On the Diversity Within Internet Search Engines: A Clustering Approach Using Query Based Metrics*

Sibel Adalı, Brandeis Hill, Malik Magdon-Ismail
Rensselaer Polytechnic Institute
{sibel, hillb, magdon}@cs.rpi.edu

Abstract

We present experimental results on how internet search engines compare with each other based on their query results. If the relationship between two rankers is substantially different for two different queries, it suggests that the queries are of a different nature. We use this observation to also investigate how queries relate to each other based on the relative performance of different internet search engines on them. Specifically, we show how clustering approaches can reveal relationships among search engines, and relationships among queries modulo *those search engines. Our results have implications on aggregation of search results and on reverse engineering of ranking formulas used by search engines.*

1 Introduction

Internet search is dominated by a small set of search engines (*Google, Yahoo, MSN,...*). In total, there are about fifty reasonably frequented search engines, which also includes meta-search engines such as *Dogpile*. While some of these fifty are offshoots of the more dominant ones, there appears to be considerable diversity among internet search engines. Diversity is very useful, especially from the point of view of search result aggregation (a vote among many unbiased and independent experts usually yields a better vote). In this respect, it is especially important to have some idea about what the independent search engines are (in addition to their "accuracies"). An aggregator which blindly averages all the results from all the search engines could fall into the following sort of trap: suppose that there are two independent search engines G and Y, and the universe of search engines consists of offshoots and clones of these two; to be specific, suppose that under different guises G is represented 49 times and Y only once. Then a blind aggregation of all 50 internet search engines is essentially equivalent to the one independent opinion G.

Reverse engineering of internet search engine ranking formulas have received much interest in research. In particular, what factors (and their corresponding weights) are used in the ranking formula. A popular approach is to use some machine learning technique, which make necessary the creation of a training data set. The resulting inferences are generally significantly affected by the specifics of the training data and the constraints used in the learning. For example, knowing that two search engines are similar constrains them to be both using somewhat similar features. Learning what their factors (and weights) are is now an easier task as one uses the data on both search engines to learn one set of features (since we know the engines are similar). As another example, if two search engines partition the query space into two sets, Q_1 on which they both produce very similar results and Q_2 on which they are quite different, this suggests a partitioning of their factors: the factors they have in common must be the factors which have high values and are discriminative in Q_1 and they should differ on the factors which have high values in Q_2.

While we do not address aggregating or reverse engineering here, we recognize the importance for these and other problems of understanding the landscape on which internet search engines live, and further how the query space is related to this landscape. Human analysis of such relationships, which may be evolving, is tedious and not scalable. Automated tools for performing such analysis would be useful, and we take a step in this direction. We develop methods to analyze the diversity among search engines. In particular, we address the following questions

- What is the diversity among internet search engines?
- Are there specific groups of queries identified by differences in how internet search engines relate to each other on these groups?

Our approach is to use a metric between incomplete ordered

*This work was partially supported by the National Science Foundation under grants EIA-0091505, IIS-0324947, and CNS-0323324. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

lists to build an estimate of similarities between search engines using their search results, which are ordered lists. We then use clustering algorithms (allowing for overlaps) to identify clusters of search engines which behave similarly. Our approach is validated by the fact that we do reproduce many known relationships among internet search engines. Our results are also interesting in that we also find some unexpected relationships. Armed with a technique for identifying the similar and dissimilar search engines with respect to query results, we can now compare for two given queries q_1 and q_2 whether they induce the same relationships among the search engines. If so, then the queries themselves are "similar". Thus, we can cluster the queries into groups, where in each group the search engines are similarly related and between two groups the relationships between the search engines differ. Thus, we are clustering the query space with respect to the search engine behavior.

Paper organization. Next we discuss some related work, followed by some preliminaries. We then present our methodologies in Section 4 followed by some experimental results. We conclude in Section 6.

2 Related Literature

To our knowledge, there is no work in studying the diversity among search engines and clustering of search engines based on the similarity of their rankings. Clustering has been used for analyzing user logs [2] to classify user queries into topics. Wen, Nie and Zhang [14] construct dense query clusters using both textual similarity between queries and the similarity between the documents clicked by the user in response to these queries. This approach does not use any labeled information in contrast with the previous one.

Work on reverse-engineering ranking functions from query results of one or more search engines is relevant to our work. The methods used in determining unknown ranking functions typically learn the weights of a number of features in the target ranking function. Sedigh and Roudaki [12] use linear regression over a number of selected features to learn the rank of a webpage. Pringle, Allison and Dowe [11] use decision trees to learn the ranking function. Bifet et. al. [4] use a combination of learning methods such as linear regression, support vector machines and decision trees, and find that different combination of these methods perform well for different queries. This indicates that queries fall into different partitions, where ranker behavior differs. This is one of the motivations behind our work. Cohen et. al. [5] use expert classifiers to learn whether an object should be higher than another for a specific domain. They use different queries in this domain as experts and use an online learning algorithm based on the "Hedge" algorithm to find a linear combination of ranking experts. Diez et. al. [6] use self-organizing maps to learn a ranking function. Their methods are able to learn both linear and non-linear functions that use thresholding. Joachims [9] learns the relevant objects for a search query and their importance by using the order in which a user clicks on the objects. This information is then fed to a learning method to learn the best aggregation method for a meta-search engine that combines rank information with other features. Finally, statistical analysis is used in locating pages that are considered spam [8, 3]. These methods can benefit greatly from principled methods for selecting training queries and search engines.

3 Preliminaries

We assume that $\mathcal{O} = \{o_1, \ldots, o_n\}$ denotes the universe of all objects that can be queried by search engines. A *search query* $\mathbf{q}$, is a request for information, normally expressed by a set of keywords. A *ranked list* denoted by ℓ is a partial list containing objects from $\mathcal{O}$. The function $rank(\ell, o)$ returns the rank of object o in the ranked list ℓ (rank 1 is the highest hanked (best) object).

A *search engine* or *ranker* $\mathcal{R}$ indexes a subset of the objects in $\mathcal{O}$. These objects are called the database of this ranker $\mathcal{R}$. Each ranker $\mathcal{R}$ returns a ranked list of objects from its database in response to a search query $\mathbf{q}$. The specifics of how search queries are computed is not of interest in this paper. We are only interested in ranker pairs whose databases have non-empty intersection.

3.1 Similarity Between Incomplete Ordered Lists

To evaluate the closeness of two ranked lists, we use the precision and Kendall-tau measures as is common in the information retrieval literature. Since the number of objects returned by search engines may vary from search engine to search engine, and from query to query, we need to normalize the values returned by each measure.

For two ranked lists ℓ_1 and ℓ_2, the *symmetric, normalized Precision* is the percent of the larger list contained in the smaller list, $\overline{pr}(\ell_1, \ell_2) = |\ell_1 \cap \ell_2| / \max\{|\ell_1, \ell_2\}$. The Kendall-tau, $\tau(\ell_1, \ell_2)$, counts the number of pairs of objects that appear in different relative order in ℓ_1 and ℓ_2. This is a more informative measure than the Precision for comparing ranked lists as it takes into account the actual rankings of common objects (the Precision only looks at the number of common objects): $\tau(\ell_1, \ell_2) = \sum_{i<j} \epsilon_{ij}$, where the sum is over all pairs of objects (o_i, o_j) appearing in $\ell_1 \cup \ell_2$ and $\epsilon_{ij} = 1$ if $(rank(\ell_1, o_i) - rank(\ell_2, o_j) \cdot (rank(\ell_1, o_i) - rank(\ell_2, o_j)) < 0$ and zero otherwise. Following Fagin et. al. [7] we set $rank(\ell, o) = |\ell| + 1$ if o does not appear in ℓ. This assumes that all engines rank all objects that appear in the union of the ranked lists, which is a reasonable assumption for most of the large search engines. When o_i, o_j appear in one list, and neither in the other, $\epsilon_{ij} = 0$ which is an implicit bias toward existing rankers since it assumes their ranking to be accurate.

Note that precision measures similarity, but Kendall-tau is a distance. Clustering algorithms typically operate using

similarities, so we convert the Kendall-tau to a similarity measure by first normalizing it to the range $[0, 1]$ (by dividing by the max possible value it can have for the lists ℓ_1, ℓ_2), and then subtracting from 1. Thus, the *rank agreement* measure, denoted by $sim_\tau(\ell_1, \ell_2)$ is given by

$$sim_\tau(\ell_1, \ell_2) = 1 - \frac{\tau(\ell_1, \ell_2)}{\tau_{max}(\ell_1, \ell_2)}$$

For our choice of ϵ_{ij}, $\tau_{max}(\ell_1, \ell_2) = |\ell_1||\ell_2| - \binom{|\ell_1 \cap \ell_2|+1}{2}$. Note that a high rank agreement implies high precision, but a high precision does not imply a high rank agreement.

3.2 Clustering

We now introduce a number of techniques for comparing ranked lists. In these tests, we use the clustering algorithm introduced in [1] to construct overlapping clusters in a weighted graph. The main idea of the algorithm is based on defining a cluster as a locally optimal subset of the nodes, with respect to some clustering metric (density). Let C be a candidate cluster. The two clustering metrics we use, which were suggested in [1] are

$$E(C) = \frac{p_{in}}{p_{in} + p_{out}}, \qquad E(C) = \frac{W_{in}}{W_{in} + W_{out}},$$

where W_{in} is the sum of edge weights internal to the cluster and W_{out} is the sum of edge weights on edges connecting nodes in the cluster to the outside; p_{in} normalizes W_{in} by the number of possible internal edges (hence it is the average internal weight) and similarly p_{out}. The algorithm consists of two parts. First, a candidate set of clusters is constructed. We used the Link-Aggregate procedure in [1] for this step. Each of these candidate seed clusters is updated iteratively until a locally optimum point is reached. The Iterative-Scan algorithm [1] accomplishes this task by updating one node at a time in the direction of increasing density. Efficient algorithms for performing these iterative updates were given in [1]. The set of clusters resulting after iterative optimization of all the seed clusters is the set of clusters we use in our analysis.

This clustering algorithm differs from the conventional clustering methods that try to partition the graph. In our setting, it is desirable to allow clusters to overlap, and the algorithms in [1] allow us to discover distinct but possibly overlapping clusters.

4 Comparing Search Engines

We will use the clustering algorithms to compare the ranked lists returned by search engines. An object in our setting is a URL ranked by a search engine. Suppose, we are given search engines $\mathcal{R}_1, \ldots, \mathcal{R}_s$, queries $\mathbf{q}_1, \ldots, \mathbf{q}_z$ and ranked lists ℓ_i^y corresponding to search engine $\mathcal{R}_i$'s output for query $\mathbf{q}_y$. For each ranked list, we use the top K results, where K was set to 200 (note that some engines return fewer than K results in which case all their results are used).

4.1 Similarities Between Search Engines

To investigate the diversity among search engines, we construct the *search engine similarity graph*, which is a complete undirected weighted graph $G_\mathcal{R}(V, E)$ on the vertex set $V = \{\mathcal{R}_1, \ldots, \mathcal{R}_s\}$. The weight w_{ij} of the edge between search engines $\mathcal{R}_i, \mathcal{R}_j$ is the average similarity over all queries between the two search engine ranked lists,

$$w_{ij} = \frac{1}{z} \sum_{y=1}^{z} sim_\tau(\ell_i^y, \ell_j^y)$$

An example is shown in Figure 1. Clustering this graph will return sets of search engines which are similar (on average) to each other, and somewhat different compared to engines outside the cluster – a cluster tends to have large internal edge weights (similarity) and small external edge weights.

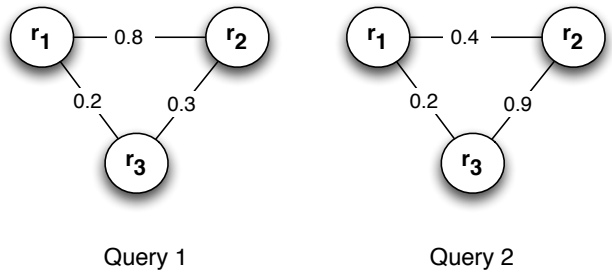

Figure 1. Rank agreement between search engines for two different queries

4.2 Similarities Between Queries

We also wish to find groups of queries that produce similar ranking behavior among the search engines. To define what similar ranking means, we can use the rank agreement (sim_τ) measure introduced earlier. To this end, we compute a query specific version of the graph from the previous section (instead of averaging over queries). In other words, let $G_\mathcal{R}^y(V, E)$ be a complete undirected weighted similarity graph on the the same vertex set $V = \{\mathcal{R}_1, \ldots, \mathcal{R}_s\}$ where now $w(i, j) = sim_\tau(\ell_i^y, \ell_j^y)$. We will call this the *query specific search engine similarity graph.* Two such example graphs are shown in Figure 1. In this example, we can conclude that $\mathcal{R}_1, \mathcal{R}_2$ are similar with respect to Query 1, but $\mathcal{R}_2, \mathcal{R}_3$ are similar with respect to Query 2, or that these queries are somehow distinct in nature in that they induce different relative behavior among the search engines. As we argued earlier, we can surmise in this example that factors

which score high for query 1 must have similar weights in $\mathcal{R}_1$ and $\mathcal{R}_2$ and factors which score high for query 2 must have similar weights in $\mathcal{R}_2$ and $\mathcal{R}_3$. This example is illustrative with the caveat that a real-life scenario is going to be much more complicated. There are two ways to compare the queries based on the similarity graph. The first is to directly compare the similarity graphs via their distance matrices. We give two measures of such similarity based on the eigen-directions of the distance matrices and the Frobenius distance between the distance matrices. The second approach is to compare the similarity of some high-level features of the graphs, in particular how the search engines cluster into groups. If the groups are similar in both cases, then we say that the queries are similar. We give two approaches to comparing clusterings of two query specific similarity graphs based on a matching and a fuzzy matching between the clusters.

Instead of analyzing pairs of queries as in the above example, we would like to analyze all the queries in our set simultaneously. Once we have the similarities between queries, we can compute the *query similarity graph* $G_{\mathbf{q}}(V, E)$, which is a complete undirected weighted graph on the vertex set $V = \{\mathbf{q}_1, \ldots, \mathbf{q}_s\}$. The weight of edge w_{xy} is the query similarity between $\mathbf{q}_x$ and $\mathbf{q}_y$. We can now cluster this query similarity graph as a whole to identify structure in query space.

Principal Eigenvector Similarity ($Eigen(\mathbf{q}_x, \mathbf{q}_y)$) Assume that for two queries $\mathbf{q}_x$ and $\mathbf{q}_y$, we have computed the query specific search engine similarity graphs $G^x_{\mathcal{R}}(V, E)$ and $G^y_{\mathcal{R}}(V, E)$. $Eigen$ is based on the weighted adjacency matrix representations M_x, M_y of of $G^x_{\mathcal{R}}(V, E), G^y_{\mathcal{R}}(V, E)$. Let $\mathbf{e}_x, \mathbf{e}_y$ be the normalized principal eigenvectors M_x, M_y respectively. Then, $Eigen$ is the cosine of the angle between the eigenvectors,

$$Eigen(\mathbf{q}_x, \mathbf{q}_y) = |\mathbf{e}_x \cdot \mathbf{e}_y|$$

Frobenius Norm Similarity $Frobenius(\mathbf{q}_x, \mathbf{q}_y)$ $Frobenius$ is a matrix norm which compares the adjacency matrices M_x, M_y elementwise by taking the sum of squared differences,

$$Frobenius(\mathbf{q}_x, \mathbf{q}_y) = \| M_x - M_y \|_F,$$

where $\| A \|_F = \sqrt{\sum_{i,j=1}^{s} A_{ij}^2}$. Since the Frobenius norm is a distance, we convert it to a similarity by normalizing to the range $[0, 1]$ over all pairs of queries and then subtracting from 1.

Note that both $Eigen$ and $Frobenius$ are global measures that compute similarity over the whole graph. The next two measures focus on local properties of the search engine graphs, namely the clusters.

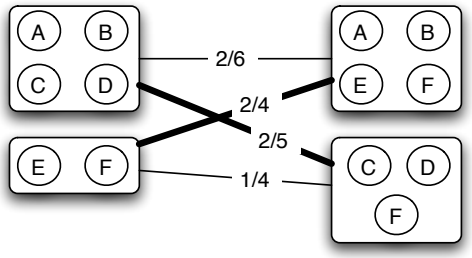

Figure 2. An example of one-to-one matching of clusters

Cluster Matching Similarity We now develop similarity measures which do not depend on the specifics of the search engine similarity graphs, but rather some of its high level features. In particular, if both similarity graphs have similar clusters, then the graphs are similar. Assume that the search engine similarity graphs $G^x_{\mathcal{R}}(V, E), G^y_{\mathcal{R}}(V, E)$ have been computed for queries $\mathbf{q}_x$ and $\mathbf{q}_y$. We cluster each graph independently. Let $\mathcal{C}_x = \{C^x_1, \ldots, C^x_p\}$ be the clusters in $G^x_{\mathcal{R}}$, and similarily define $\mathcal{C}_y = \{C^y_1, \ldots, C^y_q\}$. Note that the clusterings may overlap and they need not cover all the search engines. We can assume that $p = q$ by adding empty sets to one of the clusterings. We will use a similarity measure between the two sets of clusters $\mathcal{C}_x, \mathcal{C}_y$ as a measure of the similarity between the two queries $\mathbf{q}_x, \mathbf{q}_y$. We first need a similarity measure between two clusters (sets). We will use the set difference measure,

$$sim_M(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}.$$

Note that sim_M is defined even when one of the sets is empty. To measure similarity between the sets of clusters $\mathcal{C}_x, \mathcal{C}_y$, we first find a maximum matching for the complete weighted bipartite graph on the vertex set $\{C^x_1, \ldots, C^x_p; C^y_1, \ldots, C^y_q\}$ with the implied bi-partition. The weight of the edge (C^x_i, C^y_j) is $sim_M(C^x_i, C^y_j)$. We define the similarity between the clusterings $Cluster(\mathcal{C}_x, \mathcal{C}_y)$ as the average weight (per pair) of this maximum matching. For efficiency purposes, we use a greedy approximation to the weight of the maximum matching:

1: $\mathsf{sim} \leftarrow 0$; $V_1 \leftarrow \mathcal{C}_x$, $V_2 \leftarrow \mathcal{C}_y$;
2: **while** $|V_1 \cup V_2| > 0$ **do**
3: $\quad (a^*, b^*) = \text{argmax}_{a \in V_1, b \in V_2} sim_M(a, b)$;
4: $\quad \mathsf{sim} += sim_M(a^*, b^*)$; $V_1 \leftarrow V_1 \setminus a^*$, $V_2 \leftarrow V_2 \setminus b^*$;
5: **return** $\mathsf{sim} / \max\{|\mathcal{C}_x|, |\mathcal{C}_y|\}$

Figure 2 shows an example with the maximum matching selected in bold. The similarity, given by the average weight, is $\frac{1}{2} * (\frac{2}{4} + \frac{2}{5}) = 0.45$.

Fuzzy Cluster Matching Similarity The previous measure allows one to one matches only. However, it does not consider the contents of these clusters. Suppose query $\mathbf{q}_x$ induces two clusters $\{1,2\}$ and $\{3,4\}$ and query $\mathbf{q}_y$ induces the single cluster $\{1,2,3,4\}$. It is possible to match both $\{1,2\}$ and $\{3,4\}$ with the $\{1,2,3,4\}$ and assign a weight to each match accordingly. The many-to-many match becomes especially important in our setting since the clusters in a clustering may overlap. To compute the fuzzy match, we use an algorithm from the image retrieval community on matching regions [10]. Let $den(C)$ denote the density of cluster C, as specified by some metric, for example p_{in}, the average weight of the edges in the cluster. Since the edge weights (similarities between search engines) are in $[0,1]$, the density is also in $[0,1]$. We give the main idea behind this similarity measure, and refer to [10] for the details. The main idea is that one matches density, not clusters. To be specific, begin as with the greedy matching algorithm by identifying the pair of clusters $C_x \in \mathcal{C}_x$ and $C_y \in \mathcal{C}_y$ which maximize $sim_M(C_x, C_y)$. The weight of the fuzzy matching is incremented by $sim_M(C_x, C_y) \cdot \min den(C_x), den(C_y)$ (i.e. the similarity of the match is weighted by the density of the match). If $den(C_x) = den(C_y)$ then both clusters completely match each other, and they are removed from the remainder of the algorithm, which continues as usual. Otherwise suppose that $den(C_x) < den(C_y)$ (the reverse case is similar). In this case, C_x is removed and the $den(C_y) \leftarrow den(C_y) - den(C_x)$ (i.e. part of the density of C_y has been accounted for, and only the remaining density can be matched). After one of the clusterings have been exhausted, the resulting fuzzy matching weight is divided by $|\mathcal{C}_x| + |\mathcal{C}_y|$ to obtain the similarity measure $F - Cluster(\mathbf{q}_x, \mathbf{q}_y)$. The fuzzy matching allows many-to-many matches taking into account the cluster densities. Generally, one expects a dense cluster to match with multiple sparse clusters. The results depend on the number of clusters for each query, so we only consider the top 2 matches in our experiments with 11 search engines. It is rare to have more than 2 clusters so this decision does not significantly impact our results.

5 Experimental Results

We have collected ranked lists for 40 search engines and 322 queries. In most cases, we had about 200 results for each query and search engine. But, the number differed from query to query. The first group of 49 queries were chosen from the 2003 TREC competition on topic distillation [13]. The remaining 273 queries were collected manually from the search spy functionality of the Excite search engine. We have sampled queries from Excite at different points in time over a span of one week. We excluded queries for specific sites (such as ebay.com and ebay) and involving sexually explicit terms.

The 40 search engines used in our tests are given in Figure 3. We used this large set for validating our results. There are multiple cases where the same search engine results are returned under different names. We display the high similarity ($> 90\%$) amongst search engines in Figure 4. For example, we observed MetaCrawler and WebCrawler return almost identical results. Similarly, A9 and Google are identical if personalization is not considered. There are also different relationships between different search engines that may result similarity between the results. For example, certain search engines use the directory services of DMOZ. We extracted the URLs for only the regular search results excluding sponsored links. Figure 5 shows the clusters for the search engine graph for the 49 queries using the 40 search engines.

For our query similarity tests, we used a smaller subset of search engines to exclude almost all search engines that are identical. This list includes regular search engines as well as metesearch engines (Clusty, Dogpile, Metacrawler, Search). We will examine the similarities between metasearch and search engines in detail in this section. The 11 search engines are marked with * in Figure 3. In the first set of experiments, we examined the 49 queries from TREC for each similarity measure between queries. We observed that the eigenvector and Frobenius norm measure produce either all queries in one cluster or there were several clusters. These keywords of these queries in the set of small clusters are seemingly unrelated. For example, for the eigenvector measure, the queries 'affirmative action' and 'early childhood education' formed a cluster while for the Frobenius norm, there is a cluster of queries 'anthrax', 'death penalty', 'juvenile delinquency', 'Lewis and Clark expedition' and 'legalization of marijuana'. Even though the keywords of the queries are not similar; however,the underlying evaluation of a webpage for these search engines are related. Essentially, the search engines may determine the order of the webpages in the same manner, either making a more accurate ordering for the user or making similar mistakes. The cluster match measures formed one large cluster of nearly all the queries and a number of small queries of size 2. Interestingly, the small clusters formed from the cluster match were different from the clusters formed using both the eigenvector and Frobenius norm measure. This results provides evidence that there are similarities amongst queries and search engines where one measure highlights one feature of a search engine and another measure weighs another feature more highly.

In the second set of experiments, we evaluated 273 manually collected queries. The clusters formed from using the eigenvector and Frobenius norm vectors produced similar results seen with the TREC queries. The many-to-many cluster match measure produces a number of clusters of

varying sizes. Once again, the keywords of these queries have little in common, but clusters are formed because the search engines within a cluster evaluated the webpages similarly. This leads to the conclusion that either the search engines are similar, e.g. same database of URLs, or the features of those webpages are evaluated similarly.

6 Conclusion

Our results on clustering of search engines reveal some expected results, by discovering some well known and strong relationships between "instances" of the same search technology. In particular, the Google cluster, which includes AOL, A9, Alexa, and Netscape; the Yahoo! cluster, which includes Altavista and Kanoodle. Interestingly though, some unsuspected relationships were found, in particular, it seems possible to determine which search engines are the more dominant in some of the meta-search engines. In particular, Search seems to fall into the Google cluster, AlltheWeb seems to fall into the Yahoo! cluster. The other meta search engines appear to form a cluster, indicating that they probably do not rely too heavily on any one search engine, and in this respect they are similar. Figure 5 gives the details of the clusters found. As can be seen, some clusters are subsets of larger clusters, which indicates the strengths of the relationships.

With respect to clustering in query space, our preliminary results do find clusters, however, to the human eye, these patterns do not seem to follow any pattern. In a mild sense, the clusters seem to fall into very specific keywords and very non-specific keywords. One would expect all search engines to be somewhat random on very non-specific queries, in which case their relationship will be somewhat random. On the other hand, one would expect that on very specific queries, all search engines would return very similar results, and hence the relationship would be one very tightly linked cluster. On the one hand the fact that the we did not observe a clear pattern in the query clusters does not allow us to validate our techniques. On the other hand, it indicates that such a task of clustering queries may be quite subtle and hard for a human eye, which justifies the need for some automated approaches to these issues. We have taken a step in this direction using the relative performance of search engines on queries as a means of differentiating between classes of queries.

References

[1] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *IEEE Symposium on Intelligence and Security Informatics (ISI 2005)*, pages 27–36, 2005.

[2] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2005)*, pages 42–49, 2005.

[3] A. Benczur, K. Csalogany, T. Sarlos, and M. Uher. Spamrank - fully automatic link spam detection. In *International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[4] A. Bifet, C. Castillo, P. A. Chirita, and I. Weber. An analysis of factors used in search engine ranking. In *Proceedings of the International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[5] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research (JAIR)*, 10:243–270, 1999.

[6] J. Diez, J. J. del Coz, O. Luaces, F. Goyache, J. Alonso, A. M. Pena, and A. Bahamonde. Learning to assess from pair-wise comparisons. In *Proceeding of the Ibero-American Conference on Artificial Intelligence (IBERAMIA)*, volume 2527 of *Lecture Notes in Computer Science*, pages 481–490. Springer, 2002.

[7] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discrete Mathematics*, 17(1):134–160, 2003.

[8] D. Fetterly, M. Manasse, and N. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the International Workshop on the Web and Databases*, pages 1–6, 2004.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD*, pages 133–142, 2002.

[10] J. Li, J. Z. Wang, and G. Wiederhold. IRM: integrated region matching for image retrieval. In *ACM Multimedia*, pages 147–156, 2000.

[11] G. Pringle, L. Allison, and D. L. Dowe. What is a tall poppy among web pages? *Computer Networks and ISDN Systems*, 30:369–377, 1998.

[12] A. K. Sedigh and M. Roudaki. Identification of the dynamics of the google's ranking algorithm. In *IFAC Symposium On System Identification*, 2003.

[13] trec.nist.gov/data/topics_eng/2003.distillation_topics.1 50.txt. Trec 2003 web topic distillation topics, 2003.

[14] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, 2002.

A9	A9	CLU	Clusty*	FND	Findwhat	JAY	Jayde	NET	Netscape
AOL	AOL	DMOZ	DMOZ	GIG	Gigablast	KAN	Kanoodle	OVR	Overture
ABO	About	DOG	Dogpile*	GO	Go	LKS	Looksmart*	SCH	Search*
ALX	Alexa	ENT	EntireWeb	GOOG	Google*	LYC	Lycos	TEO	Teoma*
ALL	AlltheWeb	ESP	Espotting	HOT	Hotbot	MSN	MSN*	TES	Tessame
ALT	Altavista*	EUR	Euroseek	IWON	Iwon	MAM	Mamma	WBC	Webcrawler
ASK	Ask	EXS	ExactSeek	ICE	IceRocket	MTC	Metacrawler*	WIS	Wisenut
BUS	Business	EXT	Excite*	OXQ	Ixquick	MIV	Miva	YAH	Yahoo*

Figure 3. The forty search engines studied; the eleven used in the query analysis have asterisks

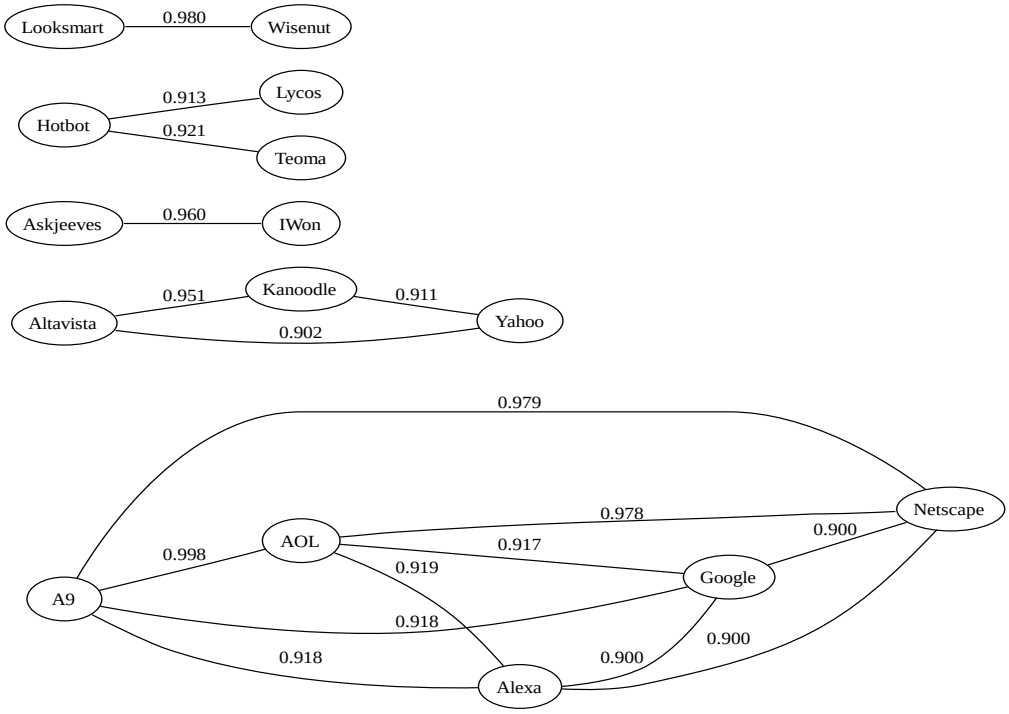

Figure 4. similarity of forty search engines as of Jan. 2006

Clusters 01/05/2006	A9, AOL, ALX, BUS, GOOG, ICE, NET, SCH
	ALL, ALT, KAN, YAH
	A9, AOL, ALX, ASK, BUS, GOOG, HOT, IWON, ICE, LYC, NET, SCH, TEO
	MTC, WBC
	ASK, DOG, HOT, IWON, LKS, LYC, MTC, TEO, WBC, WIS
	A9, AOL, ALX, BUS, GOOG, ICE, LKS, NET, SCH, WIS

Figure 5. Clusters of search engines for the 49 queries