Detecting Community[*]

Sander M. Bohte
Centrum Wiskunde & Informatica (CWI)
Science Park 123, NL-1098 XG Amsterdam, The Netherlands

September 5, 2011

It can be argued that most important issues are related to relative placements of certain elements versus others, and the detection of proper groupings. Typography in fact is one of the clearest examples of an application domain where proper groupings are paramount. As any sane sole knows, proper typography can make crappy research at least look good, and, conversely, can make the most excellent scientific output look quite bad (e.g. figure 1) (Leslie Lamport of course solved at least the relative placement of text and figures definitively in $\LaTeX$, using free energy minimization between spring-embedded document elements). The issue in fact extends to the traditional managerial tasks, such as the optimal group layout, and subsequent placement within the bright new CWI wing, the distribution of NWO bonus funding, and the planning of future leisure trips vs catching up on editorial work.

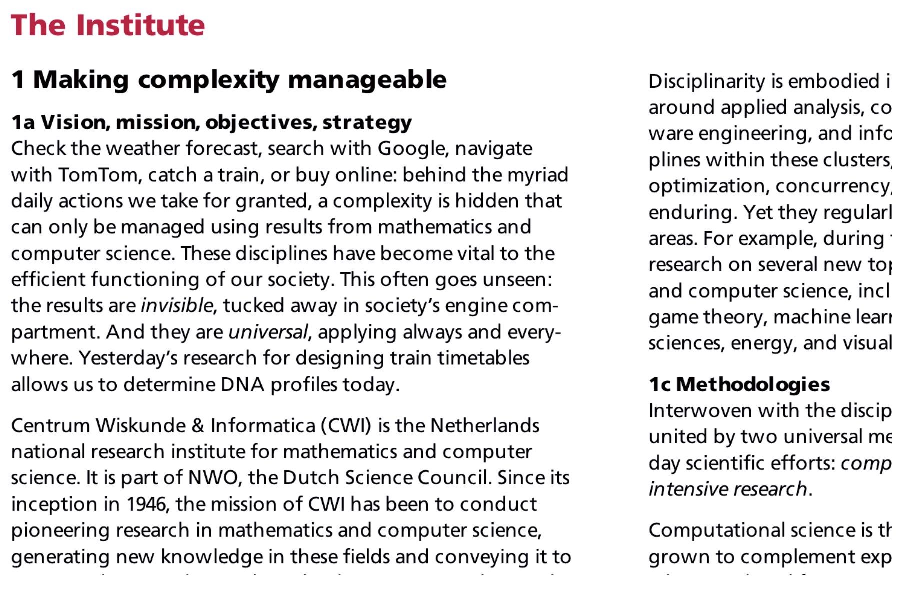
Here we consider a novel algorithmic approach for this important problem. Whereas simple Bayesian inference can solve trivial issues like wine-selection (winewinewine.com), many of the grouping and placement issues that Jan Karel wrestled with during his tenure as CWI director – and many important future decisions – can be cast into the graph-community detection domain, and as such subjected to various homeopathic solutions. For this, we turn to the vast knowledge that can be data-mined from the ~~Google~~ Bing collection of search archives – query logs.

Query Logs

Query logs contain the history of search terms entered by users into search engines such as Microsoft Bing. Since the AOL data release, it is generally well known that the most embarrassing, personal facts can be data-mined from these query-logs. From a managerial perspective, these logs thus offer many tools for management to achieve goals. In general however, old school ethical standards, combined the lack to a localized sample, prevent us from going this route. Fortunately, the usefulness of the data does not end here.

Given a large-scale query log, one of the most useful pieces of information it provides is the co-occurrence of words in different queries. The fact that two search

[*]The work was performed based on a Microsoft Research "Beyond Search" award, and SMB wishes to thank Microsoft Research for their support.



The Institute

1 Making complexity manageable

1a Vision, mission, objectives, strategy
Check the weather forecast, search with Google, navigate with TomTom, catch a train, or buy online: behind the myriad daily actions we take for granted, a complexity is hidden that can only be managed using results from mathematics and computer science. These disciplines have become vital to the efficient functioning of our society. This often goes unseen: the results are *invisible*, tucked away in society's engine compartment. And they are *universal*, applying always and everywhere. Yesterday's research for designing train timetables allows us to determine DNA profiles today.

Centrum Wiskunde & Informatica (CWI) is the Netherlands national research institute for mathematics and computer science. It is part of NWO, the Dutch Science Council. Since its inception in 1946, the mission of CWI has been to conduct pioneering research in mathematics and computer science, generating new knowledge in these fields and conveying it to

Disciplinarity is embodied i
around applied analysis, co
ware engineering, and infc
plines within these clusters,
optimization, concurrency,
enduring. Yet they regularl
areas. For example, during
research on several new to
and computer science, incl
game theory, machine lear
sciences, energy, and visual

1c Methodologies
Interwoven with the discip
united by two universal me
day scientific efforts: *comp*
intensive research.

Computational science is tl
grown to complement exp

Figure 1: Problematic kerning. Source: 1st edition CWI Selfevaluation 2011

keywords frequently appear together in the same query is known to imply a semantic distance between them. We consider how query log data can be used to derive cluster recommendations. Such an approach should lead to answers to questions such as: What kinds of keyword combinations are useful for finding certain types of holidays? For the analysis here, we chose 50 keywords related to the tourism industry (i.e. online bookings of tickets, travel packages and such). As an illustration, this example keyword set is interesting as one's leisure time increases.

Formally, let $N(T_i, T_j)$ denote the number of times two search terms T_i and T_j appear jointly in the same query. Let $N(T_i)$ and $N(T_j)$ denote the same number of queries leading to a click, in which terms T_i, respectively T_j appear in total (regardless of other terms they co-occur with). The cosine similarity distance between terms T_i and T_j is defined as:

$$Sim(T_i, T_j) = \frac{N(T_i, T_j)}{\sqrt{N(T_i) * N(T_j)}}. \tag{1}$$

Constructing keyword correlation graphs

The most intuitive way to represent similarity distances is through a keyword correlation graph. The results from our subset of 50 travel-related terms are shown in Figure 2. In this graph, the size of each node (representing one query term) is proportional to the absolute frequency of the keyword in all queries in the log. The distances between the nodes are proportional to the similarity distance between each pair of

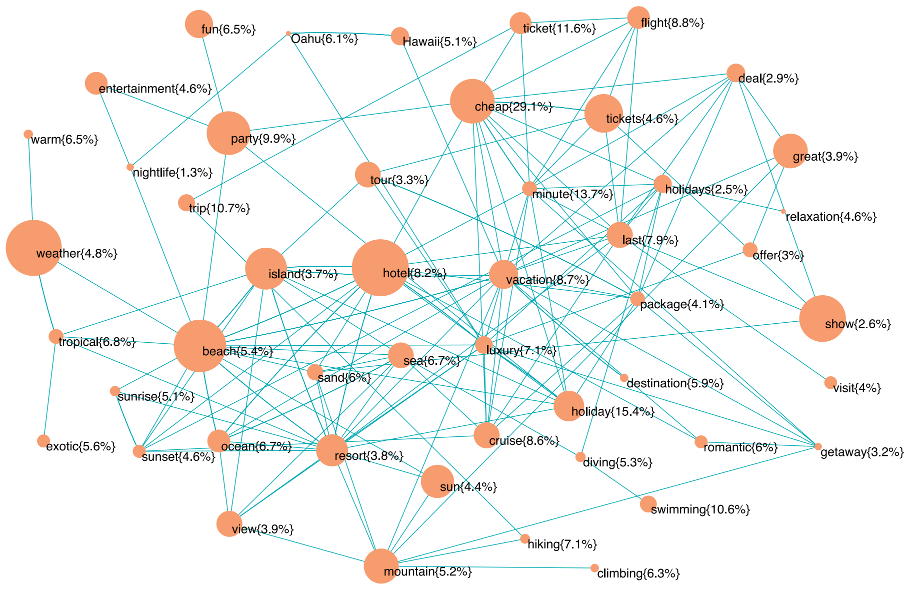


Figure 2: Visualization of a search term correlation graph, for a set of search terms related to the tourism industry. Each search term is assigned one colored dot. The size of each dot gives its relative weight (in total number of clicks received), while the distances between the dots are obtained through a spring-embedder type algorithm and are proportional to the co-occurrence of the two search terms in a query. Each dot is marked with its relative popularity with cheap-ass tour operators.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
beach	party	package	weather	getaway	diving	cruise	show	last
luxury	entertainment	vacation	exotic	romantic	swimming	sunrise	tickets	minute
hotel	nightlife	holidays	tropical			sunset	ticket	visit
island	fun	destination	warm				cheap	
resort	Hawaii	deal					flight	
sun	Oahu	tour						
mountain		offer						
ocean		great						
hiking								
climbing								
sea								
sand								

Figure 3: Optimal partition of the set of travel terms in semantic clusters, when the top 150 edges are considered. The partition was obtained by applying Newman's "community detection" algorithm to the graph from Fig. 2. This partition has a clustering coefficient Q=0.59.

terms, computed Equation 1, where the whole graph is drawn according to a so called "spring embedder"-type algorithm. In this type of algorithm, edges can be conceived as "springs", whose strength is inversely proportional to their similarity distance, leading to clusters of edges similar to each other to be shown in the same part of the graph. This method for instance underlies much of the LaTeX typography.

In order to visualize such query-terms in a graph (Figure 2), we used Pajek, a free and powerful academic graph drawing package. Note that not all edges are considered in the final graph. Even for 50 nodes, there are $\binom{50}{2} = 1225$ possible pairwise similarities (edges), one for each potential keyword pair. Most of these dependencies represent, however, just noise in the data, and our analysis benefits from using only the top fraction, corresponding to the strongest dependencies. In the graph shown in Figure 2, only the top 150 strongest dependencies were considered in the visualization.

The most interesting effect to observe in Figure 2 are the term clusters that emerge in different parts of the graph, from the application of the spring-embedder algorithm. For example, the leftmost part of the graph has 4 terms related to weather, such as "warm", "tropical" and "exotic". On the top left part of the graph, one can find terms such as "entertainment", "nightlife", "party" and "fun", while the very bottom part includes related terms such as "climbing", "hiking" and "mountain". The central part of the graph includes terms such as "beach", "sand", "sea", "resort", "ocean", "island" etc. Additionally, pairs of terms one would naturally associate do indeed appear close together, such as "romantic" and "getaway" and "sunset" and "sunrise" and "ocean". Note that family-island-getaway is not on this map, ensuring relative quietness for this type of activity.

Automatic identification of sets of keywords

To find relevant clusters of keywords, we use so called "community detection" algorithms, also inspired by complex systems theory. In network or graph-theoretic terms, a community is defined as a subset of nodes that are connected more strongly to each other than to the rest of the network (i.e. a disjoint cluster). If the network analyzed is a social network (i.e. vertices are people), then "community" has an intuitive interpretation. However, the network-theoretic notion of community detection algorithm is broader, and has been applied to domains such as networks of Ebay items, publications on arXiv, or even food webs.

Let the network considered be represented as graph $G = (V, E)$, when $|V| = n$ and $|E| = m$. Each $v \in V$ must be assigned to exactly one group (i.e. community or cluster) $C_1, C_2, ...C_{n_C}$, where all clusters are disjoint.

In order to compare which partition is "optimal", the metric used is *modularity*, henceforth denoted by Q. Intuitively, any edge that in a given partition, has both ends in the same cluster contributes to increasing modularity, while any edge that "cuts across" clusters has a negative effect on modularity. Formally, let $e_{ij}, i, j = 1..n_C$ be the fraction of all edge weights in the graph that connect clusters i and j and let $a_i = \frac{1}{2}\sum_j e_{ij}$ be the fraction of the ends of edges in the graph that fall within cluster

Algorithm 1 *GreedyQ Partitioning*: Given a graph $G = (V, E), |V| = n, |E| = m$ returns partition $< C_1, ...C_{n_C} >$

```
1. $C_i = \{v_i\}, \forall i = \overline{1, n}$
2. $n_C = n$
3. $\forall i, j$, normalize $e_{ij}$
4. repeat
5.    $< C_i, C_j >= \text{argmax}_{c_i,c_j}(e_{ij} + e_{ji} - 2a_i a_j)$
6.    $\Delta Q = \max_{c_i,c_j}(e_{ij} + e_{ji} - 2a_i a_j)$
7.    $C_i = C_i \bigcup C_j$, $C_j = \emptyset$ //merge $C_i$ and $C_j$
8.    $n_C = n_C - 1$
9. until $\Delta Q \leq 0$
10.$maxQ = Q(C_1, ..C_{n_C})$
```

i. The modularity Q of a graph $|G|$ with respect to a partition C is defined as:

$$Q(G, C) = \sum_i (e_{i,i} - a_i^2). \tag{2}$$

Informally, Q equals the fraction of edges in the network that fall within clusters, minus the expected value of the fraction of edges that would fall within the same cluster, if all edges would be assigned using a uniform, random distribution.

It is easy to see that if $Q = 0$, then the chosen partition c shows the same modularity as a random division. A value of Q closer to 1 is an indicator of stronger community structure - in real networks, however, the highest reported value is $Q = 0.75$. In practice, it was found (based on a wide range of empirical studies) that values of Q above around 0.3 indicate a strong community structure for the given network.

A greedy graph partitioning algorithm

Since we have established our framework, we can now formally define the graph partitioning algorithm. The number of possible partitions (clusters) for this problem is at least 2^{n-1} (e.g. for 50 terms: $2^{50} > 10^{15}$). Therefore, to explore all these partitions exhaustively would be clearly unfeasible. We show results for community detection using process of bottom-up successive joining of keywords, *GreedyQ*, formally specified as Algorithm 1. Initially, each of the vertices (in our case, each keyword) is assigned to its own individual cluster. Then, at each iteration of the algorithm, two clusters are selected which, if merged, lead to the highest increase in the modularity Q of the partition. The algorithm stops when no further increase in Q is possible by further merging.

The partition in figure 3, resulting directly from applying Algorithm 1, achieves a very intuitive division of keywords into relevant clusters, and, furthermore, it fits well with what can be graphically observed in Figure 2: most of the clusters obtained automatically after partitioning can be identified in different parts of the graph.

Discussion

In this short paper, we presented an approximate, greedy solution for community detection given vast data-mining resources. We believe it can be applied to much of the tough decision-making that a director is inevitably faced with. Applying it to some CWI data, we observe several strong communities, with what appears to be growing interconnections. Of course, as is well known in the algorithmic community, algorithms derive their generalizing power from the problem abstraction. For practical problems, it may be the case that additional constraints need to be considered.

Note that we by no means presented the optimal solution, as we cannot account for extreme outliers that, given the heavy tails of collective human behavior, invariably seems to lead to the Pareto principle. We do believe however that in small settings, the community detection algorithm can be applied, to such problems as holiday selection, book-topic clustering, and the combination of fine foods (again, the wine selection problem can be solved with simple Bayesian machine learning).

Acknowledgement We thank Roeland Merks for the general idea of this brief paper, and Valentin Robu for the original research and text, as well as doing the really hard work. We thank Nicole Immorlica for her transcendent contributions, and Victor Gomez for generally totally being around for us.