Multiscale Visual Analysis of Lexical Networks

Guillaume Artignan, Mountaz Hascoët, Mathieu Lafourcade

Multiscale Visual Analysis of Lexical Networks

Guillaume Artignan, Mountaz Hascoët, Mathieu Lafourcade
Univ. Montpellier II
LIRMM, UMR 5506 du CNRS,
161, rue Ada 34392 MONTPELLIER Cedex, France
{artignan,mountaz,lafourcade}@lirmm.fr

Abstract

A lexical network is a very useful resource for natural language processing systems. However, building high quality lexical networks is a complex task. "Jeux de mots" is a web game which aims at building a lexical network for the French language. At the time of this paper's writing, "jeux de mots" contains 164 480 lexical terms and 397 362 associations. Both lexical terms and associations are weighted with a metric that determines the importance of a given term or association. Associations between lexical terms are typed. The network grows as new games are played. The analysis of such a lexical network is challenging.

The aim of our work is to propose a multi-scale interactive visualization of the network to facilitate its analysis. Our work builds on previous work in multi-scale visualization of graphs. Our main contribution in this domain includes (1) the automatic computation of compound graphs, (2) the proximity measure used to compute compound nodes, and (3) the computation of the containment relation used to exhibit the dense relation between one important node and a set of related nodes.

Keywords—**Multiscale, Hierarchical Graph, Visualization, Hierarchical Clustering**

1 Introduction

Many natural language processing tasks like information retrieval or anaphora resolution require lexical information usually found in resources such as thesauri, ontologies, or lexical networks. Creating such resources can be done either manually in the case of Wordnet [10] for example or automatically from text corpora as in [21]. In both cases, the generation of accurate and comprehensive data over time is a complex task.

"Jeux de Mots" [17, 16] is a game where players contribute to the creation of a complex lexical network by playing. The game is a two player blind game based on agreement: at the beginning of a game session player A is given an instruction related to a target term. For example: *give any term that is related to "cat"*. User A has a limited amount of time to propose as many terms as possible. At the end of the session, player A's proposed terms are compared to those of a previous player say player B. Points are earned on the basis of agreement, e.g. the intersection of the two sets of terms proposed by A and B. The lexical network of "Jeux de Mots" is built by adding the terms in the agreement. A relation to the target term is also added. The relation between the target term and the terms agreed depends on the initial instruction. In the previous example the relation is a relation of type association. There are 35 other types of relations in "Jeux de mots" including synonymy, antonymy, hyperonymy, etc. Weights are further computed for terms and for relations between terms in order to reflect their importance in the network [15]. At the time of this paper's writing, "jeux de mots" contains 164 480 lexical terms and 397 362 associations. Therefore, the visualization of the network is challenging. JeuxDeMots lexical network can be considered as a large graph with terms as nodes and semantic relations between terms as edges.

Multiscale interactive visualization of graphs is an interesting solution to the visual analysis of large graphs. Hierarchical graphs, introduced in [9] for the first time, have largely influenced the literature in this domain. Approaches vary at different levels. Our approach is based on compound graph construction and full zoom exploration. The construction of the compound graph is further based on a proximity measure used to compute compound nodes and the computation of the containment relation used to exhibit dense relation between one important node and a set of related nodes.

This paper is organized as follows: we start by a review of related work, we further present the data and a careful analysis of some properties that matter for visualization. We further present our main contributions e.g. compound graph construction (section 4) and full-zoom exploration of JeuxDeMots lexical network (section 5).

2 Related Work

Our approach to the visual analysis of JeuxDeMots lexical network is based on previous work and mainly related to multilevel graph exploration.

Multilevel graphs are largely used in graph visualization. Indeed multilevel graph drawing methods have been studied in order to accelerate run time and also to improve the visual quality of graph drawing algorithms. In [24], Chris Walshaw presents a multilevel optimization of the Fruchterman's and Reingold's spring embedder algorithm. The GRIP algorithm [11] coarsens a graph by applying a filtration to the nodes. This filtration is based on shortest path distance. Fast Multipole Multilevel Method (FM^3) [13] is also a force-directed layout algorithm. FM^3 is proved subquadratic (more precisely in $O(NlogN+E)$) in time, contrary to previous algorithms. Work in [3] is based on the detection of topological structures in graphs. This algorithm encodes each topological structure by a metanode to construct a hierarchical graph.

Graph hierarchies are also used in Focus-based multilevel clustering. In [6, 7, 8] several hierarchical clustering techniques are proposed, to visualize large graphs. These contributions are mainly concerned with accounting for a user focus in the construction of a multi-level structure. Sometimes this results in new multi-level structures such as for example MuSi-Tree (Multilevel Silhouette Tree) in [8]. Other approaches are based on zooming strategies that include level-of-details dependant of one or more foci [12].

Multilevel graph exploration is challenging. Multilevel graph exploration systems can be divided into two categories : systems needing precomputation to create a hierarchical structure and systems which create hierarchy during the exploration. Our approach fall into the first category.

Approaches that fall in the first category take more time during the construction step but they facilitate multi-level exploration. In [9] the authors propose an algorithm for creating a graph hierarchy in three dimensions, each level is drawn on a plane. In [20] authors propose a comparison between two methods of multi-level exploration:"Fisheyes View" and "Full-zoom" methods. Work in [12] is based on a zooming technique associated with a precomputed hierarchical graph. The level of detail is computed on-the-fly and depends on the distance to one or more foci. Abello et al. [1] define a compound fisheye view based on a hierarchy graph. In addition the authors link a treemap with a graph hierarchy. In [23], the authors create a force directed layout, and use it on graphs in order to highlight clusters. This technique is similar to the approaches that merge clusters in small world vizualization. In [5] the contribution is to propose the visualisation of complex software in 3D or in 2D. Edge bundles are created in order to simplify edges. This method uses visual simplification of graphs using a level-of-detail approach.

Approaches that fall into the second category compute a hierarchical graph during the exploration step. The layout of the graph is computed on the fly. The authors of [22] present a tool, ASK-GraphView, based on clustering and interactive navigation. Hierarchical clustering is obtained by detecting biconnex components, and by a recursive call to a clustering algorithm on biconnex components. In [4] the authors propose Grouse. Grouse is based on previous work [3] and decomposes the graph based on topological features. Grouse further uses adapted layout algorithms to layout subgraphs.

3 Data

In the rest of this paper we focus on a subgraph of JeuxDeMots. The subgraph is obtained by studying only the edges that correspond to the relation of type "Associated Idea". Furthermore, we filtered nodes that were not connected to the biggest connected component. The resulting subgraph is composed of 20 238 words and 64 564 edges.

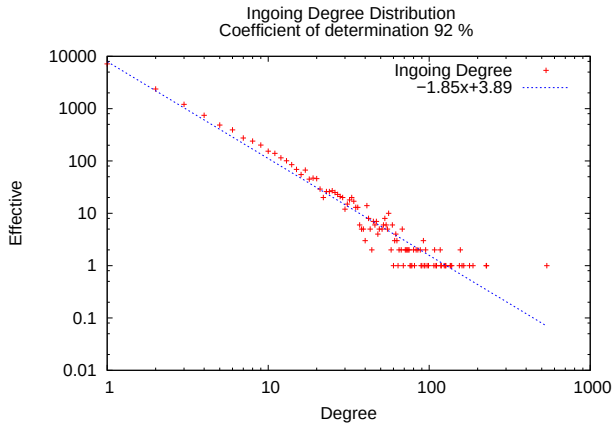

Figure 1: Degree repartition, for ingoing degree.

3.1 Data analysis

The aim of this section is to better characterise the type of graph we are working on. A study of degree repartitions (distribution of ingoing edges Fig. 1 and distribution of outgoing edges Fig. 2) is useful to show that our degree distributions have power-law tails. $\gamma_{in} = -1.85$ the indegree exponent and $\gamma_{out} = -2.27$ the outdegree exponent, are high determination coefficients [2].

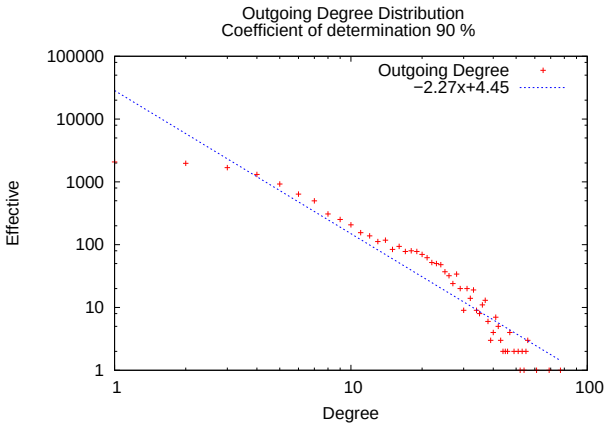

Figure 2: Degree repartition, for outgoing degree.

The graph studied is clearly a scale-free graph [2]. A second study can be made to compute the clustering coefficient [2]. The average of our graph $\overline{C} = 0.2617$, and the degree average is $\overline{D} = 6.3805$. Moreover the clustering coefficient of a random graph of the same size and average degree is $C_{rand} = 0,00032$. Our graph has an average clustering coefficient order of magnitude higher than the coefficient of clustering of a random graph with the same size and the same average degree. Furthermore, the diameter of our graph is 12. For all these reasons, our graph can be considered as a small world network.

4 Compound graph construction

In order to provide full-zoom exploration of the lexical network it is necessary to automatically compute a hierarchical graph that is coherent for an end-user of lexical networks like, for example, a searcher in natural language engineering or a lexicographer.

The originality of our approach is (1) that it is based on metrics derived from natural language engineering metrics that compute at low cost, and (2) we create a compound graph instead of a clustered graph. It is important to stress that the clustered graph approach is the most frequent one found in the litterature and that it constitutes a serious drawback when it comes to lexical network exploration as will be discussed in the section 4.2.

In order to create a compound graph, we first adapted one proximity measure used in information retrieval and natural language processing tools and we further extend it to provide a multilevel proximity measure used in the construction of the compound graph.

4.1 Proximity Measure

The "Direct Proximity Measure" is computed for an edge in a graph. This measure is useful in computing another measure the "Hierarchical Proximity Measure" which will be described in the next section. The hierarchical measure applies to two nodes n and m of a hierarchy, and accounts for the direct proximity measure of the edges linking n to m.

4.1.1 Direct Proximity Measure

The proximity measure is adapted from the measure of tf-idf (term frequency - inverse document frequency) [19]. It is computed on each edge and accounts for the weight of the edge. The weight of each edge represents a degree of confidence. This measure is defined as follows:

Let G be a graph such that $G = (V, E)$, we take an edge $e \in E$ and a node $n \in N$ and we define :

- $source(e)$ the node source of edge e, and $target(e)$ the node target of edge e.
- $\omega(e)$ the weight of edge e.
- $\delta^+(n)$ is the weighted outgoing degree of n, and $\delta^-(n)$ is the weighted ingoing degree of n.

The proximity value [18] is computed using the following formula :

$$prox(e) = \frac{\omega(e)}{\delta^-(target(e))} \times \frac{\omega(e)}{\delta^+(source(e))}$$

The first (resp. second) factor of our formula corresponds to the proportion of the weight of e in ingoing edges (resp. outgoing edges) of e. The proximity measure, can be computed for a weighted graph, oriented or not. It reflects the importance of e for its extremities. In the example Fig. 3 the importance of e is weak in comparison to the total weight of all incident edges. Consequently, the two nodes have a weak proximity as shown here $prox(e) = \frac{10}{460} \times \frac{10}{285} = 0,00072$

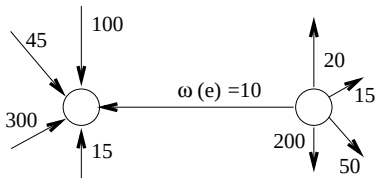

Figure 3: A sample for the proximity metric

4.1.2 Hierarchical Proximity Measure

A hierarchical proximity measurement is computed between two nodes x and y on a hierarchy of l levels with $l >= 0$ (see figure 5). This measure accounts for the edges e_i between x and nodes that are in the shortest path between x and y.

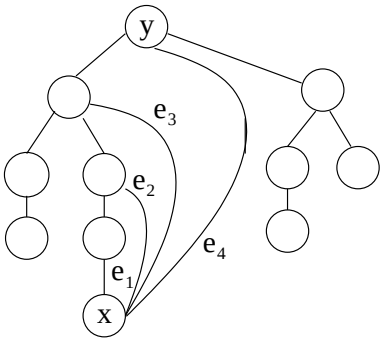

Figure 4: Hierarchical proximity measure

The measure $prox_\rho(x)$ is computed with the following formula :

$$prox_\rho(x,y) = \sum_{i=1}^{l} prox(e_i) * (l+1-i)$$

When an edge doesn't exist we will consider its weight to be equal to zero. The hierarchical measure takes parents in the multilevel graph into account and favours close parents over more distant parents.

The Fig. 5 gives an example of the computation of the hierarchical proximity measure. In this example, the computation of the hierarchical proximity measure is the following:

$$prox_\rho(x,y) = \\ prox(e_1) * 4 + prox(e_2) * 3 + prox(e_3) * 2 + prox(e_4)$$

4.2 Compound graph versus clustered graph

As mentionned previously, we chose to construct a compound graph instead of a clustered graph. The main difference between compound graphs and clustered graphs is that in the latter case meta-nodes that represent clusters are created [14]. A difficulty is then to find labels to attach to the meta-nodes created. By building a compound graph we avoid this problem since no new node has to be created. The final structure contains only the nodes of the original graph. Important nodes are used as clusters or compound nodes and the containment hierarchy can be used to express important relations between compound nodes and related nodes. This strategy offers several advantages. Firstly it underlines important nodes. Secondly, it encodes edges with high proximity measure by the containment relation of our compound graph. This graphical coding is not only stronger than simple links, it also simplifies the graphical representation by eliminating a lot of links. Thirdly, as mentionned above, there is no additionnal work to find representatives for meta-nodes, since meta-nodes are nodes, their name is directly found and meaningful.

4.3 Algorithm

In this section we present and explain our algorithm. Our algorithm Alg. 1 can be decomposed into three parts : (1) The initialisation, from line 1 to line 3, (2) the grouping of neighbours, from line 5 to line 10, and (3) the reassignement of neighbours, from line 11 to line 20.

Algorithm: Graph2GraphHierarchy(Graph G;X,Y,Z integers)

```
1  max ← getMaxDegreeNode(G,X);
2  color all nodes in max in BLACK;
3  leaves ← max ;
4  while leaves ≠ ∅ do
5      leaves2 ← get the neighbours not BLACK of
       leaves;
6      for each node n ∈ leaves2 do
7          near ← neighbours of n in leaves;
8          p ← give a node n' in near maximizing
           prox_ρ(n,root(n'));
9          child(p) ← child(p) ∪ node;
10         color node in BLACK ;
11     for each leaf ∈ leaves do
12         children ← Child(leaf);
13         selected ←
           MaxProx&DegNode(Y,Z,children);
14         for each n ∈ children \ selected do
15             near ← neighbours of n in selected;
16             node ← give a node n' in near
               maximizing prox_ρ(n,root(n'));
17             if node ≠ parent(n) then
18                 remove n from child(parent(n));
19                 child(node) ← child(node) ∪ n;
20         leave3 ← leave3 ∪ child(leaf);
21     leave ← leave3;
```

The initialisation consists in choosing X vertices with a maximum weighted degree (line 1). These vertices constitute seeds for our sub-hierarchies at the level 1. The Fig. 5 (A) describes this initialisation, here we take the nodes $\{a, b, c\}$.

The parts 2 and 3 are enclosed in a Breadth-First search algorithm. The nodes are colored in black in order to know which nodes have already been processed.

The second part of our algorithm consists in taking the neighbourhood of our seeds (line 5). Each seed will be the compound nodes of all nodes in the neighbourhood. Each seed has a different neighbourhood, nevertheless a node can be in several different neighbourhoods, see (B) in Fig. 5, the node d can be affected in two different groups.

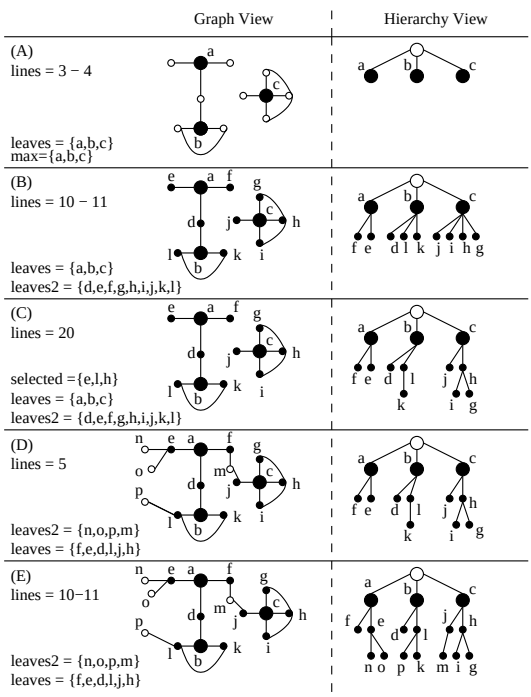

Figure 5: Execution of Graph2GraphHierarchy Algorithm

The third part of our algorithm is the reassignment. Each seed now has a list of child nodes noted $children$. We select the Y nodes in $children$ which have the highest weighted degree, and on these nodes we select the Z nodes which maximize the proximity with their parents. We obtain a list of $selected$ nodes considered as important (line 13) in the algorithm. We must further reassign all previously added nodes, to nodes in the $selected$ set of nodes if they maximize the proximity value. For example, see the Fig. 5, line (C), node i, g, k are reassigned to new parent nodes in $selected$.

We iterate with nodes contained in the next level of our hierarchy see Fig. 5 areas (D) and (E). The algorithm stops when all nodes are colored black.

The algorithm is particulary adaptable to scale-free graphs. In particular, it is possible to adapt parameter Y (number of nodes of maximal degree) and Z (number of nodes of maximal proximity) in order to favour either closer or higher degree nodes in the selected set of nodes. If the value of parameter Y is chosen in order to favour the nodes with higher degree it helps to reduce the number of links displayed (replaced by the containment relation) which in turn makes the diagram clearer.

5 Full-zoom exploration

Zoom is used to support multi-level exploration of the lexical network. It is based on the compound graph generated according to the procedure described in the previous section.

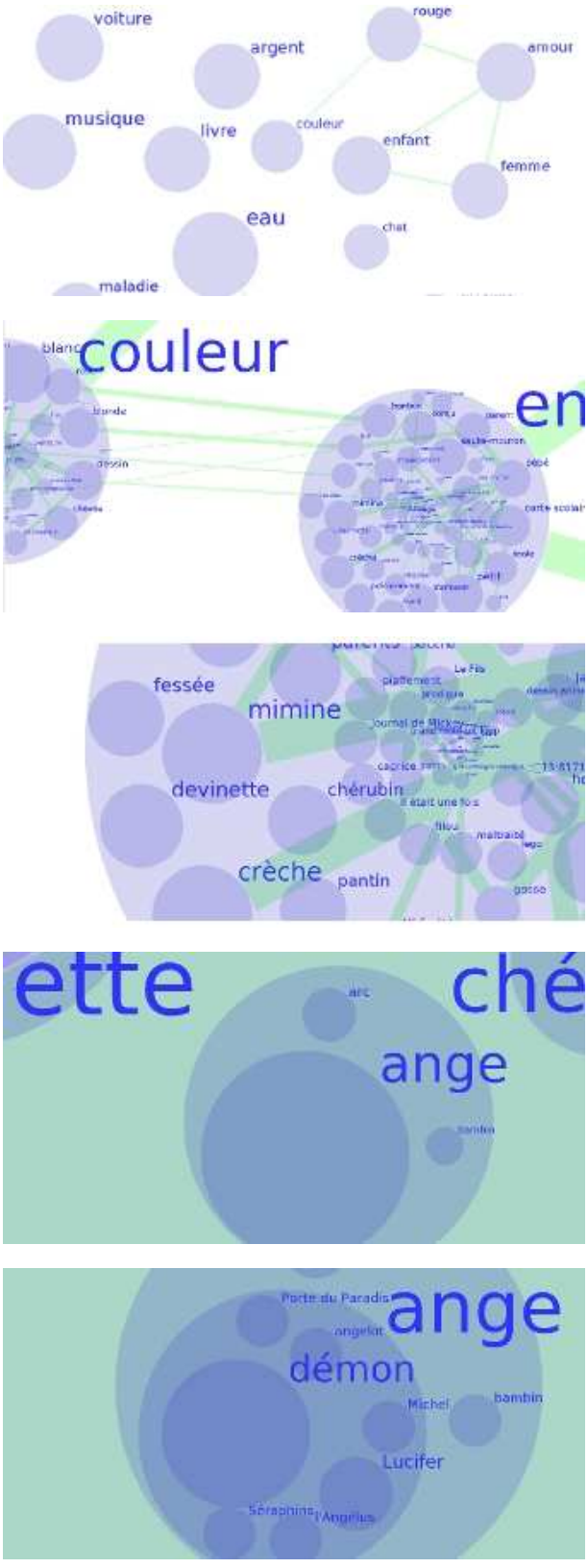

Figure 6: Full-zoom exploration

In the graphical representation, (see figure 5) nodes are represented by circles. The compound graph structure imposes that a node can contain a graph which can be empty or contains other nodes.

At each zoom level, the computation of the surface of a node is defined by :

$$surface(x) = \begin{cases} 4 * \pi^2 & \text{if } child(x) = \emptyset \\ \phi \times \sum_{c \in child(x)} surface(c) & otherwise \end{cases}$$

where ϕ is a percentage of freedom. For instance if $\phi = 120\%$, 20% of the total surface of children is left empty for the legibility of the diagram.

Furthermore, a node is expanded if its surface is higher than a percentage ζ. For instance, if $\zeta = 25\%$ the node will be expanded when its surface takes more than 25% of the screen surface. This choice allows us to adapt to various screen resolutions.

6 Conclusion and perspective

In this paper we have proposed an original method for the multi-level exploration of a lexical network. The graph underlying the lexical network has scale free and small-world properties. Even though we applied our approach to a given lexical network, we believe that our approach is general enough to apply to other networks with similar scale-free and small-world properties. For example, an interesting application would be the multiscale visualization of tags in social bookmarking systems.

In future work, we plan to extend our multi-level exploration tool with editing capacities so that it is possible for a user to modify the generated compound graph when necessary. We also want to integrate a search tool so that it is possible to automatically animate the graph toward a specific term. Finally, we plan to conduct controlled experiments to validate the approach on various datasets.

References

[1] J. Abello, S. G. Kobourov, and R. Yusufov. Visualizing large graphs with compound-fisheye views and treemaps. pages 431–441. Springer, 2004.

[2] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[3] D. Archambault and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE TVCG*, 13(2):305–317, 2007.

[4] D. Archambault, T. Munzner, and D. Auber. Grouse: Feature-based, steerable graph hierarchy exploration. pages 67–74, 2007.

[5] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In Seok-Hee Hong and Kwan-Liu Ma, editors, *APVIS*, pages 133–140. IEEE, 2007.

[6] F. Boutin and M. Hascoët. Focus dependent multi-level graph clustering. In *AVI'04*, pages 167–170. ACM, 2004.

[7] F. Boutin and M. Hascoët. Multi-level exploration of citation graphs. In Rachel Heery and Liz Lyon, editors, *ECDL*, volume 3232, pages 366–377. Springer, 2004.

[8] F. Boutin, J. Thièvre, and M. Hascoët. Multilevel compound tree - construction visualization and interaction. In *INTERACT*, volume 3585, pages 847–860. Springer, 2005.

[9] P. Eades and Q. Feng. Multilevel visualization of clustered graphs. pages 101–112. Springer-Verlag, 1996.

[10] C. Fellbaum. *Wordnet an electronic lexical database*. MIT Press, 1998.

[11] P. Gajer and S. G. Kobourov. Grip: Graph drawing with intelligent placement. *JGAA*, 6:2002, 2000.

[12] Emden R. Gansner. Topological fisheye views for visualizing large graphs. *IEEE TVCG*, 11(4):457–468, 2005.

[13] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm (extended abstract), 2004.

[14] Michael Junger and Mutzel Petra. *Graph Drawing Software*. Springer, 2004.

[15] M. Lafourcade. Conceptual vectors, lexical networks, morphosyntactic trees and ants : a bestiary for semantic analysis. In *SNLP 2007*, 2007.

[16] M. Lafourcade. Making people play for lexical acquisition. In *SNLP 2007*, 2007.

[17] Mathieu Lafourcade. Jeuxdemots website, November 2007-2008. http://jeuxdemots.org.

[18] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[19] G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGrawHill, 1983.

[20] D. Schaffer, S. Greenberg, L. Bartram, J. Dill, and M. Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *TOCHI*, 3:162–188, 1996.

[21] K. Sparck Jones. *Synonymy and Semantic Classification*. Edinburgh University Press, 1986.

[22] F. van Ham and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE TVCG*, 12(5):669–676, 2006. Member-Abello,, J.

[23] F. van Ham and Jarke J. van Wijk. Interactive visualization of small world graphs. In *INFOVIS'04*, pages 199–206. IEEE Computer Society, 2004.

[24] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *GD'00*, pages 171–182. Springer-Verlag, 2001.