Detecting keywords used by paedophiles

Christian Belbeze
Université de Toulouse, Institut de Recherche en Informatique de Toulouse. Email: christian@belbeze.com
Matthieu Latapy
LIP6 – CNRS and UPMC. Email: matthieu.latapy@lip6.fr

Abstract

We propose here a method to compute sets of strongly related keywords from logs of user queries. This method relies on the construction and analysis of a huge (weighted) (directed) graph, from which we extract aggregates of words. The challenge which we address is to obtain relevant aggregates with reasonable computational costs.

Keywords : Keyword, log file, search engine, cluster, aggregate, graph.

1. From a log file to a directed weighted graph

Given a large amount of queries (170 millions) entered by users searching for files in a P2P network: “Ten weeks in the life of an eDonkey server”, we create a graph. In this graph nodes are keywords and links are co-occurrence links: two words are linked together if they appear in the same query. Therefore only queries including more than one keyword are considered. The obtained graph has 2.8 million nodes (keywords) and 68 million links.

In the graph each element, node and link, has an associated **weight**. This weight is the number of times that the node or link has been found in the log file (excluding one keyword queries). We denote by $\mathbf{W_A}$ the weight of node A and by $\mathbf{W_{AB}}$ the weight of the link between A and B.

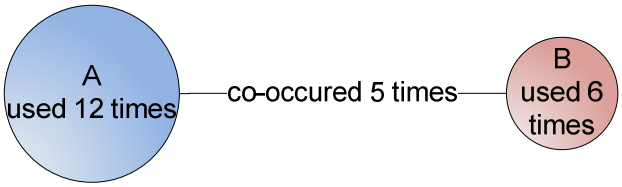


Figure 1: From log file to a graph: weighted nodes and link.

We now turn the obtained weighted undirected graph into a directed version as follows. Given two nodes A and B we define the Coefficient of Reliability (CR) from A to B, denoted by

$$CR_{A\longrightarrow B} = \frac{w_{AB}}{w_A}$$

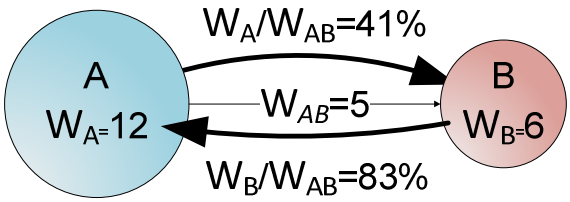
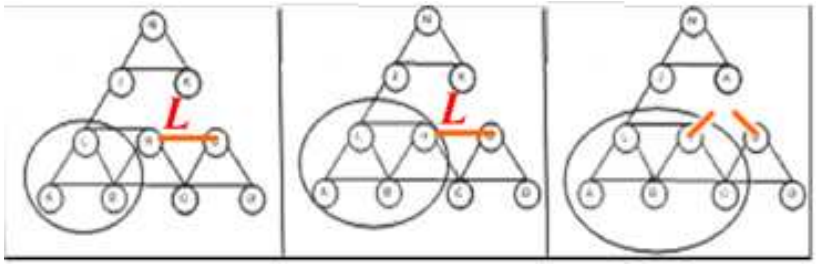


Figure 2: Directed version of the graph.

2. First method

The graph defined above encodes much information on the relations between words entered by users in their queries. Our goal now is to use this encoding to exhibit groups of words, called aggregates, with particularly strong relations. To achieve this, we will proceed as follows. We first compute the triangles in the graph (i.e. sets of three words with all these possible links between then) which constitute our initial aggregate. We then grow aggregates by merging triangles following these rules:



L is ***removed*** if $CR_{A \rightarrow B}$ **And** $CR_{B \rightarrow A}$ < **Vdw** [Too weak to keep]
But *L* is **kept** if $CR_{A \rightarrow B}$ **Or** $CR_{A \rightarrow B}$ > **Av** [Too big to ignore]

Figure 3: aggregate triangles and removing links.

- **Aggregate must remain biconnex :** A biconnex graph is a graph where each node is connected by at least two paths to any other node of the graph.
- **Links with a CR lower than a predefined minimal Value (Vdw) for the two nodes are removed the relation except if one of its CR is higher than an Activation value (Av), (cf. Figure 3).**

Experimentation and results.

Applying this method produces a massive aggregate containing typically more than 2 000 000 of words. This aggregate is too large to have any sense. The reason of that is that many links concern very rare words. Indeed than 50% of keywords are used once or twice. These links are always kept because the **CR is at least 50%.**

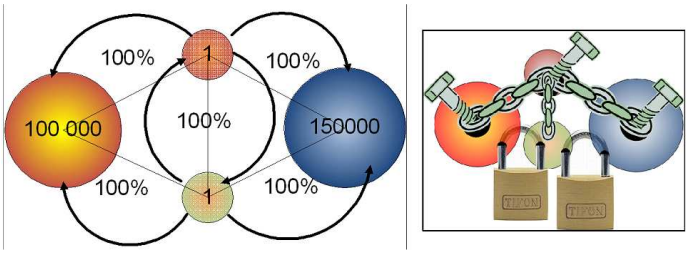


Figure 4: Rare words are too strongly linked with others words and especially with frequent words.

3. Improved method

The basic solution to improve the method above could be to remove rare words. This would not have really sense in pursuing our goal. We want to keep the opportunity to get rare and very used words in aggregates. To make this possible, we propose an algorithm that limits the size of aggregates. It relies on adapting parameters when aggregates reach a maximal defined size.

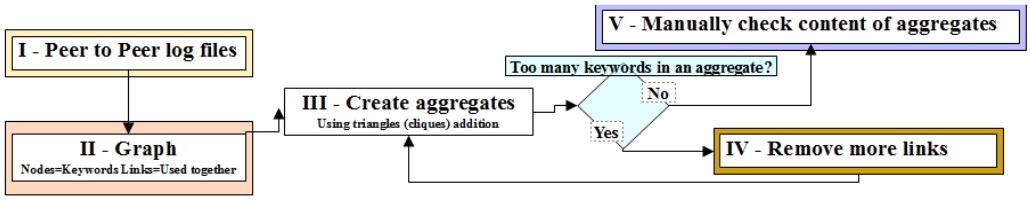


Figure 5: Aggregation including size limitation principle

Using the same algorithm as above, we first define a maximum aggregate size (**MAS**). Then we adapt 4 criteria I, the algorithm to keep aggregates size under this value (**MAS**). We defined for each of the four criteria, a start value and a final value. We defined too a number of step (**NbSteps**) to reach final criteria values. Each time aggregate reaches the maximum predefined size (**MAS**) we will increase Value Double Way (**Vdw**) and Activation Value (**Av**) to remove more links. At this step, we also modify the minimum (**Min valid weight**) and maximum (**Max valid weight)** weight of a node. We increase minimum weight and decrease maximum weight of node using specific formulas. By this way we will get limited number of keywords in each aggregate.

Parameters	Start value	Final value	New incremented values **IStep**=**IStep**+1
Vdw	3	10	3 + 7 * (**IStep** / **NbSteps**)
Av	10	51	10 + 41 * (**IStep**/ **NbSteps**)
Min valid weight	**Min(G.weight)** 1	**Avg(G.weight)** 70	**Avg(G.weight)** ^ (**IStep**/**NbSteps**)
Max valid weight	**Max(G.weight)** 328000	**Avg(G.weight)** +1 71	Max(**G.weight**) ^ ((**NbStep-IStep**)/**NbStep**) + **Avg(G.weight)**

IStep: Number of times aggregate reached the maximum size (**MAS**); **Max(G.weight):** Maximum number of use for a word in the graph; **Min(G.weight):** Minimum number of use for a word in the graph; **Avg(G.weight):** Average weight of keywords in the graph.

Figure 6: Limits and step modification. Numeric values are the ones use in the experimentation.

Experimentation and results.

We chose to use 50 steps and a size limit of 80 keywords by aggregate. Starting with a set of well known keywords the improved method creates aggregates including these keywords. More the keywords' weight decreases and more the number of aggregates, including the keyword, decreases too. In figure 7 you will find a very short result of how number of aggregates and size of then evaluate in comparing the included well known node weight.

Keywords	Weight	Number of aggregates	Max Size	Avg Size	Min Size
pthc	45737	96	78	9	3
incest	13609	70	52	11	3
ygold	9183	19	61	15	3
ptsc	3189	14	11	6	3
incesti	1277	2	4	3.5	3
inceste	1220	3	17	12	7
4yo	1042	4	14	9	4
3yo	832	3	12	10	8

Figure 7: Aggregates including 8 well known words.

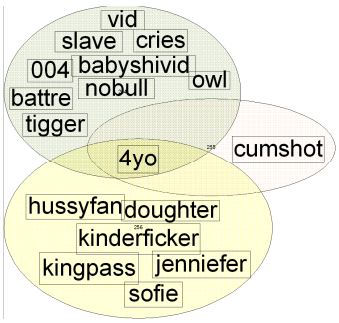


Figure 8: Sample of aggregates.

Figure 8 presents parts of aggregates including the word "4yo".

CONCLUSION

We presented a method to create aggregate of keywords from user queries encoded in a large graph. The method presents the advantage to keep a reasonable and a predefined maximum size for each aggregate. It produces seemingly relevant results, while keeping its computational cost reasonable. Assessing and refining the results is the next step to investigate.

REFERENCES

Aidouni, F. and Latapy, M. and Magnien, C. Ten weeks in the life of an eDonkey server, Sixth International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P 2009), May 29, 2009, Rome, Italy.

Cui, H., Wen et al 2002, Probabilistic query expansion using query logs. Proceedings of the eleventh international conference on World Wide Web, pp. 325 – 332.

Fu, L. et al , 2003, Collaborative querying through a hybrid query clustering approach, 2003, Digital libraries: Technology and management of indigenous knowledge for global access, ICADL, pp. 111-122..

Gangnet, M. and Rosenberg, B., 1993, Constraint programming and graph algorithms, Annals of Mathematics and Artificial Intelligence 8(3–4): 271–284.

Hoffmann, C. et al, 2000, Decomposition plans for geometric constraint systems, Proc. J. Symbolic Computation 2000.

Latapy, M., 2007. Grands graphes de terrain – mesure et métrologie, analyse, modélisation, algorithmique. Habilitation `a diriger des recherches, Université Pierre et Marie Curie, Paris, France.