Running Documentation of Work on SBIR Contract HHSN268200900053C
Accuracy of Models for Mapping the Medical Sciences

Schedule and Tasks

by Kevin W. Boyack

The proposed schedule and tasks from the original proposal have been clarified and broken into subtasks as shown in Figure 1. The task list shown here will be used to document progress on this SBIR Phase I project.

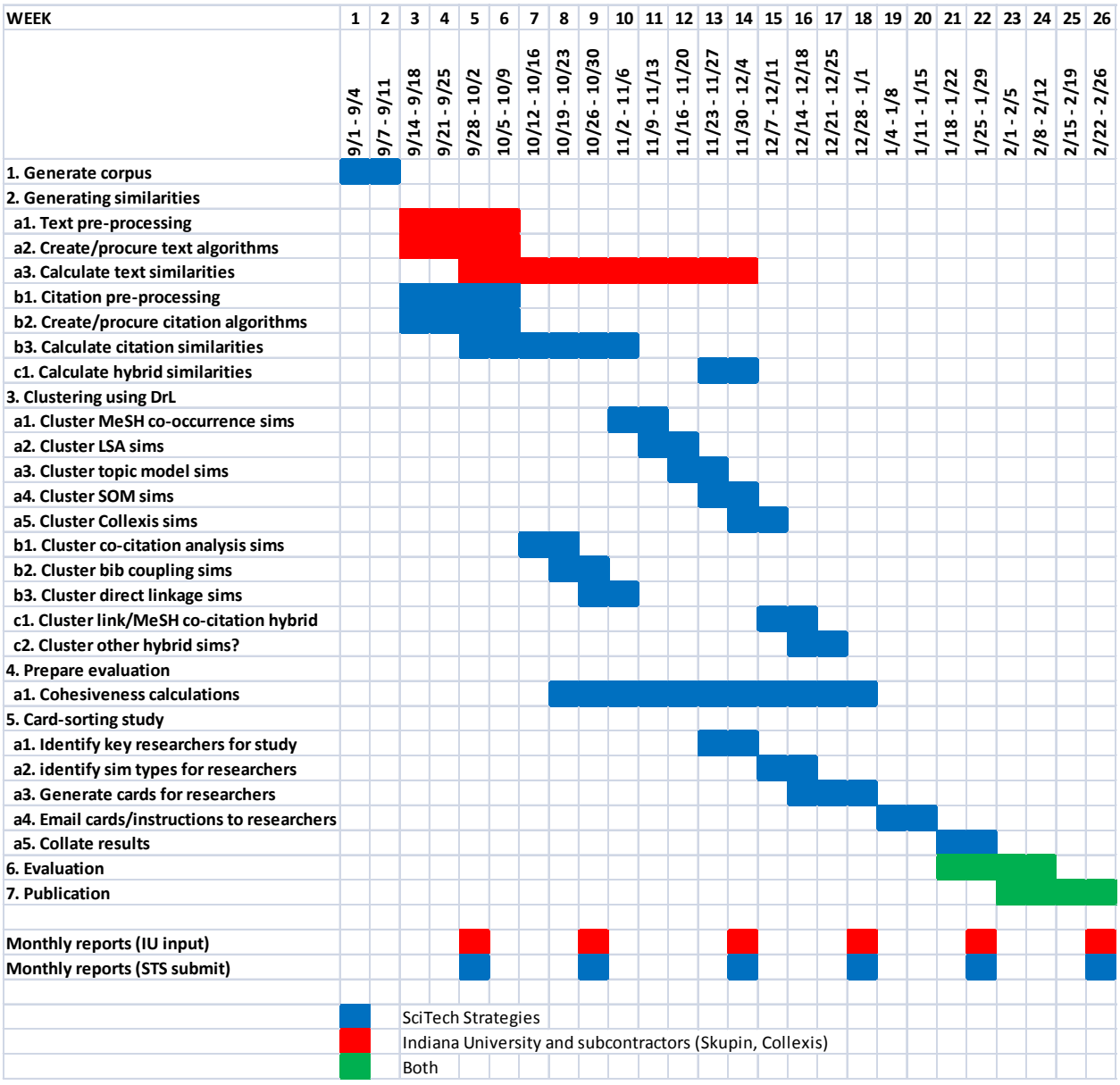

Figure 1. Tasks and proposed schedule for this SBIR Phase 1 project.

It was decided that all work will be documented in real time and at a level of detail that supports the exact replication of work. All subcontractors will have access to this documentation as well as to intermediate data results. While the Scopus data cannot be made available, all MEDLINE based derivative data will be made freely available from a web page linked from http://sci.slis.indiana.edu/sts, see Appendix A. Ultimately, this will facilitate easy replication as well as comparisons with algorithms that might be developed in the future. We will work with TREC (http://trec.nist.gov) and KDD (http://www.sigkdd.org) organizers to ensure that the MEDLINE portion of this dataset is known and available for algorithm comparisons.

Task 1: Generation of a Corpus of Medical Documents
by Kevin W. Boyack

The ultimate purpose of this project is to provide a highly accurate interactive map of medical research that can be easily used by both technical and non-technical users. Most current science maps in use today are small in scale and have not been validated. Accurate decisions require high quality and high coverage data, well defined and tested data analysis workflows, and a resulting representation that matches the visual perception and cognitive processing capabilities of human users. Phase I of this project has been designed to compare and determine the relative accuracies of maps of medical research based on commonly used text-based and citation-based similarity measures at a scale of over two million documents.

In order to properly compare text-based and citation-based techniques, a large corpus for which both text and citations are available is required. Task 1 of this project has produced such a corpus. SciTech Strategies, Inc. (STS) has a license to use raw data from the Elsevier citation database known as Scopus, a database that purports to index all of MEDLINE. In addition, Indiana University maintains a copy of the MEDLINE database in its Scholarly Database (SDB, online at http://sdb.slis.indiana.edu and led by Nianli Ma). Thus, for purposes of this project, the participants have access to the necessary data to generate the large corpus necessary for this work. This corpus was generated using the following process.

1) Records from the SDB version of MEDLINE were matched to Scopus records maintained in a database at STS to generate a one-to-one matching between records. The purpose of this matching step is to identify those records (articles, etc.) for which MeSH terms, titles, abstracts, and references are available. This matching was carried out on a segment of both databases using publications from 2003 to 2008 to ensure that at least 2 million records would be matched. The steps used in this matching process, along with the results, are detailed here.

a) Over the past couple of years, STS has maintained a matched list of MEDLINE and Scopus journals. This list was used to add the Scopus journal ID number to 99.8% of the MEDLINE records (corresponding to 7611 different MEDLINE journal abbreviations) from 2003 to 2008.
b) A sequence of steps using different matching criteria was then used to match MEDLINE and Scopus article data. Matching was done without replacement, meaning that if a particular MEDLINE record was matched in one step, it was removed from the list and not available for matching in a subsequent step. The matching criteria, in order, were:
1 – Journal ID AND starting page AND (volume OR pubyear) AND soundex(title)

2 – Journal ID AND volume AND soundex(title)
3 – Journal ID AND pubyear AND soundex(title)
4 – Journal ID AND soundex(title)
5 – Given that each of the above matching steps (1-4) generated some duplicate matches to PMIDs (PubMed ID), the matched set was restricted to unique PMID-ScopusID matches (meaning each PMID and ScopusID could only appear once in the full list).
6 – For any duplicate matches, matches where the first five initials of the first author's last name did not match were removed.
7 – All remaining unmatched PMID were left as unmatched. Results of these matching steps are given in Table 1.

Table 1. Efficiency of matching MEDLINE to Scopus records.

Step	Counts	Fraction	Total matched
0 – initial MEDLINE records	3,647,481		
1 – matching criteria 1	2,847,197	78.06%	2,847,197
2 – matching criteria 2	557,991	15.30%	3,405,188
3 – matching criteria 3	91,985	2.52%	3,497,173
4 – matching criteria 4	2,676	0.07%	3,499,849
5-7 – remove duplicate/false matches	3,475,573	95.29%	

Note that the "soundex" function mentioned above is a function in MySQL that strips all non-alphanumeric characters from text, and converts the remaining text to a string based on phonetics. Two strings that sound the same, but that have different spellings, can thus have the same soundex value. Use of the soundex function allows us to match some records where there are simple misspellings or punctuation differences in the article titles between the two databases.

As shown in Table 1, the overall matching rate (unique PMID to ScopusID) for the entire set of MEDLINE documents from 2003 to 2008 was 95.3%. The matching rate for 2008 (92%) is lower than for previous years (over 96%) because the full 2008 data were not yet available in Scopus.

2) Additional data were added to the matched data: numbers of references from Scopus, numbers of MeSH terms from MEDLINE, and the existence of a MEDLINE abstract. These data were then used to limit the set of records to those with sufficient text and citation information to form an appropriate corpus for this study. It was clear from the numbers that all 6 years (2003 to 2008) of records were not needed to give a set of around two million documents. We thus restricted the set to five years (2004 to 2008). Numbers of documents by year, using different limitations, are given in Table 2.

Table 2. Numbers of records by year using different limitations.

Year	MEDLINE	in Scopus	w/ Abstr	w/ >=5 Refs	w/ Abstr OR >=5 Refs	w/ Abstr + >=5 Refs	Final
2004	575,938	553,743	454,023	436,421	489,545	400,899	389,353
2005	603,166	579,359	480,477	469,777	517,773	432,481	420,059
2006	626,895	605,734	504,747	498,328	547,663	455,412	442,743

2007	644,457	620,386	523,805	520,196	566,781	477,220	464,479
2008	650,069	597,839	506,852	490,034	547,110	449,776	437,135
Total	3,100,525	2,957,061	2,469,504	2,414,756	2,668,872	2,215,788	2,153,769

Several observations can be made from Table 2. First, only 80-85% of the MEDLINE records in Scopus actually have references. Scopus has been unable to obtain reference information from publishers for the other 15% or so of MEDLINE records. Second, not all records have abstracts. This is no fault of either MEDLINE or Scopus, but simply reflects the data that is supplied to them by publishers.

3) The final decision on which records to keep in the corpus was determined after examination of the distributions of numbers of references and numbers of MeSH terms, which are shown in Figure 2. It was decided that only records with abstracts, at least 5 references and at least 5 MeSH terms would be kept. This ensures that each record has sufficient numbers of references and MeSH terms for generation of article-to-article similarities using both text- and citation-based similarity calculations. In addition, there were several hundred articles with very large numbers of references. In our experience, articles with large numbers of references can lead to over-aggregation of citation clusters. Thus, we arbitrarily set a threshold of a maximum of 400 references. Any article with more than 400 references was excluded from the corpus. The final numbers of articles by year that met these criteria are listed in the final column of Table 2.

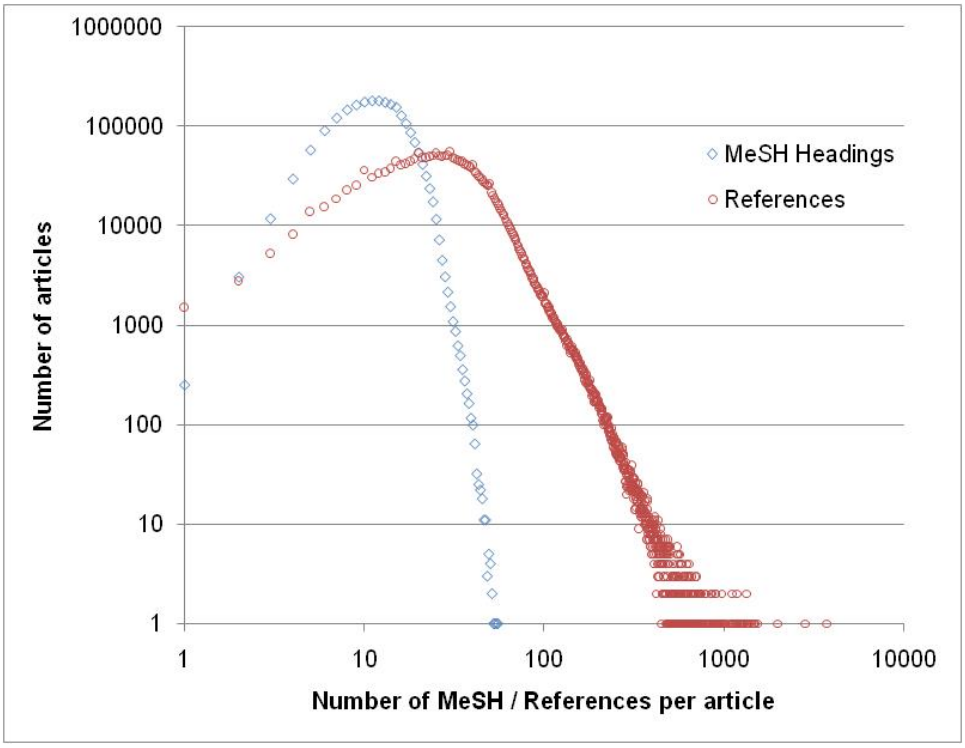

Figure 2. Distributions of numbers of MeSH terms and numbers of references per record.

The corpus identified for this project thus consists of 2,153,769 unique MEDLINE records to which we have added the references from the Scopus database, see Appendix A. This corpus

represents the most complete multi-year set of medical sciences literature for which both MeSH terms and references exists that can be identified, and will be used for all subsequent tasks in this project.

Task 2: Generation of Document-Document Similarities

This task comprises the pre-processing and algorithm development necessary to generate similarity files for each of the text- and citation-based measures to be tested in this project. The ultimate output of Task 2 is a similarity file for each measure that contains lists of the strongest pairwise similarities for each article in the corpus. Each similarity file may contain on the order of 20,000,000 – 30,000,000 triples – where the triples are paper_id1, paper_id2, sim_value.

2.a1: Text Pre-processing
by Russell J. Duhon and Katy Börner

Katy Börner's team at Indiana University has extracted two different types of term-document matrices from the MEDLINE corpus: a MeSH-document matrix and a word-document matrix. Words for the latter were parsed from the titles and abstracts of the documents. These two matrices are provided as input to six different text-based approaches to similarity, see section 2.a2. The following describes the pre-processing that was required to generate the MeSH-document and word-document matrices.

2.a1.1: MeSH Pre-processing

PMIDs and associated MeSH terms (without qualifiers) were extracted from the SDB version of MEDLINE for all documents in the corpus. Whitespace was stripped off each end of each term, and all leading "*" characters were also stripped. No tokenization of MeSH terms was required because they are all standardized indexing terms. The numbers of articles and fraction of articles in the corpus associated with each MeSH term were then computed (see Figure 3, top) to determine if any thresholding of terms would be needed.

MeSH terms were limited in the following two ways:

- All terms that are not Class 1 descriptors per the 2009 MeSH data (http://mbr.nlm.nih.gov/Download/2009/MeSH/README) were removed from the set. This had the effect of removing the Class 3 (Check Tags) and Class 4 (Geographical Locations) terms, which have little or nothing to do with content.
- To maintain consistency with the references data, one of the natural thresholds associated with the references data (explained below) was adopted. All MeSH terms that were listed for fewer than 4 documents were removed from the set. The upper end of the distribution was left intact because many of the researchers on this contract felt that those terms occurring in many documents (e.g., the MeSH term "Humans" is indexed for 66% of the documents) would be useful to the calculation of similarities. The final distribution of MeSH terms thus identified is shown in Figure 3, bottom.

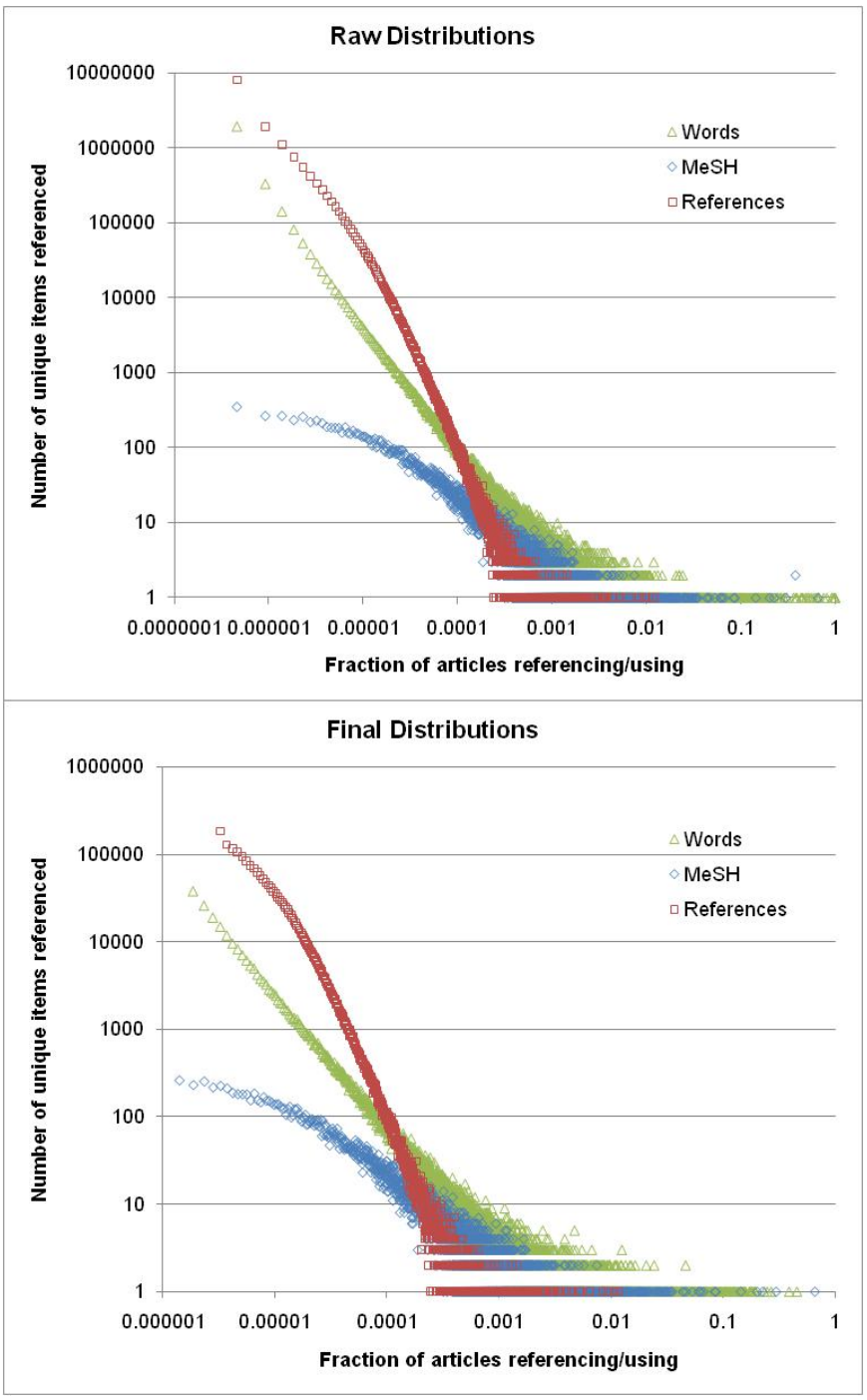

Figure 3. Initial/raw and final (after thresholding) distributions of title/abstract words, MeSH terms, and cited references.

The final MeSH-document matrix is based on the MeSH term list associated with the final distribution in Figure 3, and has been made available for generation of similarities to all subcontractors. The number of unique MeSH terms in the final MeSH-document matrix is thus 23,347 with a total number of 25,901,212 MeSH terms occurring in all 2,153,769 documents, see Appendix A.

2.a1.2: Title/Abstract (TA) Pre-processing

PMIDs and associated titles and abstracts were extracted from the SDB version of MEDLINE for all documents in the corpus, and made available for stemming, tokenization, and stopwording. An initial distribution of words, similar to the MeSH term distribution mentioned above, is shown in Figure 3, top.

After reviewing the functionality of the standard stemmer and tokenizer in the Natural Language Toolkit (NLTK), a standard natural language processing library, the stemmer and tokenizer were replaced with a simplified procedure that produces identical output (except as described below), since most punctuation is discarded. The simplified procedure is given here.

All punctuation characters except ' were removed from the text, and replaced with a single space each. The resulting text was converted to all lower case, then split on whitespace, leaving only tokens with no whitespace in them, and no empty tokens. Then, in each token, contractions ending in 'll, 're, 've, 's, 'd, or n't were separated into a root and a contraction, and the contraction portions were removed (all of those contraction suffixes are forms of words on the stopword list or possessive forms of other words). All apostrophes were then removed from the token. All tokens appearing on the stopword list, see Appendix A (which is a combination of an official MEDLINE stopword list of 132 words, and a second list of 300+ words commonly used at NIH and provided by David Newman, UC Irvine) were then removed, as were tokens consisting of a sequence of digits.

Tokens were then further limited using the same methodology applied to MeSH terms. Thus, all tokens that were listed for fewer than 4 documents were removed from the set. The final distribution of tokens from titles and abstracts thus identified is shown in Figure 3, bottom, and has been made available to all project researchers for generation of similarities. The number of unique tokens in the final word-document matrix is thus 272,926 with a total number of 175,412,213 tokens occurring in the 2,153,769 documents. The total number of individual token occurrences is 277,008,604. Thus, there are on average 128.6 individual title/abstract-based token occurrences associated with each document.

2.a2: Creation/Procurement of Text Algorithms

The (23,347 x 2,153,769) MeSH-document matrix and the (272,926 x 2,153,769) title/abstract (TA) word-document matrix are provided as adjacency lists as input to different text-based approaches to similarity. The five different approaches, along with associated collaborators/subcontractors, are listed here:

- Co-word analysis (Co-word), also known by labels such as term co-occurrence—to be performed by Russell Duhon, IUB.
- Latent Semantic Analysis (LSA)—to be performed by Russell Duhon, IUB.
- Topic modeling algorithms based on Latent Dirichlet Allocation (Topics)—to be performed by Russell Duhon, IUB and separately by Dave Newman, funded by NIH/NINDS.

- Self-Organizing Maps (SOM)—Subcontract to Andre Skupin, SDSU.
- Collexis engine—Subcontract to Bob Schijvenaars and Aaron Sorensen, Collexis.

Efficient code for calculating the sets of 2 trillion+ cosine similarities and retaining the top few documents most similar to a given document was written specificially for this project. The code used to compute the MeSH similarities was modified to compute the TA similarities, because the TA data could not fit in the available memory if stored in the same fashion as were the MeSH data.

Commonly available Latent Semantic Analysis algorithms were finding the task intractable, and after some searching an algorithm was found that uses the Generalized Hebbian Algorithm (available at http://www.dcs.shef.ac.uk/~genevieve). The custom cosine similarity code from above was once again altered to run efficiently given the much higher density of LSA output.

Dave Newman likewise prepared custom similarity calculation code with some minor assistance from the group at IUB related to running it on available computing resources.

The work described above created significant capabilities for running similarities of very large data sets with a range of densities. The time to run even the largest data set (the TA similarities) with the most complex similarity calculation took just five calendar days from start to finish with over 310 days of processing time on IU's Quarry supercomputer. That is about 75,000 similarities per second per process, or about 4.5 million similarities per second across all the processes running in parallel.

2.a3: Calculation of Text-based Similarities

This section describes how the different approaches are used to calculate similarity files for the MeSH-document matrix and the term-document matrix. Each similarity file contains triples of paper_id1, paper_id2, sim_value.

Each original text-based similarity file contained at least the top 15 similarity pairs for each document. However, large similarity files (2M documents with the top15 or higher similarity pairs for each document) are too large for our clustering routines to run efficiently. Thus, we filter the similarities to generate a reduced size similarity file. Contrary to intuition, it has been our experience that filtering of similarity lists acts as noise reduction, and actually increases the accuracy of a clustering solution.

Our solution for filtering the similarities was to generate a top-n similarity file for each case. The basic idea behind this filtering is that papers that are more strongly linked (have higher similarities) should contribute more edges to the solution space. Papers with small similarities should not contribute as many edges because they are not very similar to anything. To give an example, consider paper A whose average similarity value over the top15 similarities is 0.85. Then consider paper B whose average similarity over the top15 similarities is 0.15. Note that we picked paper A because it has the highest average top15 similarity value, and paper B because it has the lowest average top15 similarity value of the entire set of 2.15M papers. These two papers thus define the top15 similarity range; paper A contributes 15 edges to the similarity file, and

paper B contributes only 5 edges to the similarity file. We then scale all other papers between these two based on *log(avg(top15 sim))*. Thus, each paper contributes between 5 and 15 edges to the similarity file. We then de-duplicated all (A:B – B:A) pairs for efficiency, and used these top-n similarity files as input to the clustering steps.

2.a3.1 Co-Word Analysis (Co-Word)
by Russell J. Duhon and Katy Börner

The co-word analysis steps described here were run in turn on both the MeSH-document and word (TA)-document matrices. A minor preprocessing step was taken first, to reduce later computational overhead. All terms and documents were replaced with integers that uniquely identified them. This allows later calculations to only store a single integer per document or term, instead of having to store an entire string of characters.

Before running the similarity calculations, tf-idf (term frequency, inverse document frequency) was performed on the input data. No special work was required to have tf-idf run quickly, though new code was written for this purpose (to run on the data format being used, instead of requiring the data format be transformed to work with existing means for calculating tf-idf). The code operated as follows:

- Load the number of documents each term occurs in, *d*, from a file containing the values. This file was created as part of the original processing.
- For each term *i*, compute the inverse document frequency as $idf_i = log(D/d_i)$, where *D* is the total number of documents in the corpus.
- Read the file with the raw occurrence data, which consists of term *i*, document *j*, raw occurrence ($n_{i,j}$) triples. While reading each triple, keep track of the total occurrences of all terms *k* in eachdocument *j* as ($sum(n_{k,j})$).
- Read the file with raw occurrence data again. This time, as each term, document, raw occurrence triple is read, calculate the term frequency as $tf_{i,j} = n_{i,j} / sum(n_{k,j})$.
- Calculate tf-idf as $tf\text{-}idf_{i,j} = tf_{i,j} * idf_i$, and write out each term, document, tf-idf triple to an output file.

Then, the cosine similarity code mentioned in the previous section was run. That code computes the fifty most similar documents for each document in a range of a few thousand documents. Computations covering different ranges of documents are done in parallel. The code works as follows:

- Load the tf-idf term occurrence data for the range of documents needing similarities computed, and all documents they will be compared to, into memory.
- Specifically, for each document, retain a sorted list of terms that are non-zero, and a mapping from each term to the tf-idf of that term in that document.
- For every document *A* loaded into memory, compute the Euclidean norm $||A_j||$ of the term vector as $||A_j|| = sqrt(sum(tf\text{-}idf_{i,j}^2))$ over all terms *i*.
- For every document in the range needing similarities computed, prepare an empty min-heap to hold the top 50 values.
- For every document *A*, calculate the cosine similarity between it and all documents B with which it is to be compared, as $cosine = A \bullet B / ||A|| \, ||B||$, where the numerator is the dot product of the tf-idf vectors associated with documents *A* and *B*, and the denominator

consists of the Euclidean norms of *A* and *B*. An efficient way to calculate the dot product of the two vectors is to compute the intersection of the sorted term lists for each document, then multiply and sum only the tf-idfs of the intersecting elements.

- While computing the cosine similarities for any document A, the top 50 list can be effectively managed using the min-heap as follows. Load the first 50 similarities calculated into the min-heap sorted by descending cosine value. For each succeeding similarity value, if newly calculated similarity for B is larger than the minimum similarity in the min-heap, add the new similarity into the heap using cosine value ranking, and discard the lowest value. If the newly calculated value for B is less than the minimum value in the heap, the newly calculated value can be discarded.
- At the end, write out all documents and their similarities for the min-heap of each document in the range of documents needing similarities computed (this could alternatively be done immediately after the loop step for that document, before the next document's most similar documents were computed).

When parsing out the cosine calculation to multiple processes and processors, the MeSH similarity code uses all documents B with each document A. However, the TA calculation was much larger (272,926 TA tokens vs. 23,347 MeSH tokens); thus all documents B could not be placed in the same calculation for any given document A. Instead, each run of the code uses a large chunk of the total data as the documents to be compared to. This does not add much overhead compared to the having the entire data set in memory. Due to the specific constraints of the computers used to run the calculation, seventeen chunks were used. For the MeSH similarities, the range of documents needing similarities computed was 20,000 in size, and for the Title/Abstract similarities, the range was 5,000 in size. Note that these numbers are specific to this corpus and IU hardware configuration and would need to be adjusted for other data sets or hardware. The target should be, absent external constraints, to allow as many processes to fit in memory simultaneously as there are processing cores on the computers being run on. No matter how fast the individual processors, a substantial number will be required to complete the work in a reasonable length of time, as described in 2.a2. Approximately sixty-fold parallelism was used in this case.

2.a3.2 Latent Semantic Analysis (LSA)
by Russell J. Duhon and Katy Börner

Latent Semantic Analysis (LSA) is a technique for reducing the dimensionality of a document-term matrix by finding lower rank matrices that together approximate the information in the full matrix. It works by minimizing the sum of the squares of differences between the entries in the original matrix and the approximations of those values. It is typically computed using Singular Value Decomposition (SVD), though it can also be computed using techniques such as Principal Components Analysis.

LSA typically uses the following process:

- The values in the term-document matrix are converted to weights, in this case to tf-idf weights, using the process mentioned in section 2.a3.1. Conversion to tf-idf weights are a current best practice, although log-entropy weighting is also commonly used.

- An SVD calculation is performed on the entire matrix to compute the singular value matrix **S** using

  $$\mathbf{X} = \mathbf{T}\,\mathbf{S}\,\mathbf{D}^{\mathbf{T}}$$

  where **X** is the original term-document matrix with *D* documents and *N* terms, **T** is the 'term' matrix composed of *N* terms and *k* singular vectors (or concepts onto which the documents load to varying degrees), **S** is the singular value matrix with *k* singular values along its diagonal, and **D** is the 'document' matrix composed of *D* documents and *k* singular vectors. *k* is typically on the order of 300-500, and is thus much smaller than *N*. The representation of the matrix is thus much smaller than the original matrix, and much easier to manipulate further. In addition, LSA accounts for synonymy to some degree in that similar terms will end up loading highly on the same factor.
- Similarities between document pairs are then typically computed as cosines between rows of the reduced matrix **D** using the dot product between rows.

The use of LSA, besides greatly reducing the size of the data, has been shown to lead to similarities between documents that are better at satisfying human expectations than do typical co-word approaches (see, for instance, Gorrell 2005, Generalized Hebbian Algorithm for Latent Semantic Analysis)[1]. Within an LSA calculation, there are many different methods used to either directly calculate, or to approximate, the matrix **S**. The most standard method is to use the SVD calculation mentioned above; however, SVD is not always practical at the scale of 2 million documents.

The particular method used to calculate **S** in this project is based on the Generalized Hebbian Algorithm (GHA). GHA LSA approximates the top concepts of the LSA (the singular values of the **S** matrix) one at a time, with tunable parameters governing desired convergence. Since only the top concepts are desired, this is much faster than approaches that calculate all of the LSA concepts simultaneously. While it does not fully minimize the error described above, the eventual output is only in terms of similarities, and only the highest similarities. Thus, provided the documents have similar relative loadings on the GHA LSA vectors towards the extremes, failing to completely minimize the error described above will have little or no effect on the results. As such, parameters were chosen that provided acceptable runtimes, and the first few vectors were compared against more strict approximations to verify that this assumption was reasonable. Specifically, GHA LSA was run with a convergence length (the parameter that governs how exactly vectors are approximated) of 1,000,000 for the MeSH data, retaining the first 200 concepts, and a convergence length of 10,000 for the title/abstract data, retaining the first 100 concepts. The code was modified (as recommended in a comment by the author, by altering a constant) to allow all data to be loaded into memory. Flush statements were also added after each time the code writes to a file or standard out, because data was being buffered for too long before being written.

The output of GHA LSA is the diagonal matrix of singular values **S** and the *N* by *k* term matrix **T**. From this, it is possible to quickly compute the document matrix **D** by multiplying the inverse of the singular value matrix by the transpose of the term-concept matrix by the transpose of the document-term matrix, and taking the transpose of the result. By writing the result out in the

[1] http://www.dcs.shef.ac.uk/~genevieve/gorrell_webb.pdf

same format as the data used for the title/abstract cosine similarity calculations, the same cosine code from section 2.a3.1 can be used to compute cosine similarities of the concept vectors for each pair of documents, with some small modifications. These are:

- Since the LSA output is much denser than the previous input data, but with many fewer dimensions, all entries (zero and non-zero) for each document are stored in fixed-width arrays, and the dot products of the arrays are calculated using BLAS (Basic Linear Algebra Subprograms).
- Also, since the LSA output for MeSH and for title/abstract data are similar in density and dimension, the same similarity calculation code can be used in each case.

In addition, when the top-n sim file for the LSA sets were calculated, an edge range of 5 – 13 was used rather than the typical 5 – 15. The 5 – 15 range was calculated first, but gave a very large number of edges. Thus, the range was reduced to 5 – 13 to reduce the total number of edges to something closer to what was ultimately used for most of the other similarity files.

2.a3.3 Topic modeling algorithms based on Latent Dirichlet Allocation (Topics) – UCI by Dave Newman

The topic modeling algorithms here were only run on the TA data, and were not run on the MeSH data.

Preprocessing: We obtained the word-document data as is from the STS website at IU. We conducted a small amount of additional preprocessing to the word-document data set. First, 131 topically-uninteresting, but frequently occurring words were removed from the data (e.g. words such as 'study', 'studies', 'result', 'results', etc.) Next, all terms that occurred fewer than 50 times across the entire corpus were removed. The resulting word-document data retained all 2,153,769 documents, but reduced the number of unique tokens to 65,776. The total number of word-document triples was 243,724,698, thus giving an average length of 113 words per document.

Modeling: We ran the standard Gibbs-sampled topic model on the collection. Three separate topics models were learned at the following topic resolutions: T=500, T=1000 and T=2000 topics. These topics models were run for: 1600, 1500 and 1200 iterations (i.e. entire sweeps through the corpus) respectively. We used the following Dirichlet prior hyperparameter settings: beta=0.01 and alpha=0.05*N/(D*T). Examples of topics are in http://www.ics.uci.edu/~newman/katy/. See:
http://www.ics.uci.edu/~newman/katy/topics.T500.iter1600.txt
http://www.ics.uci.edu/~newman/katy/topics.T1000.iter1500.txt
http://www.ics.uci.edu/~newman/katy/topics.T2000.iter1200.txt
Reports comparing similar topics are:
http://www.ics.uci.edu/~newman/katy/closetopics.T500.txt
http://www.ics.uci.edu/~newman/katy/closetopics.T1000.txt
http://www.ics.uci.edu/~newman/katy/closetopics.T2000.txt

Similarity Computation: We computed the top 20 most similar documents for each of the 2,153,769 documents in the corpus. A topic-based similarity metric was used, using an equal

weighting of the T=500, T=1000 and T=2000 topic models. Specifically, we computed the similarity between document *a* and document *b* using the following formula:

$$sim(a,b) = 1 - (L1(a500-b500) + L1(a1000-b1000) + L1(a2000-b2000))/6$$

where *L1* is the L-1 norm, and *a500*, etc. is the distribution over T=500 topics of document *a*. We spot-checked a small number of similarities using PubMed, and this spot-checking indicated good agreement and consistency with PubMed. Similarity results are in: http://www.ics.uci.edu/~newman/katy/newman_topic_model_sim.txt.gz

2.a3.4 Self-Organizing Maps (SOM)
by André *Skupin and Joseph Biberstine*

The self-organizing map (SOM) method is a form of artificial neural network that generates a low-dimensional geometric model from high-dimensional data. The software used in this study is one of the most widely known SOM implementations, SOM_PAK, which was created during the mid-1990s by the Neural Networks Research Centre at Helsinki University of Technology, under the direction of the inventor of the SOM method, Teuvo Kohonen.

In order to allow processing of the Medline-based data, a number of modifications had to be made to the SOM_PAK program and the whole package recompiled for the particular hardware being used. Notable changes include a greatly increased maximum line size for input files. That is due to the fact that dimensions numbered in the thousands were likely not anticipated by the original programmers, yet, with terms used in the documents treated as dimensions, and each dimension ultimately represented with floating-point weights (one value for each neuron and dimension), lines can be several hundred thousand characters long.

Another modification considered the distance metric used. SOM_PAK uses the Euclidean metric throughout, which is not appropriate for the extremely sparse vectors encountered with these kinds of text documents. That's an adaptation of SOM_PAK using the cosine measure was ported over to this version of SOM_PAK (since the current version of SOM_PAK used is different from the one for which the cosine measure was originally created).

Early during preprocessing it became clear that the dimensionality of the word-document matrix (d=272,926) would far exceed what could be done when raw term counts where used. Thus, further processing was focused on the MeSH-document matrix, which has a 90% smaller set of raw dimensions (d= 23,347).

Even using the MeSH-document data, the SOM training with SOM_PAK and raw data failed when presented with anywhere near the number of original dimensions (20,000+) or a large number of neurons (2 million plus). After numerous experiments, it was decided to keep MeSH terms occurring in a minimum of 2,000 documents (i.e., 2,236 terms). Further experimentation showed that SOM_PAK failed when the number of these 2,236-dimensional SOM neurons approached 80,000. A workable solution was finally found with a SOM consisting of 275x275 neurons, which would allow making distinctions among 75,625 locales in the high-dimensional space.

Even with reduced dimensionality, the complete SOM input file of all 2.1 million documents still amounts to over 9GB, due to the sparseness of the resulting binary vectors. That is far too large to deal with directly. Therefore, during neural network initialization and training, several independently drawn random samples of the data set were used. Yet run time were on the order of days. We have not yet been able to evaluate the success of this training, due to numerous technical issues – all related to the sheer size of the input data – and mapping of the full data set onto the trained SOM is estimated to take several weeks of raw computing time. That does not include various necessary post-processing steps. The peculiar rise in the quantization error during training is also still under investigation.

Thus, it has thus become clear in the course of this study that a brute-force use of SOM for representing text content may be suitable for corpi containing several thousand to several tens of thousands documents, but that a large corpus requires a more refined approach.

2.a3.5 Collexis Engine
by Bob Schijvenaars and Aaron Sorensen, Collexis

Collexis processed the data using the typical vector space model as well. On both datasets (the MeSH-document and word-document) the *bm25* algorithm was applied (Stephen Robertson, Karen Spärck Jones) as described below.

The specific formula used to compute the similarity between a document $q = (w_1, \ldots, w_n)$ and another document d was:

$$s(q,d) = \sum_{i=1}^{n} IDF(w_i) \cdot \frac{f(w_i)(k_1+1)}{f(q_i) + k_1\left(1 - b + b\,|D| / \overline{|D|}\right)},$$

Where $f(w_i)$ is the frequency of word w_i in document d. Note that $f(w_i)$=0 for words that are in document q but not in d. For the constants k_1 and b typical values were chosen (2 and 0.75 respectively). Document length $|D|$ was estimated by adding the word frequencies w_i per article. The average document length $\overline{|D|}$ was computed over the entire document set. The IDF value for a particular word w_i was computed using:

$$IDF(w_i) = \log \frac{N - n(w_i) + 0.5}{n(w_i) + 0.5}$$

where N is the total number of documents in the dataset and $n(w_i)$ is the number of documents containing word w_i. Note that each individual term in the summation in the first formula is independent of document q. Hence these were computed first and to keep computing time within acceptable limits, all scores below 2.0 were discarded. (Note, this threshold of IDF $\geq$ 2.0 acts to limit the word set to words where $n(w_i) < 21{,}324$, or words that occur in less than 0.99% of the documents).

For the MeSH terms a slightly different filter was applied to keep computation time acceptable since term frequencies are not available for MeSH terms. Therefore, scores for individual terms follow a different distribution. Scores below 1.5 were discarded, with the exception of the terms of two documents that only contained terms with a score < 1.5.

2.a3.6 PubMed Related Articles (pmra)
by Kevin W. Boyack

In addition to the text-based similarity types detailed above, all of which were mentioned in the original proposal, we added one text-based similarity type to the study – the PubMed Related Articles (*pmra*) similarity. When one does a search on PubMed/MEDLINE and displays the results in Summary mode, most records show a *Related Articles* link.

> "The *Related Articles* link is as straightforward as it sounds. PubMed uses a powerful word-weighted algorithm to compare words from the Title and Abstract of each citation, as well as the MeSH headings assigned. The best matches for each citation are pre-calculated and stored as a set." [2]

The *pmra* algorithm used to pre-calculate these *Related Articles* (Lin & Wilbur) has been through sufficient testing and review to have been accepted for use in PubMed, and is thus a *de facto* standard. It seemed wise to add a set of calculations based on this method to the project as an additional point of comparison. More detail on the *pmra* method is given in an Appendix.

Dave Newman (UCI) wrote and ran a script that queried PubMed for these pre-calculated matches for each document in our corpus. This script returned a rank-ordered list of the *Related Articles*, which was then post-processed to limit the lists to only documents that were in the corpus. After post-processing, these lists contained from 2 to 20 *Related Articles* for each article in the corpus, listed in rank order, but without similarity values.

In order to compare a map of science using these *Related Articles* to the other maps, it was necessary to convert the rank-ordered lists of relationships into similarity values. We estimated the *Related Articles* similarity values as

$$RA_{i,j} = 1.00 - 0.02*(51 - rank_{i,j})$$

for all articles *j* related by $rank_{i,j}$ to article *i*. Thus, for any article *i*, the first ranked *Related Article* was assigned a similarity value of 1.00, the second a similarity value of 0.98, etc. We cannot note strongly enough that these are not the original similarity values calculated using the *pmra* method, but are rather estimated values from rank orders. This similarity method is referred to hereafter as *pmra-est*.

Given that the original distribution of similarities ranged from 2-20, rather than from the desired 5-15 (the definition of our top-n similarity files), we needed to generate a top-n similarity file that would have a similar edge distribution to the other text-based similarity files. To do this, we used the following transformation:

[2] http://www.nlm.nih.gov/bsd/disted/pubmedtutorial/020_190.html

Original number of similarities	Top-n similarities used
2-6	No change
7-8	7
9-10	8
11-12	9
13-14	10
15-16	11
17-18	12
19-20	13

The final *pmra-est* top-n similarity file contained 18,511,515 document pairs.

2.b1: Citation Pre-Processing
by Kevin W. Boyack

Several different citation-based maps have been generated from the corpus: one based on co-citation analysis, one based on bibliographic coupling, and two based on direct citation. Each of these methods requires different processing from a single starting point. This starting point is the full list of citing_document, reference_document (citing:cited) pairs from the corpus. This initial list of citing:cited pairs consists of 80,754,581 pairs that cite 15,503,380 unique references; the distribution for this unfiltered set of references is shown in Figure 3, top.

2.b1.1: Co-citation method

For purposes of generating a co-citation similarity, references were filtered using the following formula which is based on the standard co-citation practice at STS:

- The articles from the corpus were grouped by publication year, as shown in the "Final" column of Table 2, and separate citing:cited pair files were generated for each of the five publication years.
- For each yearly set, the citations to each unique reference document were counted.
- For each yearly set, reference papes that did not meet the following criteria were filtered out of the set:
  (age = 0 and ncites $\geq$ 3) OR (age $\leq$ 3 and ncited $\geq$ (age+1)) OR ncited $\geq$ 5
  where age = citing publication year – cited publication year.
- The remaining references from the five yearly sets were combined to form the full set of references, resulting in a set of 2,473,611 unique references. The original citing:cited pair list was then filtered to contain only those pairs where the cited document was in the final reference list. This resulted in a pairs file containing 50,221,140 citing:cited pairs. The distribution corresponding to this set of references is shown in Figure 3, bottom; all references were cited at least 4 times, and the most highly cited reference was cited by 1.1876% of the citing documents.

The final set of citing:cited pairs was then used to generate a similarity measure using the following process:

- Co-citation frequencies between pairs of reference documents were calculated. A co-citation occurs when two documents are cited by the same citing paper. All such

instances were counted, resulting in a matrix with the numbers of co-citation counts between cited document pairs. The total number of cited document pairs with a non-zero co-citation count was xxx,yyy,zzz.

- Each co-citation value was modified using *1/log(p(C+1))* where *p(C) = C(C-1)/2* and *C* is co-citation counts.
- K50 values were generated from each modified frequency value as:

$$K50_{i,j} = K50_{j,i} = \max\left[\frac{\left(F_{i,j} - E_{i,j}\right)}{\sqrt{S_i S_j}}, \frac{\left(F_{j,i} - E_{j,i}\right)}{\sqrt{S_i S_j}}\right],$$

where $F_{i,j}$ is the modified frequency for the pair of references *i* and *j*, and

$$\overline{F_i} = \frac{1}{n}\sum_{k=1}^{n} F_{i,k}, \quad k \neq i \quad , \qquad S_i = \sum_{j=1}^{n} F_{i,j}, \quad j \neq i \quad ,$$

$$SS = \sum_{i=1}^{n} S_i \quad , \qquad E_{i,j} = \frac{S_i S_j}{\left(SS - S_i\right)} \quad .$$

For K50, note that $F_{i,j} = F_{j,i}$, but that $E_{i,j} \neq E_{j,i}$. *E* is an expected value of *F*, and varies with S_j . K50 differs from most other measures in that it is a relative measure that subtracts out the expected value. Thus K50 will only be positive for those reference paper interactions that are larger than expected given the matrix row and column sums.

- The full K50 matrix is too large for our clustering routines, thus we filter the similarities to generate a reduced size similarity file. Contrary to intuition, it has been our experience that filtering of similarity lists acts as noise reduction, and actually increases the accuracy of a clustering solution. Filtering was done by 1) removing all pairs with negative K50 values, 2) sorting the remaining list by reference and descending K50 value, and removing all references which had more than 15 K50 values tied in the first position, and 3) using the total degree (or number of pairs) distribution for each reference, and scaling the log(degree) values to a 5-15 scale. The degree for each reference thus determines how many pairs that reference brings into the final similarity file, varying between 5 and 15. We call this a top-*n* similarity file. Although the clustering routines to be used ignore (A:B – B:A) duplicates, it is useful to de-duplicate the similarity file for efficiency. After de-duplication, the total number of cited document pairs in the cited document similarity file was 15,537,317.

2.b1.2: Bibliographic coupling method

While co-citation generates a similarity between pairs of cited documents, bibliographic coupling generates a similarity between pairs of citing documents using the original (unfiltered) citing:cited pair list mentioned above. No filtering of citing documents was needed because the full set (2.15M documents) is well within the range of what our clustering routines can manage. The full citing:cited pair list was used to generate a similarity using the following process:

- Bibliographic coupling frequencies between pairs of citing documents were calculated. Bibliographic coupling occurs when two documents reference the same reference paper. All such instances were counted, resulting in a matrix with the numbers of bibliographic coupling counts between citing document pairs. The total number of citing document pairs (full matrix) with a non-zero coupling count was 170,835,050.

- The next three steps were the same as the final three steps listed for the co-citation method above: calculating a modified frequency using *1/log(p(C+1))*, calculating K50 values, and filtering the full list of similarities to a top-n similarity file. After de-duplication, the total number of citing document pairs in the bibliographic coupling similarity file was 14,159,303.

2.b1.3: Direct citation method

Direct citation is where one document within the set cites another document within the set. Thus, although pre-filtering is not necessary, documents that either do not cite or are not cited by any other document within the set will not be a part of the calculation. The citing:cited pairs (or direct citations) for this calculation were determined by finding all pairs where both the citing and cited document were within the set. This reduced the citing:cited pair list to 23,218,091 pairs. This list was used to generate a similarity using the following process:

- Each citing:cited pair was assigned a weight *wt = 1/n* where n is the total number of papers cited by the citing paper in the pair.
- Since this citing:cited:wt list is directional, and thus comprises only the upper half of a full citation matrix, this list was flipped (cited:citing:wt) and concatenated to the upper half, thus forming a full matrix.
- The next two steps were the same as the final two steps listed for the co-citation method above: calculating K50 values, and filtering the full list of similarities to a top-n similarity file. After de-duplication, the total number of citing document pairs in the direct citation similarity file was 7,581,738. A total of 1,996,050 of the possible 2,153,769 citing documents are included in the calculation using this direct citation method.

Add top-n similarity distributions (5-15 range) for each sim?

Task 3: Clustering
by Kevin W. Boyack

After all of the similarities files were generated, the next step in the process was to generate a detailed clustering of documents from each similarity method. The same clustering method was used for all similarity measures; thus the clustering method should not contribute to any variability in the final results. The basic clustering process was comprised of 3 main steps.

1) The DrL graph layout routine was run 10 separate times using a similarity file as input, starting with 10 different random seeds, and using a cutting parameter of 0.975. DrL uses a random walk routine and prunes edges based on degree and edge distance. A typical DrL run using an input file of 2M nodes and 15M edges will cut approximately 60% of the input edges, where an edge represents a single document-document similarity pair from the original input file. At the end of the layout calculation, roughly 40% of the original edges will remain. Different starting seeds (essentially different starting points for the random walk) will give rise to different graph layouts and different (but typically highly overlapping) sets of remaining edges. We use these differences to our advantage in this clustering process.

2) Those edges that appear in 6 or more out of the 10 DrL solutions are considered to be the robust edges, and are listed in a separate file. These are used as the input edges to a single-linkage clustering routine (see Appendix …), which essentially finds and outputs all distinct graph components. Each separate component is a cluster, and these clusters are referred to as level 0 clusters.
3) Logic dictates that a cluster should have a minimum size; otherwise there is not enough content to differentiate it from other clusters. In our experience, a cluster should contain a minimum of approximately five papers per year (or 25 papers over the five year length of the corpus) to be considered topical. Thus we take all clusters with fewer than 25 papers, and aggregate them. This is done by calculating a K50 similarity between the small clusters and all other clusters, and aggregating the small cluster with the cluster to which it has the largest K50 similarity. K50 values are calculated from aggregated modified frequency values (the *1/log(p(C+1))* values) where available, and from the aggregated top-n similarity values in all other cases.

Previous experience has shown that a cluster solution based on the combination of 10 DrL runs is much more robust than that from a single DrL run. For example, using a co-citation model of roughly 2.1M documents and 16M edges, the adjusted Rand index (a measure of correspondence) between pairs of single DrL solutions was 0.32, while the adjusted Rand index between pairs of 10xDrL solutions was over 0.80. Requiring that an edge persist in 6 out of 10 separate DrL solutions thus limits the final solution to only those edges (and thus the resulting clusters) that are relatively robust. However, this robustness can also have a deleterious effect on the final solution – any nodes that do not have any edges persisting in 6/10 DrL solutions will drop out of the solution. To some degree, this is in itself a measure of the robustness (or conversely, ambiguity) in the similarity measure. Solutions in which many nodes are dropped do so because of ambiguity in the overall similarity space of the document set. Thus, similarity measures that generate solutions dropping many nodes can be thought of as more ambiguous than similarity measures that drop few nodes.

3.a: Co-citation Clustering

Nearly all of the similarity measures employed in this study are between the actual MEDLINE documents that are being clustered. Thus, the process detailed above works directly for these measures. However, there were two calculations – the co-citation calculation and one of the direct citation calculations – where the similarity and resulting clustering were based on reference papers rather than the MEDLINE documents. For these two cases the output of step (2) is a list of the reference papers assigned to each level 0 cluster.

In co-citation analysis, clusters of reference papers are generated, and then the citing papers are assigned to those cluster based on the reference patterns. One of the benefits of co-citation analysis is that citing articles can be assigned to multiple clusters if they cite literature from multiple clusters. We use the following detailed methods (between steps (2) and (3) above) to assign citing papers to the level 0 reference paper clusters.

- For each citing paper (from the original MEDLINE corpus), the number of references to papers in each of the level 0 clusters was calculated from the reference lists.

- For each citing paper, the maximum number of references to level 0 clusters was calculated, and the paper was labeled as *unambiguous* if that maximum number was greater than 1. Citing papers with a maximum number of 1 were labeled as *ambiguous*.
- Calculate a t-value for each citing paper, level 0 cluster pair where t is the number of references to papers to the level 0 cluster divided by the square root of the number of papers in the level 0 cluster. Thus, the t-value is related to the fraction of the cluster that is cited by the citing paper. For unambiguous papers, t-values are only calculated for paper/cluster papers where the number of references to the cluster is greater than 1. This avoids long tails for unambiguous papers.
- For each citing paper, normalize the t-values so that they add up to 1. These normalized t-values are used as the current paper to level 0 cluster weights.
- For each citing paper, if it is also a reference paper in a level 0 cluster, the fractions are adjusted in the following way. The fractions for these papers are set to one half their previous values, thus giving each a summed fractional value of 0.5. The remaining 0.5 fraction is assigned to the level 0 cluster to which the paper belongs as a reference paper. The citing paper, level 0 fractions are then re-summed to give the final fractional assignments.

Once the citing paper assignment have been made, then clustering step (3) above is applied to merge clusters that contain fewer than 25 papers into larger clusters.

3.b: Tuning of the Clustering Process for Text-based Methods

It is important that any classification or map of science have a very high coverage of the document space. Early results from the citation-based approaches using the clustering approach detailed above (requiring an edge to appear in 6/10 DrL solutions) showed that they each retained between 90-98% of the articles in the cluster solution. However, the first two text-based approaches (pmra and Collexis TA) that were run using this same clustering criterion retained only 84.0% and 83.0% of the articles in the corpus, respectively.

This coverage of less than 85% is not sufficient. We thus decided to attempt tune the clustering criteria in a way that would increase coverage while minimizing changes to the cluster size distribution, or to the accuracy of the cluster solution. We found that by reducing the threshold to 4/10 edges, coverage was increased by 10.2% and 10.9% (to around 94%), respectively, for our two cases. However, along with the increase in coverage, there was a large increase in the total number of edges retained in the solution, and a corresponding increase in overall graph connectedness, resulting in a giant component containing over 90% of the documents. This particular clustering result is not useful. To reduce the size of the giant component, and to obtain a cluster distribution that was similar to that obtained from the 6/10 edge solutions, we implemented an additional criterion into the single-link algorithm that would only merge clusters if the edge on which the merge would be based had an edge weight of 7/10 or higher. The addition of this criterion gave us an appropriate cluster solution; Table 3 shows that this new 4/10 solution gives a slightly larger number of clusters and a slightly larger largest cluster than the original 6/10 solution, but that the two solutions are similar overall. The textual coherence (to be explained in Section 4) of the solutions using the new 4/10 edge criterion were also very similar to those obtained using the 6/10 edge criterion, as shown in Figure 4. We thus decided to

use the 4/10 edge criterion with the 7/10 cluster merge criterion for all of the text-based clustering solutions.

Table 3. Comparison of clustering characteristics for the 6/10 and 4/10 edge criterion solutions.

Method	#A 10xDrL	%A retain	#Lev0 clust	#Lev1 clust	Max Lev1	Coh (text)
Collexis TA 6/10	1,788,084	83.02%	181,806	27,608	536	0.09903
Collexis TA 4/10	2,022,694	93.91%	163,877	28,858	764	0.09798
pmra-est 6/10	1,809,526	84.02%	138,330	27,481	632	0.10567
pmra-est 4/10	2,029,564	94.23%	129,505	28,963	921	0.10364

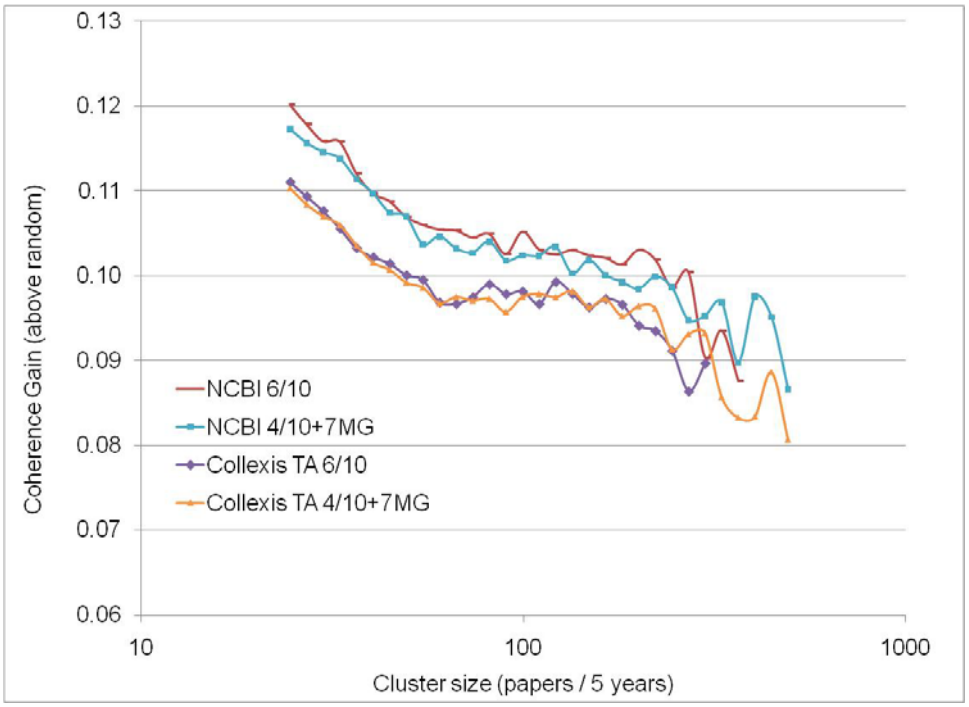

Figure 4. Comparison of the textual coherence of different clustering criteria for two text-based approaches.

3.c: Clustering Results

Metrics from the 10xDrL cluster solutions from each of the various similarity metrics are given in Table 4. Cluster size distributions are shown in Figure 5.

Table 4. Clustering characteristics of the various similarity files.

Method	Top-n sims	Compute	#A per DrL	#A 10xDrL	%A retain
Bib coupling	14,159,303	Low	2,146,549	2,081,022	96.62%
Co-citation	15,537,317	Low		2,118,644	98.37%
Direct citation whl	7,581,738	Low	1,996,050	1,940,535	90.10%
Direct citation frac	7,581,738	Low	1,996,050	1,996,050	92.68%

Co-word MeSH	17,575,220	Medium	2,153,769	2,062,642	95.77%
LSA MeSH		Very high			
SOM MeSH					
Collexis MeSH	18,414,440	Medium	2,153,769	2,011,339	93.39%
Co-word TA	24,276,859	High	2,153,769	1,796,349	83.41%
LSA TA	22,804,907	Very high	2,153,769	1,259,740	58.49%
Topics TA	18,752,066	High	2,153,769	2,033,221	94.40%
Collexis TA	16,604,589	High	2,153,769	2,022,694	93.91%
pmra-est	18,511,515	Very Low	2,153,769	2,029,564	94.23%
Collexis Full					

Method	**#Lev0 clust**	**#Lev1 clust**	**MaxSz Lev1**		
Bib coupling	207,764	32,782	778		
Co-citation	188,561	32,184	3245		
Direct citation whl	456,112	50,505	221		
Direct citation frac	456,112	50,719	376		
Co-word MeSH	95,334	24,708	1517		
LSA MeSH					
SOM MeSH					
Collexis MeSH	146,040	26,864	1015		
Co-word TA	200,233	21,388	657		
LSA TA	335,012	22,757	369		
Topics TA	108,808	24,163	1422		
Collexis TA	163,877	28,858	764		
pmra-est	129,505	28,963	921		
Collexis Full					

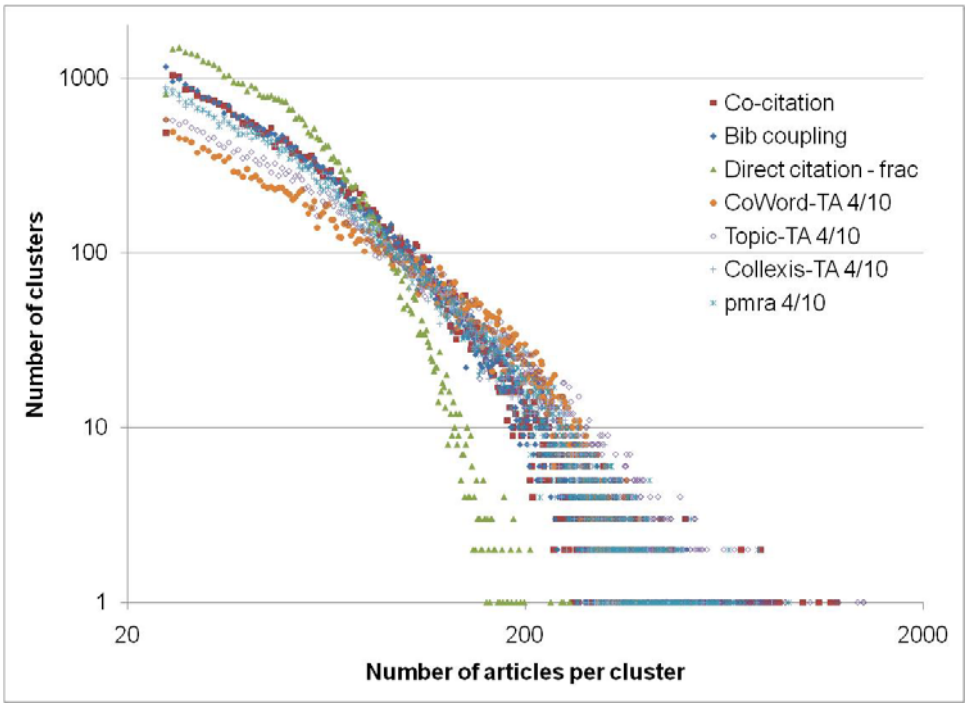

Figure 5. Cluster size distributions for many of the cluster solutions.

The clustering results lead to several observations. First, the direct citation method gives the largest number of level1 clusters, and has the smallest cluster sizes. In fact, there were nearly 10,000 level 0 clusters with fewer than 25 members (including 7,800 with only two members) that could not be aggregated into level1 clusters because there was simply no direct citation relationship with any member of any other clusters. Thus, these level 0 clusters were carried forward to be level 1 clusters. These are not shown in Figure 5. In addition, only 90% of the corpus was placed into clusters by the direct citation method; of the 10% that were not placed in clusters, 7.3% were not even in the original similarity file due to a lack of connectedness with any other article within the corpus, while the other 2.6% were dropped due to the 6/10 edge criteria used in clustering. These results related to direct citation are not surprising – by its very nature direct citation within a five-year window will have far less connection within the set than will the other citation or text-based methods.

Second, co-citation gives by far the largest cluster sizes for its largest clusters. However, once those clusters with size > 400 articles are accounted for, the co-citation and bibliographic coupling distributions are very similar. Co-citation and bibliographic coupling also have the highest article retention rates using the 6/10 edge criterion at 98.4% and 96.6%, respectively. These two methods provide the most comprehensive coverage of the corpus using the clustering parameters selected.

Third, the co-word TA method has by far the lowest coverage of any of the methods at 83.4%. It also had by far the largest number of similarities in its input file (24.28 million) of any measure tested. These two facts are likely related. Although the generation of the co-word TA top-n similarity file was done using the exact same method used for other text-based similarities, the distribution of similarities (leading to the top-n assignment) was quite different, and gave rise to

a larger similarity file. Although the reason for this is unknown, we speculate that it is due to very slight variations in similarity between document sets arising primarily from the high end of the word-document distribution (those words that occur in a very large fraction of documents). The other TA methods (Topic and Collexis) both applied additional processing to the matrix that would have mitigated against such behavior.

Fourth, the numbers of clusters from the bib coupling, co-citation, and many of the text-based solutions are in a similar range (26,000 – 33,000 clusters), and thus are roughly comparable for the comparisons that will be reported in a subsequent section. A few of the methods have fewer clusters (e.g., 21,400), but even those are within an acceptable range for evaluation.

Task 4: Evaluation of Cluster Solutions
by Kevin W. Boyack

Given that the main purpose of this project is to determine which method of generating a map of science (or a fine-grained categorization of science), we propose to use three different types of measurements to compare the accuracy of the cluster solutions. These include measuring coherence of clusters based on 1) textual information, and 2) reference information, and also measuring the dispersion of grant-article linkages from individual grants into the cluster solutions.

4.a: Coherence Using a Text Basis

The first evaluative measure we use in this study compares the topical coherence of the different cluster solutions. We calculate the topical coherence of each cluster in each solution using the information radius (IRad), or Jensen-Shannon divergence (Fuglede & Topsoe, 2004). IRad is calculated for each document from the word probability vector for that document, and from the word probability vector for the cluster in which the document resides as:

$$IRad = D(p\|\frac{p+q}{2}) + D(q\|\frac{p+q}{2})$$

where *p* is the frequency of a word in a document, *q* is the frequency of the same word in the cluster of documents, and *D* is the distribution over all words and documents in the cluster. The textual sample for each document used in this calculation was a concatenation of the document title and abstract. IRad is reported for each cluster, and is the average IRad value over all over documents in the cluster.

IRad is a divergence measure, meaning that if the titles and abstracts of the documents in a cluster are very different from each other, using different sets of words, the IRad value will be very high, or close to 1.0. Clusters of documents where the titles and abstracts use similar sets of words, a less diverse set of words, will have a lower divergence. In essence, IRad measures how closely the titles and abstracts in a cluster share the same vocabulary.

IRad varies with cluster size. For example, a cluster with 10 very different documents will have a larger set of unique words, and thus a higher divergence value than a cluster with only 3 very different documents. The maximum possible IRad values for various cluster sizes will occur when the documents in the cluster have completely different sets of words. These clusters can be

approximated, for a particular corpus, by forming random clusters of documents from that corpus. We have calculated IRad values for various cluster sizes from this SBIR corpus, as shown in Figure 6. Each measured divergence value in Figure 6 is an average of the divergence values from a very large number of random clusters (e.g. 5000 random clusters of size 20, 5000 random clusters of size 100, 1000 random clusters of size 500). A curve fit of the measured values was used to estimate the IRad values for every cluster size from 2 to 1000.

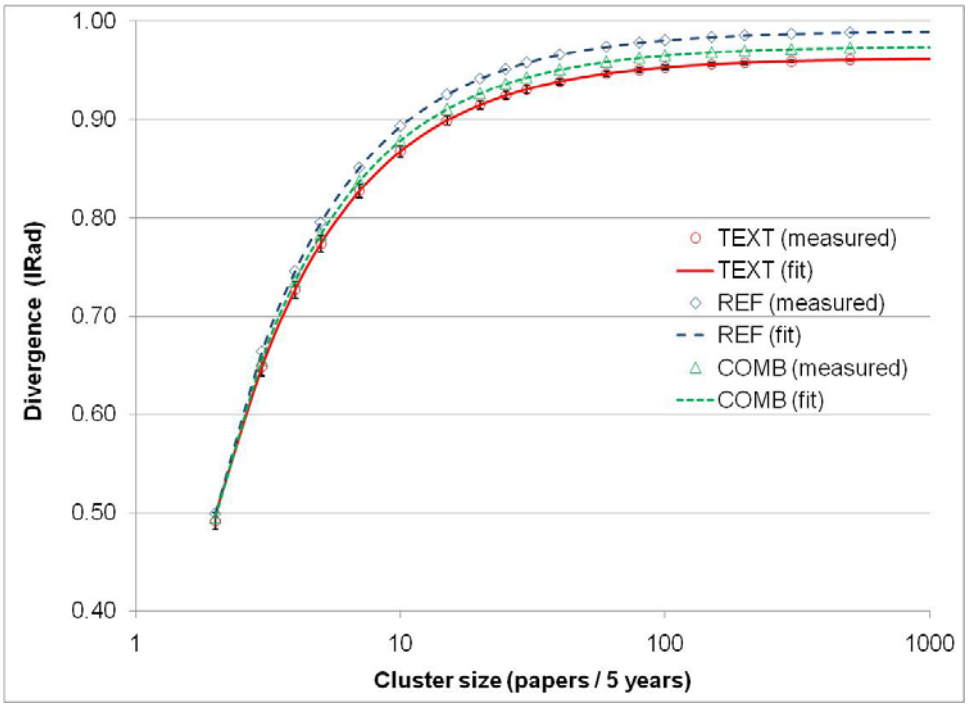

Figure 6. Divergence of random sets of documents by cluster size.

Coherence was calculated from divergence values for each cluster as:

$$Coh_i = Div(rand)_i - Div(actual)_i$$

where Div(rand) is the random divergence for the particular cluster size. The average coherence value for an entire cluster solution is then calculated as:

$$Coh = sum(n_i * Coh_i) / sum(n_i)$$

and is shown in Table 5.

Table 5. Coherence results from the various similarity files.

Method	Coverage	Coh (text)	Coh (ref)	Coh (joint)	
Bib coupling	96.62%	0.08599	0.11684	0.10007	
Co-citation	98.37%	0.08167	0.11071	0.09472	
Direct citation whl	90.10%	0.06072			
Direct citation frac	92.68%	0.06138	0.08138	0.07020	
Co-word MeSH	95.77%	0.07643	0.07360	0.07584	
LSA MeSH					
SOM MeSH					
Collexis MeSH	93.39%	0.07654	0.07360	0.07590	

Co-word TA	83.41%	0.07580	0.05761	0.06846	
LSA TA					
Topics TA	94.40%	0.09374	0.07699	0.08752	
Collexis TA	93.91%	0.09798	0.09560	0.09787	
pmra-est	94.23%	0.10364	0.10597	0.10554	
Collexis Full					

Given that this coherence value is based on textual components (words from titles and abstract), we fully expected the text-based solutions to give higher coherence values than citation-based solutions. In addition, among text measures we expected the Topic model measure to do particularly well given that a divergence measure (Kullback-Leibler divergence) that is related to IRad is used in its calculation.

4.b: Coherence Using a Reference Basis

4.c: Coherence Using a Joint Text-Reference Basis

Computation of Related Articles
(http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=helppubmed&part=pubmedhelp#pubmedhelp.Computation_of_Relat)

The neighbors of a document are those documents in the database that are the most similar to it. The similarity between documents is measured by the words they have in common, with some adjustment for document lengths. To carry out such a program, one must first define what a word is. For us, a word is basically an unbroken string of letters and numerals with at least one letter of the alphabet in it. Words end at hyphens, spaces, new lines, and punctuation. A list of 132 common, but uninformative, words (also known as stopwords) are eliminated from processing at this stage. Next, a limited amount of stemming of words is done, but no thesaurus is used in processing. Words from the abstract of a document are classified as text words. Words from titles are also classified as text words, but words from titles are added in a second time to give them a small advantage in the local weighting scheme. MeSH terms are placed in a third category, and a MeSH term with a subheading qualifier is entered twice, once without the qualifier and once with it. If a MeSH term is starred (indicating a major concept in a document), the star is ignored. These three categories of words (or phrases in the case of MeSH) comprise the representation of a document. No other fields, such as Author or Journal, enter into the calculations.

Having obtained the set of terms that represent each document, the next step is to recognize that not all words are of equal value. Each time a word is used, it is assigned a numerical weight. This numerical weight is based on information that the computer can obtain by automatic processing. Automatic processing is important because the number of different terms that have to be assigned weights is close to two million for this system. The weight or value of a term is dependent on three types of information: 1) the number of different documents in the database that contain the term; 2) the number of times the term occurs in a particular document; and 3) the number of term occurrences in the document. The first of these pieces of information is used to produce a number called the global weight of the term. The global weight is used in weighting the term throughout the database. The second and third pieces of information pertain only to a particular document and are used to produce a number called the local weight of the term in that specific document. When a word occurs in two documents, its weight is computed as the product of the global weight times the two local weights (one pertaining to each of the documents).

The global weight of a term is greater for the less frequent terms. This is reasonable because the presence of a term that occurred in most of the documents would really tell one very little about a document. On the other hand, a term that occurred in only 100 documents of one million would be very helpful in limiting the set of documents of interest. A word that occurred in only 10 documents is likely to be even more informative and will receive an even higher weight.

The local weight of a term is the measure of its importance in a particular document. Generally, the more frequent a term is within a document, the more important it is in representing the content of that document. However, this relationship is saturating, i.e., as the frequency continues to go up, the importance of the word increases less rapidly and finally comes to a finite limit. In addition, we do not want a longer document to be considered more important just

because it is longer; therefore, a length correction is applied. This local weight computation is based on the Poisson distribution and the formula can be found in Lin J and Wilbur WJ.

The similarity between two documents is computed by adding up the weights (local wt1 × local wt2 × global wt) of all of the terms the two documents have in common. This provides an indication of how related two documents are. The resultant score is an example of a vector score. Vector scoring was originated by Gerard Salton and has a long history in text retrieval. The interested reader is referred to Salton, Automatic Text Processing, Reading, MA: Addison-Wesley, 1989 for further information on this topic. Our approach differs from other approaches in the way we calculate the local weights for the individual terms. Once the similarity score of a document in relation to each of the other documents in the database has been computed, that document's neighbors are identified as the most similar (highest scoring) documents found. These closely related documents are pre-computed for each document in PubMed so that when you select Related Articles, the system has only to retrieve this list. This enables a fast response time for such queries.

Appendix A: Project Webpage
All text-based datasets are freely available at http://sci.slis.indiana.edu/sts. They comprise:

Raw Data

List of PMIDs	sts-pmids.txt.gz (5MB)
List of stop words	sts-stop-words.txt.gz (4KB)

Analysis Input Data

Title/Abstract term adjacency list	sts-text-adj.gz (642MB)
MeSH adjacency list	sts-mesh-adj.gz (98MB)

Analysis results will also be made available from this site. They will comprise:

Analysis Result Data

Linkage-based analysis

Co-citation	sts-cocite-sim.gz	sts-cocite-clust.gz (56MB)
Bibliographic coupling	sts-bibcoup-topn.sim.gz (121MB)	sts-bibcoup-clust.gz (9.1MB)
Direct citation	sts-directcit-topn.sim.gz (67MB)	sts-direct-clust.gz (8.8MB)

Title/Abstract term analysis

Co-occurrence	sts-TA-co.sim	sts-TA-co.clust
LSA	sts-TA-lsa.sim	sts-TA-lsa.clust
Topic model (IU)	sts-TA-topics-iu.sim	sts-TA-topics-iu.clust
Topic model (UCI)	sts-TA-topics-uci.sim.gz (117MB)	sts-TA-topics-uci.clust

Self-organizing maps (SOM)	sts-TA-som.sim	sts-TA-som.clust
Collexis	sts-TA-collx-topn.sim.gz (146MB)	sts-TA-collx.clust

MeSH analysis

Co-occurrence	sts-mesh-co.sim.gz (155MB)	sts-mesh-co.clust
LSA	sts-mesh-lsa.sim	sts-mesh-lsa.clust
Topic model (IU)	sts-mesh-topics-iu.sim	sts-mesh-topics-iu.clust
Topic model (UCI)	sts-mesh-topics-uci.sim	sts-mesh-topics-uci.clust
Self-organizing maps (SOM)	sts-mesh-som.sim	sts-mesh-som.clust
Collexis	sts-mesh-collexis.sim	sts-mesh-collexis.clust

Other analysis

Collexis (full engine on raw MEDLINE data)	sts-collexis-full.sim	sts-collexis-full.clust
NCBI related records data	sts-ncbi-topn.sim.gz (115MB)	sts-ncbi-topn.clust