Document Clustering using Small World Communities

Brant Chee
School of Library and Information Science
Bruce Schatz
Department of Medical Information Science
Institute for Genomic Biology, University of Illinois at Urbana-Champaign, 1206 W. Gregory, Urbana IL 61801 USA
http://www.beespace.uiuc.edu
Telephone: 1-217-244-0651
chee@uiuc.edu
schatz@uiuc.edu

ABSTRACT
Words in natural language documents exist as a small world network. Thus the extensive physics algorithms for extracting community structure are applicable. We present a novel method for semantically clustering a large collection of documents using small world communities. We combine modified physics algorithms with traditional information retrieval techniques. A term network is generated from the document collection, the terms are clustered into small world communities, the semantic term clusters are used to generate overlapping document clusters. The algorithm combines the speed of single link with the quality of complete link. Clustering takes place in nearly real-time and the results are judged to be coherent by expert users.

Categories and Subject Descriptors
H.3.3 *Information Search and Retrieval*; H3.1 *Content Analysis and Indexing*: J.3 *Life and Medical Sciences*

General Terms
Algorithms, Documentation, Languages.

Keywords
Document Clustering, Small Worlds, Scale-Free Networks, Semantic Clustering, Community Structures

1. INTRODUCTION
Document clustering groups similar documents using statistical computations on term frequencies. Ideally, related documents within the document collection are clustered. This resembles the formation of communities of interest in social networks. Many algorithms for clustering and segmentation have been discussed in the Computer Science and Information Science literatures. We present a novel approach to document clustering using small world algorithms to semantically segment a large collection of documents. Small world networks and the associated clustering algorithms have been gaining popularity in the physics literature [13-18,25]. This clustering technique is particularly helpful to an expert user who wants to define a new collection interactively, then view the subcollection using a fast clustering method.

Networks are often represented as graphs with nodes and edges, for social relationships and words in a language [7,9,15]. Many networks do not exhibit a random distribution of edges but instead small world behavior. Small world networks are graphs characterized by two features, small average path length and a large clustering coefficient. Small average path length means that the average distance between any two nodes is relatively short, often 2 or 3 edges. A large clustering coefficient means there is a high probability that for a randomly chosen person, any two acquaintances are also acquainted [18]. Scale free networks are graphs that have relatively few numbers of nodes with high connectivity or degree, and large numbers of nodes that have low connectivity or degree [16].

Researchers studying complex networks have emphasized identification of community structure [16,24]. Community structure is that the density of edges within groups is greater than should exist in a random network [4]. Community structure can be viewed as a form of clustering, where similar nodes are grouped together in some meaningful way.

Although many document clustering algorithms offer efficient segmentation for retrieval purposes and semantic segmentation for exploratory purposes, none take advantage of the community structure inherent within language. Research has shown that language exists as a small world network [6,7,25] and there are many small world clustering algorithms [17,18]. However, small world clustering of language graphs has not been applied to document collections for their semantic segmentation.

Our approach is a two-step approach: creating word clusters then mapping documents to the resulting word clusters. However it differs from other two-step algorithms such as [22] where word clusters are used to represent documents. Instead we create one lexical graph over all words in the document collection and map documents to the word clusters.

1.1 Related Work
The approach presented most resembles search result clustering. The goal of search result clustering is to provide a meaningful interpretation and organization of returning results from a search query. Previous works on search results clustering algorithms include Lingo, Fractionation, Suffix Tree Clustering, MSEEC, or SHOC [20]. Such search result clustering methods deal with small numbers of search results numbered in the hundreds with small amounts of text (known as snippets). We present an algorithm that deals with numbers of documents that are one to two orders of magnitude larger. Additionally, the amount of text is also much larger, usually several paragraphs as compared to a sentence or less than a sentence.

Like the search result clustering algorithm Lingo, we place emphasis on identity labels which are used to describe clusters. While many algorithms are used to group documents the defintion or "aboutness" of the cluster can be ill defined. Like Lingo, meaningful cluster labels are identified first through the term clustering then documents are mapped to the resulting labels in the *description-comes-first approach* [19].

This approach also differs from traditional search result clustering due to the fact that whole records are available and the system is combined with the search system. Most other algorithms act as

JCDL'07, June 17–23, 2007, Vancouver, British Columbia, Canada.

meta-layer where search results from other systems are clustered or aggregated then clustered.

As with many clustering algorithms, there is trade-off between speed and quality of resulting clusters. Clustering algorithm selection is dictated by the application of the clustered data as well as the type of input data [11] and intuitions about the characteristics of clusters thought to exist in the data. For example the Lingo algorithm is designed for small numbers of short documents (snippets), however we present an algorithm that deals with longer documents as well as several orders of magnitude more documents.

Many document clustering algorithms start with a term by document matrix A in which each term t in the document collection is represented by a row in the matrix and each document d is represented by a document. So the total size of the matrix is t x d and each element of the matrix a_{ij} represents the relationship between the ith term in the jth document. This relationship could be represented in various ways: binary (the existence of a term in a document); a count (the number of times term i occurs in document j); or a normalized value (eg. through tf/idf).

Lingo utilizes Singular Value Decomposition (SVD), which is one form of matrix factorization used to produce a low-dimensional basis for the column space of the term-document matrix. In Lingo each cluster label is a vector of the low-dimensional basis [19]. Our algorithm is more similar to link clustering methods where pair-wise similarity is calculated between all terms (Mutual Information in our system) and clusters are hierarchically grown by combining terms together into clusters based on either the two most similar terms (single link) or clusters of terms that have the smallest maximum pair-wise distance (complete link).

We present a modified physics algorithm, which approximates the speed of single link with the quality of complete link. Our method is unique in that little work has been done utilizing single-link clustering to cluster document collections [26]. The algorithm is similar to a method presented in the wireless ad hoc network literature where clusters are bounded size subtrees of a minimal spanning tree [5], thus helping to ameliorate tendencies toward chaining. The algorithm differs from traditional single link algorithms in that it takes into account not only distance in the form of edge weights, but also degree distribution of clusters.

The algorithm prefers joins between clusters with lower edge weight and low degree to ones with high edge weight and high degree. Often in small world networks, a large number of nodes have a degree of one and those nodes should join to the other node they are connected to. Relatively few nodes have large degree and these nodes are more representative of the overall graph structure. The problem with similar methods is the selection of "representative" nodes and determining the amount of constriction needed. In essence the algorithm clusters by simplifying or "coarsening" the graph similar to [27].

The running time of the algorithm presented here is faster than traditional single link and complete link algorithms ($O(n \log^2 n)$ vs. $O(n^2)$ and $O(n^2 \log n)$ respectively [12]). The algorithm has qualitative performance similar to betweenness centrality clustering in the physics literature, shown to discover community structure in social networks [9,16]. The methods presented here demonstrate a small world algorithm, with necessary modifications to normalize cluster sizes while maintaining semantic cohesiveness and fast running times.

2. SMALL WORLDS

Our novel approach to document clustering uses terms in a document collection to construct a small world (language) network where nodes are document terms (words or phrases) and edges between those nodes are weights based upon Mutual Information between terms. Small world clustering algorithms are performed on the resulting network and produce semantically related clusters of terms whose ranking is used to segment the document collections.

Small world graphs are a subset of graphs with special properties. In these, most pairs of nodes are connected through short paths and the graph demonstrates inherent clustering [15,16]. The same social networks form the basis of games like "Six Degrees of Kevin Bacon", where players try to relate any actor to Kevin Bacon via the movies he appeared in [15].

[23] formalize these properties through the use of two measures, l and C: l, the *characteristic path length*, is the average shortest geodesic distance between pairs of nodes in a graph. [25] defines l as

$$l = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i \geq j} d_{ij}$$

where d_{ij} is the geodesic distance from node i to node j.

C is the clustering coefficient and it measures "the cliquishness of a typical neighborhood (a local property) [25]." It is found that if vertex A is connected to vertex B and vertex B to vertex C, then there is a heightened probability that vertex A will also be connected to vertex C [18]. "In the language of social networks, the friend of your friend is likely also to be your friend. In terms of topology, transitivity [clustering] means the presence of a heightened number of triangles in the network." [25] defines clustering coefficient C in terms of local clustering coefficient C_v, the local clustering coefficient at a node v:

$$C_v = \frac{k_e}{k_v \frac{k_v - 1}{2}}$$

C_v represents the ratio of the actual number of edges divided by the maximum number of edges within v's neighborhood. Here, k_v is the number of neighbors of node n. If every neighbor is connected to every other neighbor, then $k_v(k_v-1)/2$ is the maximum number of edges that can exist. k_e is the actual number of edges. Thus the clustering coefficient for the network is the average of all local clustering coefficients:

$$C = \frac{1}{n} \sum_i C_i$$

A small world graph, is defined as one where $l \geq l_{rand}$ and $C \gg C_{rand}$, where l_{rand} and C_{rand} are the characteristics of a random graph with the same number of nodes and edges.

2.1 Small world community clustering

Small world graphs contain inherent community structure. While there is no consensus on the exact definition of what community structure is, a common heuristic is looking for clusters which have a higher density of intra-cluster edges than between them [7,8]. These heuristics can be framed in traditional clustering terms, where distance is viewed as edges, both the number of edges and their weight. Minimizing the number and weight of the edges

between clusters can be understood as maximizing the distance between communities or clusters.

Seminal work in [9], defined a measure of betweenness where "betweenness is some measure that favors edges that lie between communities and disfavors those that lie inside communities [17]." Derivatives of betweenness have been used by many small world clustering algorithms [17]. However, one of the originators of betweenness developed another measure named *modularity*, written Q to quantify the "goodness quality" of clusters [17].

Modularity measures "the fraction of the edges in the network that connect vertices of the same type (i.e. within-community edges) minus the expected value of the same quantity in a network with the same community division but random connections between the vertices" [17]. A community i is a sub-graph of a graph g composed of n nodes and e edges. Let e_{ii} represent the fraction of edges in i that fall inside of community i, and let e_{ij} be the fraction of edges connecting nodes in community i to nodes in community j. a_i represents the edges between a community i and all other communities to which it is connected. Modularity then, is the sum over all communities i.

$$Q = \sum_i (e_{ii} - a_i^2) \quad \text{where} \quad a_i = \sum_j e_{ij}$$

Q is between 0 and 1, where 0 indicates no community structure and is equivalent to a random graph and 1 indicates strong community structure. Empirically, [17] shows that small world networks generally have a Q between 0.3 and 0.7.

[9] presents a hierarchical agglomerative method for small world clustering. Instead of a betweenness measure to determine edges to remove, the algorithm focuses on optimizing the modularity Q. Completeing an exhaustive search over all possible divisions to find the optimal Q would take exponential time. Therefore, an approximate greedy optimization algorithm is used. Initially, all nodes are considered to be in their own community. The algorithm then iteratively joins clusters which cause the greatest increase in Q. This change is denoted below:

$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j)$$

This algorithm, like [9], deals with unweighted undirected graphs. However, [13] points out that it is relatively trivial to apply the same method to weighted graphs. The running time of the algorithm is $O((m+n)n)$ or $O(n^2)$ for a sparse graph.

[4] presents further refinement to the algorithm in [14]. The new algorithm makes use of sophisticated data structures as well as optimizations in calculating ΔQ. Here, Q becomes

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j)$$

where A_{ij} is an element of the adjacency matrix of an unweighted graph so that A_{ij} is 1 if i and j are connected and 0 if they are not. m is the number of edges in the graph, the δ function is 1 if $c_i = c_j$. k_i k_j $/2m$ represents the probability of an edge existing at random between c_i and c_j while respecting their degrees. The new algorithm updates and maintains a sparse matrix of ΔQ_{ij}s instead of an adjacency matrix and calculating ΔQ_{ij} at each step. The ΔQ_{ij} matrix only needs to keep track of communities linked by an edge.

The stated running time of the algorithm is $O(md \log n)$, where d is the depth of the dendrogram of joins performed by the algorithm. Assumptions are made that the graph is sparse where $m \sim n$ and $d \sim \log n$, leading to a running time of $O(n \log^2 n)$. The running time is non-polynomial as long as $d \ll m$, making it tractable for large networks. [4] demonstrates the algorithm on a large network of 400,000 nodes and 2 million edges.

3. ALGORITHMIC METHODS

Features of documents must be used to compare and contrast them. The selection of features is crucial to clustering. Here, we use only the text of documents to extract features, rather than structural components. We use the small world structure inherent in language to segment a collection of documents. The input to the clustering system is a collection of documents. The output is a segmented collection of documents, each representing a community collection. This process is illustrated in Figure 1.

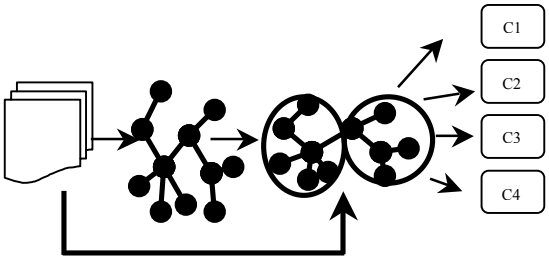

Figure 1 Documents are converted to a lexical graph, scored by mutual information. The documents are then clustered by term ranking.

Many words are polysemous, having different meanings or senses. "Mouse" for example, can refer to an input device on a computer, or it can refer to a small rodent. The way in which a word is used in a particular context denotes its intended sense. This idea is the premise of many word disambiguation systems. [8] coined an often used phrase, "you shall know the word by the company it keeps". This indicates that phrases or groups of words convey more information than do individual words. These small units of text are used to construct a lexical graph.

We will use a clustering algorithm to cluster the lexical graphs into clusters of phrases. After clustering is finished, the problem of mapping documents back to clusters remains. This has a traditional IR solution, using the terms resulting from clusters as queries to map the highest ranked documents to clusters.

3.1 Graph construction

We first run documents from the database through preprocessing tools. These tools are fairly standard and include noun phrase parsing, stop word elimination, and lexical stemming. The outcome is a reduced number of nodes and the close positioning of semantically related items. The documents are indexed after preprocessing is finished. This IR system, like many others, creates an inverted index comprised of phrases found in the document collection. Associated with that list of words are the documents that the words are found within plus the number of occurrences for particular words in a document. This step facilitates easy calculation of Mutual Information (MI) between words or word phrases.

MI is a way to quantify the relationship between any two terms, x and y. If words x and y are associated in some way or share some information, intuitively they should occur together at a more than random frequency. [3] defines $I(x,y)$ as:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

If two terms share information, then $I(x,y) > 0$. However, if x and y have no association between them (they are independent), $P(x,y) \approx P(x)\,P(y)$ so $I(x,y) \approx 0$. If we take $P(x)$ to be the probability of x occurring, this value can be calculated from an inverted index where $P(x)$ is the number of occurrences of x in a corpus divided by the number of terms in the corpus. Calculating $P(x,y)$ is more challenging. [3] suggests using a window of words w (terms), so that $P(x,y)$ is calculated by the number of times x and y occur in w throughout the corpus. By changing the window size, one can change the type of relationship found: "larger window sizes will highlight semantic concepts that hold over large scales [3]." Here, the size of w will be large, the size of a document.

3.2 Non scale-free clusters

Many small world clustering algorithms have demonstrated power law distributions in the size of resulting clusters, where there are few clusters containing many nodes and many clusters containing few nodes. [21] indicates that the distribution in cluster sizes does not necessarily have to follow a power law curve. Similarly, [1] show that such power law distributions are an artifact of the classification method. The observation that many clustering algorithms have demonstrated power law clustering distributions can be attributed to the fact that many algorithms are derivatives of the betweenness centrality algorithm in [17] or more recently modulation approximation clustering algorithm in [14].

A power law distribution in cluster size is detrimental in the case of document clustering. Through earlier experimentation with the modularity approximation clustering algorithm, it was apparent that it tends to group many items into few clusters (approximately 90% of the nodes were in one cluster), leaving many clusters with only one node. The number of nodes in a cluster appears to follow a power law distribution, as also reported by [4]. A large percentage of terms in one cluster is problematic for document clustering, as the clustered terms serve as an IR query used to map documents to term clusters. If one or several clusters contain most of the terms in the collection then many or all of the documents will map to them.

One approach to the large cluster problem would be to simply remove the large cluster and hope that there are enough smaller clusters of suitable size to segment the collection. Another approach is to hierarchically cluster the large cluster in order to break it into smaller pieces. However, due to the scale-free network graph, the resulting clusters or sub-clusters would also exhibit similar problems of uneven cluster size distribution.

3.3 Modularity approximation clustering

Modifications to the algorithm in [4] were necessary to prevent large clusters from forming. We present modifications to the weighted version of the modularity approximation clustering algorithm presented in [13]. However, the modifications make the following assumptions on the input graph: the graph has small world properties; the graph is scale free; node weights are normalized between zero and one; no parallel edges exist in the graph; and the average edge weight is $\ll 1$.

We selected the algorithm presented in [4] because of the stated running time, which is $O(m \log^2 n)$. However, the algorithm makes assumptions on the input graph, assuming $d \ll n$. The goal of the algorithm is to maximize *modularity*. As discussed, modularity is high if there is community structure in a network such that the fraction of edges falling within communities is greater than chance within a similar random network respecting the degree distribution and number of edges. [13] points out that, edge weights add information and suggests making the following algorithmic modifications to [4] to account for weight.

$$Q = \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

Whereas $2m$ was originally the number of edges in the graph, $2m$ instead represents the sum of the weights of the edges in the graph. Similarly, A_{ij} is still the adjacency matrix between communities i and j. However, it is changed to be the weighted adjacency matrix. The last change is to the value of k_i. Instead of representing the degree of a node i, k_i now represents the sum of the edge weights of the edges of i. This seems logical since the sum of a node's edges for an unweighted graph, each with weight 1, is equivalent to the degree of the node.

The $k_i k_j/2m$ term can be viewed as a discounting term that prevents the formation of one large cluster. We further modify this term to preference joins between communities, of which at least one has low order. Initially each node in a graph is considered a separate community and Q is calculated between adjoining communities. In our modifications to [13], k is changed to what it was originally, the degree of the node. Due to the constraint that all weights in the graph are between 0 and 1, the degree of a node can also be viewed as the edges of a node having maximal weight, so $k_i k_j/2m$ remains a probability.

We redefine k, where k' represents the unweighted version of k. Therefore, $k' \leq k$ and for the general case $k' \ll k$. So, $k_i k_j/2m$ is always greater than $k_i' k_j'/2m$, thus causing the ΔQ_{ij} to be smaller and the contribution of that join to be discounted. The change to the weighted algorithm discounts the value of a join between two communities. The MI graph consists of only single edges between nodes, $A_{ij} \leq 1$. For the common case, $A_{ij} \ll 1$ because A_{ij} is the MI between communities i and j (for our test data the average edge weight was between 0.001 and 0.004).

Joins between two communities with high connectivity are discounted, whereas communities with low connectivity and high MI are joined first. Since the contribution of a join is discounted, the largest increase in Q is reached earlier, causing smaller clusters to form. In the original equation, a join between two communities would never result in a negative ΔQ. However, in the modified equation it is possible for $k_i' k_j'/2m > A_{ij}$, thus also leading to smaller cluster sizes.

3.4 Term document mapping

If the problem of mapping documents to clusters of terms is viewed as an IR problem, then a cluster of terms is a query. The goal is to search for the documents that best answer the query. The index generated earlier to create the lexical graph will be used for the search process. Each cluster is viewed as a separate query, allowing for overlapping segments. This is intuitive, as one document can belong to many clusters. The top n documents are returned according to a ranking score.

We retrieve all documents with a score $t > 0$. In other words, each document has one or more query terms. The rank of each document d in the collection with at least one term t in a query q is ranked according to the following formula:

$$rank_{q,d} = coord_{q,d}\sum_{t}^{q} tf_{t,q} \cdot \frac{idf_t}{norm_q} \cdot tf_{t,d} \cdot \frac{idf_t}{|d|}$$

tf is the term frequency of a term *t* [24]. The ranking function sums the partial rank of a term over all terms in a query. The partial rank is composed of the term frequency of *t* in the query multiplied by the query normalizing factor, multiplied by the term frequency of *t* in *d*, multiplied by a document normalizing factor. This partial rank is summed over all *t* and multiplied by a coordination factor, *coord*, which puts preference on documents containing many query terms over those which contain fewer query terms, but a high score.

The coordination factor is the number of query terms *t* that are contained in *d* divided by the size of the query.

We then define a number *idf,* or inverse document frequency, as a score that discounts frequently occurring words. Words that have high frequency within a collection are not discriminating since such words return many documents in a search [24]. Less frequently occurring words will return more relevant documents and should thus be given more weight, *idf* normalizes this effect.

After all documents that contain 1 or more query terms are ranked, they are sorted according to that rank. If a cutoff is used, the top *n* documents are returned. To evaluate results, our implementation also displays the title of the top 50 documents.

4. EXPERIMENTAL RESULTS

The experimental paradigm for small worlds community clustering is to facilitate subject experts during interactive navigation. There is an ongoing problem with cluster evaluation techniques. Human judgment is hard to define numerically and can vary between individuals [21]. As illustrated by [21] it is hard to determine quantitative numbers for evaluating cluster assignments and any automated algorithm will be penalized for deviations from human assigned clusters no matter how justified the assignment. According to [21] user feedback surveys are the favored approach to estimating how useful an algorithm is.

We created two subsets of Medline relevant to specialized neuroscience research. These are generated by the queries and results in Table 1. The collection C1 is concerned with plasticity, the neural structures by which animals adapt to their environments such as by learning. The collection C2 is a specific research theme within the study of neural plasticity, concerned with the neuroanatomical structures relevant to behavioral changes signaled by certain chemicals. Collection C1 is the general theme for the Genomics of Neural and Behavioral Plasticity group at the Institute for Genomic Biology at the University of Illinois at Urbana-Champaign. Collection C2 is the specific theme of Neuroscience PhD dissertation within this group [10]. This student (Nyla Ismail) was the evaluation judge of the clustering quality of various algorithms on these collections.

The collections are indexed using Apache Lucene Java version 1.9.1 and stemmed using a Krovetz stemmer. The resulting index is used to generate the MI graph. The MI graph generation code is a multithreaded Java implementation utilizing a RAM based index to prevent IO contention between threads. The same MI graph is used in all clustering methods.

Table 1 Description of the test collections and the associated Medline searches used to create them.

Collection	Search Terms	Number of Abstracts	Number of Terms
C1	plasticity OR acetylcholine	81,746	267,981
C2	microarray OR muscarinic OR plasticity OR ((cholinergic OR noradrenergic) AND receptor)	74,533	285,623

Our clustering method is implemented in Java, utilizing a community type data structures consisting of a sparse hash map and binary search tree of modularity ΔQs, the value of a_x, and the nodes contained in the community. Tests were conducted on an eight processor AMD Opteron server with 32 gigabytes of RAM. Colt version 1.2.0, an open source library for high performance scientific and technical computing, is used to implement various data structures in the application like large resizable arrays and sparse hash maps. The virtual machine used was Sun's Java version 1.5.0_04. Graph clustering and document mapping were done separately. Maximum heap memory allocated was 1 GB for clustering and 3 GB for the mapping process. Parameters were set such that the input graph is repeatedly trimmed such that the graph was scale free and the average node degree was ~1.8. The graph was clustered and the minimal cluster membership size was set such that there was less than 40 resulting clusters.

We compare our method to three common algorithms: K-Means, Single and Complete Link (S-Link and C-Link, respectively) agglomerative hierarchical algorithms. These algorithms are implemented in CLUTO (http://glaros.dtc.umn.edu/gkhome/software) a freely available C++ based software package for clustering low- and high-dimensional data from the University of Minnesota. All parameters were left at their default and the only parameter that was changed was the number of clusters. Each algorithm was clustered at various sizes: 50, 100, 150, and 200. This was done so that the resulting K-Means clusters did not have many elements because the document mapping process would then map most documents to that cluster. Clusters with less than 4 elements or more than 50 elements were eliminated and the clustering which resulted in less than 40 clusters was chosen to be evaluated.

The Lingo algorithm was also going to be compared. However for the default maximum heap space of 1GB only 7,923 documents could be clustered. When 28GB was allocated to the heap, the plasticity collection could still not be clustered. This result was expected as the Lingo algorithm was not designed to cluster large numbers of longer documents. Internally the algorithm keeps a matrix of term by snippet, which can grow arbitrarily large with the number of documents. With the 28GB heap limit, approximately 40,000 articles could be clustered using the Lingo algorithm taking a time of 177.368s of wall clock time whereas our system could cluster the same number of documents in 29.805s of wall clock time, an improvement of over 5.95 times.

4.1 Clustering quantitative results

As stated earlier the algorithms we present were compared to three other common clustering algorithms, K-Means, single link, and complete link agglomerative algorithms. However, due to the propensity for the S-Link algorithm to form one large cluster and many clusters of only one element, the results of this algorithm were not evaluated. K-Means presented the fastest running times since it is an O(n) algorithm. Even though the MI generation step pre-calculates the distances, a complete link clustering algorithm is still order of $O(n^2 \log n)$. Table 3 lists the running times of the algorithms and the threshold used to evaluate their output.

Table 2 The threshold selected for each algorithm and its associated running time to cluster collections C1 and C2.

Collection	Algorithm	Threshold	Running Time (s)
C1	SW	N/A	40.54
	C-Link	50	214.106
	K-Means	200	11.581
C2	SW	N/A	47.35
	C-Link	100	198.147
	K-Means	200	5.538

The running time of our algorithm is very competitive. It is much slower than K-Means but more than 4 times faster than the Complete Link algorithm. Additionally, because multiple runs are not necessary, the time savings due to clustering via K-Means may be discounted due to the fact that the small world algorithm tunes parameters, including the number of clusters automatically. Thus, resulting in reducing the number of times K-Means or another algorithm needs to be re-run in order to search for the optimum number of clusters.

4.2 Clustering qualitative results

The two datasets discussed previously were used to qualitatively evaluate the results of the clustering algorithms. The goal of our system is to provide an efficient way to segment a document collection into interesting subtopics. Accordingly we created two metrics to evaluate the clustering results, coherence and utility. Each word cluster along with the top 50 documents from within the cluster were used to judge the utility and coherence of a cluster. Coherence and utility were judged on a three-point scale, 1-3, 3 for low coherence or utility and 1 for high coherence or utility.

Coherence can be thought of as how well related the words in a cluster are to one another, that the words in a cluster have a similar "aboutness." An example of a coherent cluster is: old, young, age, years, month, match. All of these terms except match can be thought of as having a topicality about time. While a cluster can be coherent, it does not necessarily mean that it is useful, the previous example illustrates this idea, while the cluster is coherent it is not useful because it does not segment the documents in a meaningful way. Ideally a cluster would be both coherent and have a high utility.

Utility can be thought of as how useful a cluster is regarding the subject material of the collection. An example cluster from C2 contained the following words: deprivation, deprived, monocular, visual, dominance, kitten, ocular, eye, binocular, retinal, sleep. This cluster was about anatomical plasticity, the visual experience dependent on plasticity. Of the top 50 documents in the cluster, 30 were judged to be relevant to the topic of the cluster and were considered to be useful. If more than 60% of the top 50 documents in a cluster are judged to be useful and related to the perceived topic of the cluster it is considered a 1, or of high utility. If 41-59% of the documents are relevant than the cluster has a utility score of 2, or medium utility, anything below 40% was considered poor. Utility is highly subjective and also application dependent, a cluster that can have low utility in one application could be considered to have a higher utility in another. For example, a cluster that consists of age ranges may have high utility in grouping articles that perform experiments on (human) subjects within those ranges but little utility in determining what the articles are about.

The datasets were clustered using our small world algorithm, K-Means, and single and complete link algorithms. The results of the three algorithms were then anonymized and presented to a doctoral candidate for evaluation. Table 3 describes the clusters formed by the various algorithms. It is apparent that the average number of documents per a cluster is quite large. This is due to the process of mapping documents into a cluster. No minimum threshold was set for the document mapping process and this may need to be explored in the future in order to reduce the average numbers of documents mapped into any given cluster and to increase the precision of the cluster. Currently if any term is found within a document, it is mapped to that cluster this leads to the problem of large cluster sizes when a term that occurs frequently in a collection is placed into a cluster.

Table 3 Description of the number of resulting clusters for each collection and associated clustering algorithm.

Collection	Algorithm	# of Clusters	Avg. # of Terms/ Cluster	Avg. # of Documents per Cluster
C1	SW	21	6	15,413
	C-Link	22	7	12,466
	K-Means	11	39	4,425
C2	SW	40	12	10,258
	C-Link	28	6	25,070
	K-Means	38	30	11,978

It is interesting to note that the SW words clusters have a high coherence and the large number of words do not adversely affect the number of documents put into a cluster. This is contrary to previous experiments with the small world algorithm where large word clusters lead to many documents being placed into the cluster.

Looking at Table 4, the small worlds clustering algorithm has higher coherence than the other algorithms on both collections.

The results of the utility metrics are encouraging; though not as markedly as for coherence. For collection C2, the utility of the small world algorithm is much better than the other two algorithms, however for collection C1 the small worlds algorithm performs similar to K-Means.

While it would be enticing to researchers to have a system that creates clusters of high utility, equally important are clusters of high coherence especially when culling search results for large

queries. Our system demonstrates noticeably higher coherence scores than K-means and Complete Link.

The evaluation of utility was related to the documents that were selected more than the utility of the word clusters. This may change somewhat with different ranking algorithms. Future work will need to explore the impact of different ranking algorithms as well as the link between evaluating utility and the way documents are placed into clusters.

It was noted by the judge that tuning algorithm parameters would vary utility measurements drastically. Increasing the recall often had a negative effect on the utility; for example in a larger collection tuning for recall might produce a cluster with month names which is of little utility for most applications. As stated previously the parameter tuning methods were done in such a way to be systematic and fair to all algorithms and also to reduce the often-exhaustive amount of time required to search for ideal parameters for a given algorithm.

The small world algorithm has two tunable parameters. The first parameter *S*, determines the minimum average degree of the input graph and the program will prune the input graph until this minimum average node degree is reached. This parameter can be seen as a fast approximation for how "small world" the graph will be. For these experiments it was set at the default of 3; the relationship between this parameter and precision and recall needs to be explored more. However, anecdotally increasing *S* appears to increase the number of clusters. The second direct parameter is the minimal number of terms *M* in a cluster, smaller numbers of terms can lead to clusters that are harder to comprehend and evaluate in terms of coherence and utility. Through the manipulation of these parameters along with a cutoff parameter for the document mapping process, one can change the resulting clusters and adjust the utility of them in order suit a particular task.

Table 4 Utility and coherence scores for each collection and clustering algorithm.

Collection			SW	C-Link	K-Means
C1	Utility	3	18	22	9
		2	1	0	1
		1	2	0	1
	Coherence	3	7	13	7
		2	6	5	3
		1	8	4	1
C2	Utility	3	37	28	38
		2	2	0	0
		1	1	0	0
	Coherence	3	9	18	38
		2	21	9	0
		1	10	1	0

5. CONCLUSIONS

We have demonstrated a novel method of small world graph clustering that provides a way to segment large document collections in fast running times. We have also demonstrated that the algorithm presented can be incorporated into a search system that enables the dynamic clustering of large numbers of search results. The word clusters generated from the small world graph exhibit semantic relatedness and topical coherence. They are particularly targeted towards domain experts interactively examining document clusters from special collections of community knowledge.

Our algorithm occupies a middle ground between speed and quality. Small Worlds is significantly faster than traditional high-quality Complete Link algorithms, and approaches the running time of the traditionally fast-speed K-Means algorithms. But on the other hand, Small Worlds is significantly better than traditional low-quality K-Means algorithms and offers comparable quality to traditional best Complete Link. Our algorithm produces good results in short time, and opens the possibility of interactive semantic clustering with high-quality results.

The modularity based clustering algorithm provides hard clusters of terms, only allowing a term to exist in one cluster. Arguably, this deficiency creates clusters that do not accurately model natural language given the existence of polysemous (a word having multiple meanings e.g. bank). However, this method provides disambiguation of word senses, clustering a term with others in the predominant sense that exists in a collection.

Our approach offers a way to ameliorate the problems associated with hard clustering. Because of the way documents are mapped to terms, they do not share the same limitations as the terms and can exist in more than one cluster. This is an important feature of this algorithm since documents can have many topics relevant to many queries.

The methods we present offer a fast way to partition documents into semantically related clusters in both homogenous and heterogeneous scientific literature collections. See Figure 2 for sample session using the dynamic clusterer within the BeeSpace System (http://beespace.igb.uiuc.edu:8080/BeeSpace/).

There are several areas of work that could be elaborated on in the future. The first is exploring the space of parameters that can be tuned; the second is comparing results to different algorithms; the third is comparing across domains and data types; and the fourth is creating an efficient, updatable model.

The datasets we used are not exhaustive. It would be interesting to perform similar experiments on different size collections, including some that are several orders of magnitude larger and smaller in document number, to determine if the documents can be effectively clustered. This could help illuminate the limitations surrounding graph size. Experiments on different genres of text including web documents, news articles, and email collections (colloquial text), could also prove valuable.

6. ACKNOWLEDGEMENTS

This work is based upon a M.S. Thesis submitted (Chee) in December 2005 to the Graduate School of Library and Information Science at the University of Illinois at Urbana-Champaign. The document clustering system is part of the BeeSpace project (Schatz), which is supported by NSF FIBR-BIO grant 0425852. We thank Nyla Ismail for judging the results of the clustering and preparing the collections used in this publication. We thank the members of BeeSpace for helpful discussions. In particular, the lexical graph with mutual information was generated using software written by lead programmer Todd Littell who helped with extracting the databases and processing the experiments. ChengXiang Zhai, the faculty lead for algorithms development, provided significant

useful advice. Azadeh Shakery shared her results on small world algorithms for dynamic document queries. Moushumi Sen Sarma evaluated early experimental results. Nigel Goldenfeld interfaced with the physics small worlds community, providing helpful references and discussions.

7. REFERENCES

[1] Caldarelli, G., Cartozo, C.C., De Los Rios, P. and Servedio, V.D.P. Widespread occurrence of the inverse square distribution in social sciences and taxonomy. *Phys. Rev. E*, *69* (3), 35101-1-3, 2004.

[2] Chung, Y., He, Q., Powell, K. and Schatz, B. Semantic indexing for a complete subject discipline. in *Proc. of the Fourth ACM Conference on Digital Libraries*, (Berkeley, CA, 1999), ACM Press, 39-48.

[3] Church, K. W. and Hanks, P. Word association norms, mutual information, and lexicography. in *Proc. of the 27th Annual Conference of the Association of Computational Linguistics*, (Vancouver, B.C., 1989), ACM Press, 76-83.

[4] Clauset, A., Newman, M. E. J., and Moore, C. Finding community structure in very large networks. *Phys. Rev. E, 70* (6), 066111, 2004.

[5] Fernandess, Y. and Malkhi, D. K-clustering in wireless ad hoc networks. in *Proc. of the second ACM international workshop on Principles of mobile computing* (Toulouse, France, 2002), ACM Press, 31-37.

[6] Ferrer i Cancho, R. and Solé, R. V. Patterns in syntactic dependency networks. *Phys. Rev. E*, *69* (5), 051915, 2004.

[7] Ferrer i Cancho, R. and Solé, R. V. The small world of human language. *Proc. R. Soc. Lond. B*, *268* (1482), 2261-2265, 2001.

[8] Firth, J. R. A synopsis of linguistic theory, 1930-1955. in Palmer F. R. ed. *Selected Papers of J. R. Firth 1952-59*, Ed., Longmans, London, 1968, 168-205.

[9] Girvan, M. and Newman, M. E. J. Community structure in social and biological networks. *PNAS*, *99* (12), 7821-7826, 2002.

[10] Ismail, N., Robinson, G. E, and Fahrbach S. E. Stimulation of muscarinic receptors mimics experience-dependent plasticity in the honey bee brain. *PNAS, 103* (1), 207-211, 2006.

[11] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: A Review. *ACM Computing Surveys*, 31 (3), 265-323, 1999.

[12] Manning, C. D. and Schutze, H. Foundations of statistical natural language processing. *MIT Press*, Cambridge, MA, 1999.

[13] Newman, M. E. J. Analysis of weighted networks. *Phys. Rev. E*, *70*, 056131, 2004.

[14] Newman, M. E. J. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, *69* (6), 066133, 2004.

[15] Newma, M. E. J. Models of the small world: A review. *J. Stat. Phys.*, *101* (3-4), 819-841, 2000.

[16] Newman, M. E. J. The structure and function of complex networks. *SIAM Rev.*, *45* (2) 167-256, 2003.

[17] Newman, M. E. J. and Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E*, *69* (2), 026113, 2004.

[18] Newman, M. E. J., Watts, D. J., and Strogatz, S. H. Random graph models of social networks. *PNAS*, *99*, Suppl. 1, Feb., 2566-2572, 2002.

[19] Osiński, S. Improving quality of search results clustering with approximate matrix factorizations. In *Proc. of the 28th European Conference on IR Research* (ECIR 2006), (London, UK, 2006), 167-178.

[20] Osiński, S. and Weiss, Dawid. Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data. In *Proc. of the International IIS: IIPWM'04 Conference*, (Zakopane, Poland, 2004), 369—378.

[21] Palla, G., Derényi, I., Farkas, I., and Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435* (Jun.), 814-818, 2005.

[22] Slonim, N. and Tishby, N. Document clustering using word clusters via the information bottleneck method. In *Proc. of the 23rd Annual Internation ACM/SIGIR Conf. on Research and Development in Information Retrieval*, (Athens, Greece, 2000), ACM Press, 208-215.

[23] Solé, R., Ferrer-Cancho, R., Montoya, J. M., and Valverde, S. Selection, tinkering, and emergence in complex networks. *Complexity*, *8* (1), 20-33, 2003.

[24] Sparck Jones, K. and Willett, P. Readings in information retrieval. Morgan Kaufman, San Francisco, 1997.

[25] Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world' networks. *Nature*, *393* (6684), 440-442, 1998.

[26] Willet, P. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24 (5), 577-597, 1988.

[27] Wu, A. Y., Garland, M., and Han, J. Mining scale-free networks using geodesic clustering. In *Proc. 10th ACM SIGKDD Int. Conf.*, (Seattle, WA, 2004), ACM Press,719-724.

SMALL WORLD WORD CLUSTERS FROM COMMUNITY COLLECTION C2

Cluster 0: hepatocellular hcc carcinoma squamous metastasi adenocarcinoma breast hepatic tissue metastatic

Cluster 1: m4 m5 subtype m1 m3 m2 selective 116 af dx 4-damp otenzepad

Cluster 2: binding bound site ligand high radioligand quinuclidinyl benzilate qnb bmax 3h-qnb competition

Cluster 3: urinary detrusor bladder incontinence void micturition overact oxybutynin tolterodine benzhydryl phenylpropanolamine cresol

Cluster 4: task discrimination training retention amnesia acquisition memory working performance drink consolidate

Cluster 5: pathway transduction signal mitogen erk mapk kinase phosphorylation p38 cascade map staurosporine alkaline

Cluster 6: stores deplete thapsigargin intracellular influx ca2 mobilization fura-2 extracellular calcium cytosolic entry free

Cluster 7: deprivation deprived monocular visual dominance kitten ocular eye binocular retinal vision sleep

Cluster 8: geniculate lateral nucleus medial forebrain basali meynert magnocellulari nbm acetyltransferase choline chat acetylcholinesterase

Cluster 9: alpha-2 alpha-1 adrenergic idazoxan 1-adrenergic 2-adrenergic phenylephrine prazosin yohimbine clonidine 2-adrenoceptors

Cluster 10: mk-801 maleate dizocilpine methyl aspartate nmdar nmda 2-amino-5-phosphonovalerate amino methylaspartate nr2a nr1 nr2b

Cluster 11: 5-ht3 ondansetron 5-ht4 serotonin 5-ht2 5-ht ketanserin 5-ht1 5-hydroxytryptamine serotonergic 5-ht2a serotoninergic

Cluster 12: app precursor amyloid alzheimer abeta plaque neurofibrillary tangle ad disease dementia senile neurodegenerative cognitive

Cluster 13: clinical trial symptom patient outcome diagnosis therapy therapeutic improve healthy

Cluster 14: interferon ifn gamma cytokine necrosi chemokine chemoattractant proinflammatory interleukin il-6

Cluster 15: amitriptyline tricycle antidepressant imipramine antidepressive reuptake agent drug mianserin antineoplastic

Cluster 16: myocardial infarction myocardium cardiac ventricular myocyte atrial inotropic atria chronotropic negative

Cluster 17: nachr alpha7 nicotine acetylcholine tubocurarine bungarotoxin bgt 125i-alpha btx machr iodine

Cluster 18: old month age young adult postnatal neonate developmental develop stage early match maturation birth period decade

Cluster 19: methylpiperidine methiodide 4-diphenylacetoxy dimethylpiperidinium 4-diphenylacetoxy-1,1 piperidine pirenzepine

Cluster 20: thymu thyme myasthenia thymoma weakness immunosuppressive tit sera autoantibody albumin anti antibody immunoglobulin

Cluster 21: nos1 nno nitric monomethyl arginine conotoxin nitro name ng nitroarginine oxide ester nos stress donor

Cluster 22: superior scg cervical ganglion ganglia autonomy preganglionic sciatic ending inferior

Cluster 23: pulmonary lung airway bronchoconstriction asthma asthmatic bronchial copd inhale hyperresponsive bromide

Cluster 24: kg microg saline mg 0.5 intraperitoneal administer iii ip administration intravenous dose microgram action pretreatment

Cluster 25: opioid endorphin mu mephe methionine g ala naloxone opiate morphine narcotic analgesia nf

Cluster 26: microarray comprehensive spotted algorithm detection bioinformatic data signature dataset discovery software

Cluster 27: 3-thiotriphosphate gtpgamma guanosine thionucleotide triphosphate nucleotide guanine gtp gdp gi galpha gq-g11 thio couple

Cluster 28: ins p3 1,4,5 1,4,5-trisphosphate inositol ip3 trisphosphate phosphate accumulation inositol-1,4,5-triphosphate phosphoinositide hydrolysi phospholipase pi 4,5-bisphosphate turnover insp3 myo breakdown plc

Cluster 29: bradycardia tachycardia heart cardiovascular arterial blood pressure press rate mean mmhg barrier beat diastolic

Cluster 30: salivary submandibular gland saliva parotid acinar amylase acini gastric fundu mucosa ulcer

Cluster 31: 1,2-bis acetoxymethyl ethane bapta chelator 2-aminophenoxy chelate tetraacetic egtazic egta

Cluster 32: vasodilator vasodilation vascular endothelium vessel artery vegf relaxing denude aorta vasoconstriction coronary

Cluster 33: contract relax relaxation smooth strip contractile contraction force isometric tension muscle tracheal relaxant circular

Cluster 34: nt-3 neurotrophin-3 neurotrophic neurotrophin bdnf trkb derive trk trka ngf p75

Cluster 35: spinal supraspinal descend cord horn dorsal intrathecal injury pain motoneuron root traumatic allodynia antinociceptive

Cluster 36: 8-bromo monophosphate cyclic adenosine camp amp creb pka forskolin adenylate adenylyl cyclase isoproterenol gmp cgmp guanylate element iso isoprenaline dibutyryl 1-methyl-3-isobutylxanthine

Cluster 37: 38393 skf benzazepine 2,3,4,5-tetrahydro-7,8 dihydroxy-1-phenyl-1h-3 d1 benzazepin-7-ol 2,3,4,5-tetrahydro-8-chloro-3-methyl-5-phenyl-1h-3 d2 sch 23390 dopamine sulpiride quinpirole sch23390 spiperone amphetamine

Cluster 38: diabetic mellitu diabetes streptozotocin insulin glucose islet pancreatic igf like pancreas exocrine 2-deoxyglucose

Cluster 39: patch pipette clamp technique whole current recording hyperpolarize mv conduct voltage outward inward depolarize

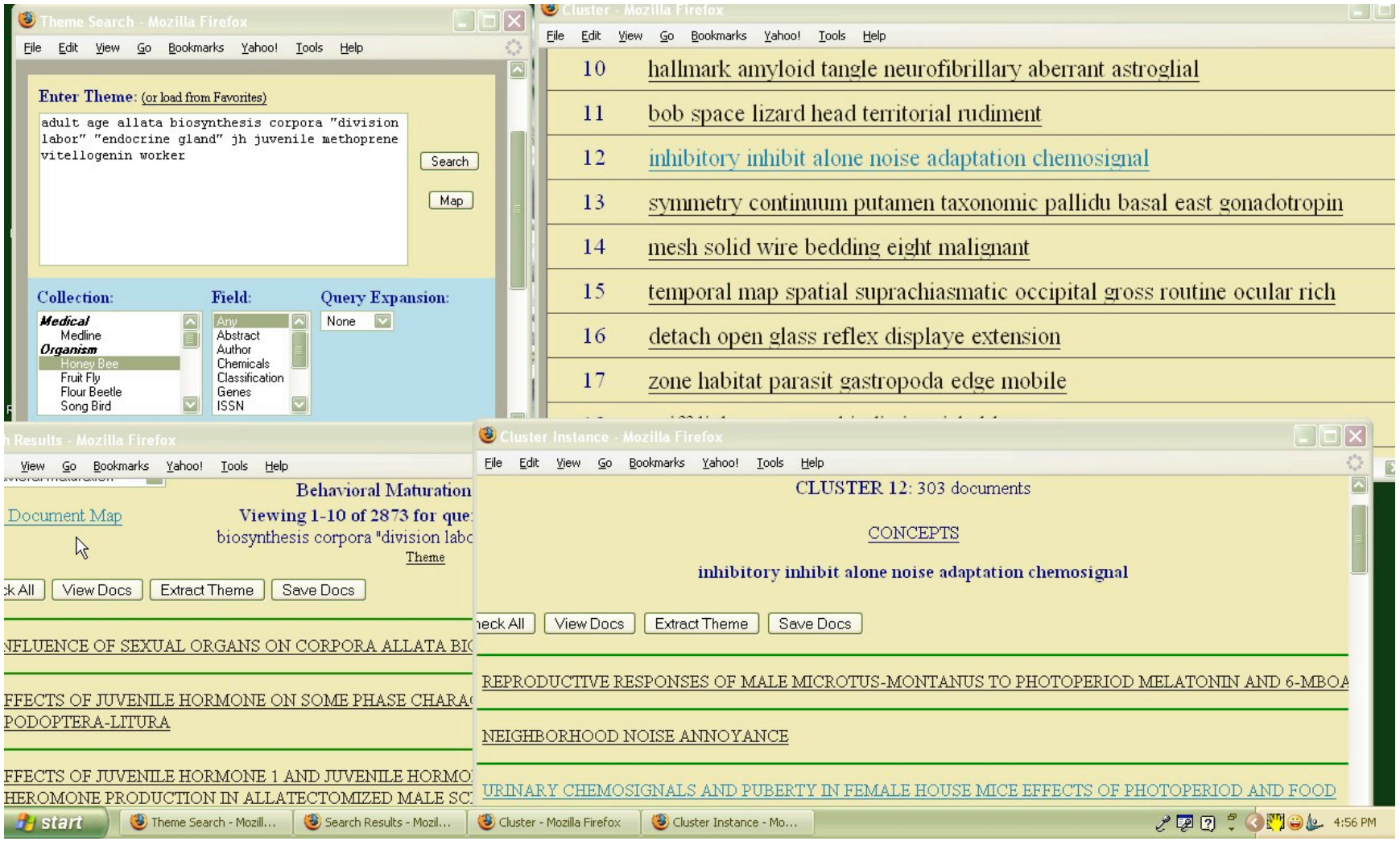

Figure 2: A sample session with the dynamic clusterer in BeeSpace v3. A biologist is interested in comparing behavioral maturation in insects versus mammals (behaviors that change as the animal ages). She has generated a master theme of phrases relevant to "juvenile hormone" by searching Medline and issuing automatic theme extraction. Loading this theme, she searches the Honey Bee collection (upper left screen). The results (not shown) verify that the theme finds relevant documents in insects. She next switches the theme terms into the special collection for Behavioral Maturation retrieving nearly 3000 documents (lower left). This is too many to examine carefully, so she issues Document Map to invoke the Dynamic Clusterer on the new special collection. Several coherent clusters are generated (52 in all; the upper right screen shows a segment). She selects a relevant cluster and examines the list of 300 documents found in this cluster (lower right). A relevant document is found on puberty effects of food deprivation in female mice. This suggests an experiment on whether juvenile hormone affecting "puberty" in bees also affects mice similarly. The dynamic clusterer is essential to making the conceptual navigation interactive, since it can be effectively invoked on regions of thousands of documents to limit focus to regions of hundreds of documents that can then be individually examined.