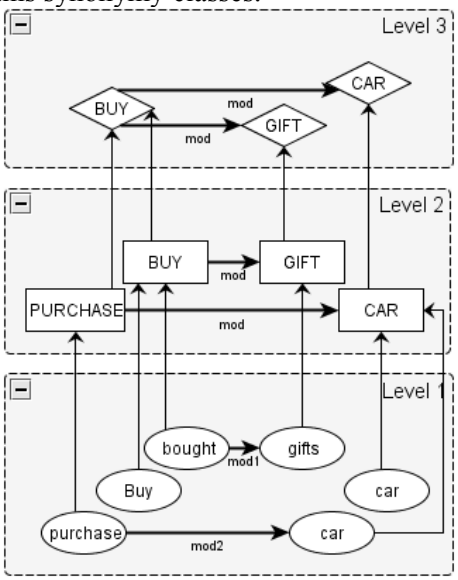

Figure 1. Example of 3-level MuLLinG graph

3.1 Definition

More precisely, a MuLLinG graph is an oriented multigraph $G^n = (V, E, F, A, \Phi, a_V, a_E)$ (for n levels) where:

- V: set of *nodes*, made of n disjoint subsets $V_1, \ldots, V_n$ (for the n levels);
- E: set of *intra-level edges*, made of n disjoint subsets $E_1, \ldots, E_n$; A: set of functions $\alpha_i : E_i \to V_i \times V_i \mid i \in \{1, \ldots, n\}$ associating an edge and its two extremities;
- F: set of *inter-level edges*, in *n-1* disjoint sets $F_1, \ldots, F_{n-1}$ defined as $F_i = \{\langle x, y \rangle \in V_i \times V_{i+1} \mid y = \varphi(x)\}$; Φ: set of functions $\varphi_i : V_i \to V_{i+1} \mid i \in \{1, \ldots, n\}$, associating a node (on a given level) and a node on the superior level);
- $a_V = \{f : V \to \Sigma_V\}$, $a_E = \{f : E \to \Sigma_E\}$ (Σ_V, Σ_E are alphabets for attributes of objects from *E* and *V*) model attributes.

3.2 Associated operators

To manipulate MuLLinG graphs, we introduce several operations, designed for their particular structure. Some of them allow elementary manipulations: add or delete a node or an edge, clean a node (delete all edges of which it is a source or a target), delete a node and its "descendants" (the nodes linked *to* it by inter-level edges, and their own descendants). There are

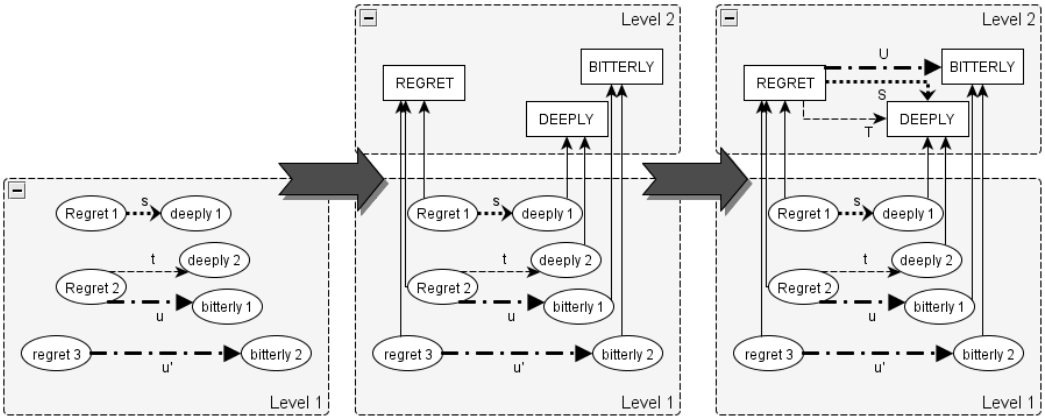

Figure 2. Two-steps emergence (nodes, then edges)

also operations to compute measures, to realize a *conditional manipulation* on nodes or edges (it can be use to *filter* the graph, by deleting nodes depending on the value of a given attribute). All these basic operations should not be directly used, but rather be called by more elaborate ones.

These operations (modifying the graph structure) take parameters fixed by the user: the level, the filtering function (which graph elements are concerned by the operation?), and computation functions (to produce attribute values for newly created elements). Graph coherence is guaranteed if the user provides correct parameters.

Emergence is the essential operation associated with MuLLinG. Its aim is to generate a superior level, by grouping elements (from the initial level) in equivalence classes. In the newly created level, each node (resp. edge) represent a equivalence class of nodes (resp. edges) from the initial level. The identification of equivalence classes is a parameter of the emergence (the user provides it). The operation goes in two steps:

- *node emergence*: for each equivalence class of nodes, it creates a node on the superior level to represent this class (and each node in the class is linked to the newly created node); figure 2 shows the emergence of nodes representing equivalence classes containing all occurrences of a same word;
- *edge emergence*: each edge added on the superior level between nodes *A* and *B* depict a set of equivalent edges between an element of *A* class and an element of *B* class; in figure 2, equivalent *u* and *u'* are grouped in a sole edge *U*, whereas *s* and *t* (not equivalent) are represented by two distinct edges *S* and *T*.

Finally, some other operations have been defined to mix information from two graphs in a third one. The *intersection* contain elements (nodes, edges) present in both graphs, with unification of identical elements. The *union* contain all elements from the two graphs, with unification of identical elements. The *difference* contain all elements from the first graph that are not identical to an element from the second one.

It is essential to recognize the identity between two nodes or two edges: *identity functions* are parameters for these "mix" operations, and should be provided by the user. Among parameters, there are also, depending on the case, functions for *fusion* (production of attributes for unified nodes or edges) or *copy* (production of attributes for elements present in only one graph).

To handle n-ary relations, we also provide a *complex* version of MuLLinG, where relations can be materialized. In that case, a relation is represented by a standard node and *numbered argument edges* linking that node to the arguments of the relation. It also allows the representation of relations between relations themselves.

We made an implementation of *MuLLinG* as a C++ library[1], based on *Boost* (open-source C++ libraries), especially for graph access and iterations. It can read and write MuLLinG graphs in GraphML format (Brandes et al., 2001).

4 Application to collocation extraction

4.1 Extraction process

We realized several experiments using our library. We remind the reader that our goal was not to obtain the more efficient method for extraction, but rather to introduce tools for simplifying the programming of extraction tasks. We present here experiments about collocation extraction. *Collocations* are particular expressions where a term is chosen arbitrarily, depending on the other

[1] Available at http://mulling.ligforge.imag.fr/ (under *CeCILL* free software license)

term, to express a particular meaning (like in "driving rain", where "driving" is used to express intensity). As the choice differs between languages[2], it causes big issues to machine translation systems (which lack resources to handle them correctly). In our experiment, the initial graph is made of relations produced by a dependency analyzer, on 1st level.

Firstly, we use the *filtering* operator to keep only pertinent relations (nouns modified by adjectives, like in figure 3, or verbs modified by adverbs), according to the analyzer. There are relations between term occurrences on 1st level, but we want relations between terms themselves: we generate them on 2nd level using emergence. So we proceed *node emergence* by considering that nodes with same attribute "lemma" are equivalent, then *edge emergence* by considering that edges expressing a modification are equivalent.

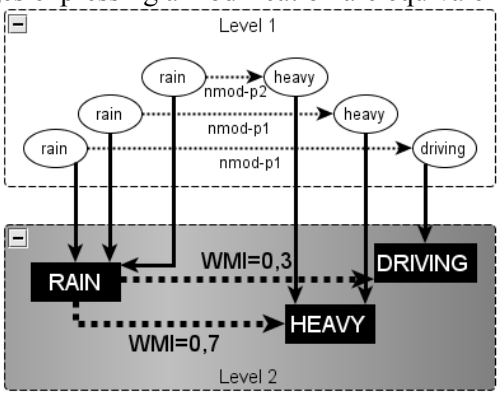

Figure 3. Collocations extraction with emergence (on 2nd level) and computation operations

The "collocation" candidates are all 2nd-level edges created during the emergence. To rank them, we use the *computation* operation (with occurrence and co-occurrence frequencies) to fix an association measure on those nodes. Figure 3 shows an example of a MuLLinG graph after emergence and computation operations.

To facilitate the description, our library contains lots of pre-defined generic functions. By example, a filter (used as a parameter of emergence) can be based on an excepted value, a threshold, etc. We also described numerous association measures; for now, new ones should be written in the C++ program.

We used our library to carry out the extraction as described previously, with LeMonde95 corpus (news articles) analyzed by Xerox's XIP parser. Thanks to MuLLinG structure, it is very easy to get all potential collocations (*heavy/driving* rain): these are the relations of which it is the source.

[2]By example, a "*heavy* smoker" is *big* in French ("*gros* fumeur") and *strong* in German ("starker Raucher").

Experiments		*verb-adverb*	*noun-adjective*
Level 1	nodes	1 155 824	1 319 474
	edges	1 780 759	2 009 051
Level 2	nodes	6 813	33 132
	edges	144 586	273 655

Table 1. Nodes and edges produced during experiments on collocation extraction

4.2 Advantages and drawbacks

With MuLLinG library, we reproduced *exactly* some experiments on collocation extraction we made before (with ad hoc programs): results are obviously coherent. The production is currently slightly slower (around 20% more time) but speed is not crucial, and could be optimized. MuLLinG has a great advantage while writing the program: it only calls functions (and declare parameters). Consequently, task description with our library is much faster (source lines of code are divided by 5), it also avoids errors. It requires less knowledge in programming, so it is far more accessible. Nevertheless, usability should still be improved: we must describe a high-level language (we believe it should be a request one). Furthermore, there is no constraint on input resources, so programs could easily be re-used with other relations (from other parsers). Finally, as graphs with millions of elements can reach RAM limits, we plan to allow database storage.

We also made bilingual experiments on collocations, taking advantage of MuLLinG complex version to materialize monolingual "collocation" nodes, and to describe bilingual relations between collocations as edges between them.

5 Conclusion

Facing the lack of tools for extraction of lexical knowledge, we looked for a new one, simple and generic. We specified MuLLinG, multilevel graph model (with no constraint on the data), associated with several simple manipulation operations (which could be combined to realize complex tasks). The ensuing tool allows to program linguistic tasks in a resource-independent manner, simpler and more efficient. One major prospect of this work concerns its implementation. As explained before, we must provide a high-level language. It is also necessary to facilitate the import and to optimize memory management. In order to provide a less NLP-centered tool, we should extend it with new operations, and with algorithms related to classic problems of graph theory. It would also be interesting to interact with semantic web tools (RDF/SPARQL).

References

Maristella Agosti and Fabio Crestani. 1993. A Methodology for the Automatic Construction of a Hypertext for Information Retrieval. In *Proceedings of 1993 ACM Symposium on Applied Computing*, 745-753.

Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt and M. Scott Marshall. 2001. GraphML Progress Report - Structural Layer Proposal. In *Proceedings of 9th International Symposium Graph Drawing (GD'01)*, 501-512.

Hany Hassan, Ahmed Hassan and Sara Noeman. 2006. Graph based semi-supervised approach for information extraction. In *Proceedings of HLT-NAACL-07 Workshop on Textgraphs-06*, 9-16.

Bruno Gaume, Karine Duvignau and Martine Vanhove. 2008. Semantic associations and confluences in paradigmatic networks. In Martine Vanhove (Ed.), *From Polysemy to Semantic Change Towards a typology of lexical semantic associations*, John Benjamins, 233-264.

Charles Sanders Peirce. 1931-1935. *Collected Papers of C. S. Peirce* (C. Hartshorne & P. Weiss, eds.), Cambridge: Harvard University Press.

Alain Polguère. 2006. Structural Properties of Lexical Systems: Monolingual and Multilingual Perspectives. In *Proceedings of Workshop on Multilingual Language Resources and Interoperability (COLING/ACL 2006)*, 50-59.

Stephen D. Richardson, William B. Dolan, and Lucy Vanderwende. 1998. MindNet: acquiring and structuring semantic information from text. In *Proceedings of COLING 1998*. 1098-1102.

John F. Sowa. 1976. Conceptual graphs for a database interface. *IBM Journal of Research and Development 20:4*, 336-357.

Dominic Widdows and Beate Dorow. 2002. A Graph Model for Unsupervised Lexical Acquisition. In *Proceedings of 19th International Conference on Computational Linguistics(COLING 2002)*. 1093-1099.

Hans Friedrich Witschel. 2007. Multi-level Association Graphs - A New Graph-Based Model for Information Retrieval. In *Proceedings of HLT-NAACL-07 Workshop on Textgraphs-07*, 9-16.