text comparison based on dynamic programming

Alexander Pertsemlidis
and Harold R. Garner

Inspired by BLAST and related sequence comparison algorithms, we have developed a method for the direct comparison of query text against database text as an improvement upon traditional keyword-based searches. The primary application of our implementation, eTBLAST, is to better select those database entries (abstracts, in the case of MEDLINE) of most relevance to a given query. eTBLAST takes as input natural text instead of keywords, allows refinement of retrieved hits through iteration, can be applied to any text (demonstrated here on biomedical databases), and allows inspection of the local space around a query through simple visualization methods. eTBLAST is available at http://invention.swmed.edu/etblast.

Introduction

A recent enumeration of molecular biology databases listed 548 different databases, including sequence repositories, databases of comparative genomics, gene expression, gene identification and structure, genetic and physical maps, intermolecular interactions, metabolic pathways and cellular regulation, mutations, protein sequence and structure, protein motifs, and RNA sequences [1]. Most of these databases are growing at an exponential rate. In most cases, this growth represents the generation of information, but not automatically the generation of knowledge. Much as knowing how to spell a word is not equivalent to understanding its meaning, and knowing all the words in a language does not mean we understand its grammar, knowing the sequence of an individual gene is not equivalent to understanding its function, and knowing all of the genes and their protein products for a given organism does not mean we know how they interact to create life.

A recent article coauthored by Harold Varmus, former director of the National Institutes of Health, called for a database similar to Genbank, but containing articles instead of sequences [2]. The problem of information retrieval from literature databases is therefore becoming more and more important. Searches of text databases are typically done by Boolean methods, in which the user inputs a series of single words connected by Boolean operators, such as AND, OR, and NOT. While this approach is useful for the experienced or sophisticated searcher who has a clear understanding of the query and of the limitations of the database against which he is searching, poorly constructed queries result in either too many hits, with a low signal-to-noise ratio, or an overly restricted result with few hits, if any. Relevant hits may be missed entirely because of query constraints, such as the failure to include relevant synonyms.

More advanced approaches, which are available in fields other than biomedical research (typically ones for which language use is more structured and less ambiguous, like law for which Lexis/Nexis and WestLaw are the standard information retrieval tools) go beyond Boolean operators and include search modifiers and proximity as part of the query construction. Search modifiers include stemming, in which a word is replaced by its base form (e.g., walked → walk), case folding, which is simply the conversion of characters in order to make the search case insensitive (e.g., Queen → queen), application of a stoplist, which involves the removal of words felt to be insignificant for the comparison (e.g., the queen → queen), synonym expansion or replacement, in which a word is expanded to a list of synonyms (e.g., monarch → king, queen), and semantic network expansion or replacement, in which a word is expanded to include conceptually related terms (e.g., queen → woman). Including proximity in query construction allows the user to specify a distance relationship between words in the query which must be satisfied by documents in the target. For example, AT/1, NEAR/4 and FAR/10 represent constraints on two words satisfied only if they are exactly one word apart, at most four words apart and at least ten words apart, respectively.

This may sound like biology is getting the short end of the stick, but this is not really the case. Biologists deal with many types of (biological) objects, from organelles to organisms, from nucleotides to networks. The purpose is sometimes reductionist and sometimes integrationist, but in either case, the ultimate goal is the determination of structure and function and how they are related. Bioinformaticists deal with only three types of objects: sequences, both nucleotide and amino acid, structures, from small molecules to protein complexes, and text, whether it is annotation for a sequence or a structure or an article or abstract. The purpose is the correlation of information and the use of that information to assign function to new sequence and to predict structures and construct models. The principal tools are based on similarity search, like BLAST, FASTA, and SSEARCH [3] for nucleotide and amino acid sequence comparison, and VAST [4], [5] and DALI [6] for structure comparison. Unfortunately, there are currently no tools available for text comparison that allow a user to specify the query.

Comparison Methods, Similarity and Homology

Fortunately, the concept of similarity search extends far beyond comparison of nucleotide sequences and amino acid

 0739-5175/04/$20.00©2004IEEE

sequences, to protein and other molecular structures, and even to text, parallel to the relationship between sequence, structure, and function. While similarity between documents is trivially measured by enumerating the words they have in common with an adjustment for different document lengths, in which the lowest frequency words are discarded insignificant [7], it is possible to borrow from the work done on sequence comparison and improve the assessment of similarity significantly.

Before the similarity of two texts can be computed, however, their proper alignment must be determined, an inherently circular problem, since evaluating an alignment requires calculating similarities. The question "How similar are two texts?" is not as simple as it seems. It is, in fact, several questions: "Is there a perfect match between the two texts?," "If there is no perfect match, what is the best alignment between the two texts?," "How should alignments be scored?," and "If gaps are allowed, how should they be scored?" [8]

Substitution Matrices and Gap Penalties

When evaluating an alignment of any kind, one would like to know how meaningful it is. For sequences, this requires a scoring matrix, or a table of values that describe the probability of a biologically meaningful amino acid or nucleotide residue pair occurring in an alignment. Typically, when two nucleotide sequences are being compared, all that is being scored is whether or not two bases at a given position are the same. All matches are given the same score (typically $+1$ or $+5$), as are all mismatches (typically -1 or -4). But with proteins, the situation is different. Substitution matrices for amino acids are more complicated and implicitly take into account everything that may affect the frequency with which any amino acid is substituted for another, such as the chemical nature and frequency of occurrence of the amino acids. The objective is to provide a relatively heavy penalty for aligning two residues together if they have a low probability of being homologous, or correctly aligned by evolutionary descent. There are two major forces that drive the amino acid substitution rates away from uniformity: not all substitutions occur with the same frequency, and some substitutions are less functionally tolerated than others and are therefore selected against. Commonly used substitution matrices include the blocks substitution (BLOSUM) [9] and point accepted mutation (PAM) [10], [11] matrices. Both are based on taking sets of high-confidence alignments of many homologous proteins and assessing the frequencies of all substitutions, but they are computed using different methods.

In the case of text, the scoring matrix is a table of values that describe the probability of a semantically or syntactically meaningful alignment. In principle, it is possible to generate high-confidence alignments based on techniques like grammar induction [12] and to assess the frequencies of substitutions in those alignments.

For text, as with sequences, we must consider not only substitutions but insertions and deletions as well. The consequence with respect to sequence alignment and comparison is the need for the introduction of gaps into one or both sequences or texts to produce a proper alignment. The penalty for the creation of a gap should be large enough that they are introduced only where needed, and the penalty for extending a gap should take into account the likelihood that insertions and deletions can occur several units at a time.

Table 1. Examples of Boolean operators, search modifiers, and proximity operators.

Boolean operators		
Operator	**Example**	**Result**
AND	Alice AND Wonderland	both terms
OR	Alice OR Wonderland	either term or both
NOT	Alice NOT Wonderland	first term but not second
Search modifiers		
Modifier	**Example**	**Result**
stemming	walked	walk
case folding	Queen	queen
stoplist	The queen	queen
synonyms	monarch	queen
semantic network	queen	woman
Proximity operators		
Operator	**Example**	**Result**
AT/1	Cheshire AT/1 Cat	Cheshire Cat
NEAR/4	head NEAR/4 off	"Off with her head."
FAR/10	Queen FAR/10 Hatter	. . . the Queen put on her spectacles, and began staring at the Hatter, who turned pale and fidgeted.

Dynamic Programming

Dynamic programming methods were first described in the 1950s, outside the context of bioinformatics, and first applied in that context by Needleman and Wunsch in 1970 [13]. These methods find an optimal solution to a given problem by breaking the original problem into smaller and smaller subproblems, until the subproblems have a trivial solution, and then using those solutions to construct solutions for larger and larger portions of the original problem. A generalization of the

recursive dynamic programming approach, the Smith-Waterman algorithm [14], is a mathematically optimal method, which handles sequence comparisons in a single computation and is guaranteed to find the highest scoring alignment. The algorithm incorporates the concepts of mismatches and gaps and identifies optimal local alignments. Local alignments, where parts of one sequence are aligned to parts of another, are more biologically relevant than global alignments, where entire sequences are aligned to each other because perfect matches of significant length are extremely rare in most biological applications. Interestingly, local alignments are also more relevant for text comparison, where perfect matches of significant length are also rare (except in cases of plagiarism).

As fast as computers are, and as efficient as the dynamic programming algorithms are, they are still far too slow to enable exhaustive searches of huge nucleotide sequence repositories such as GenBank [15] or amino acid sequence databases like SWISS-PROT [16], or even literature databases like MEDLINE. This means that one cannot search the database exhaustively for the optimal local alignment. A heuristic is therefore required to shorten the time required by approximating the best local alignment.

Text comparison algorithms can take advantage of the same heuristics employed by sequence comparison algorithms. The sequence comparison algorithms BLAST and FASTA both operate on the assumption that true matches are likely to have at least some short stretches of high-scoring similarity, but where FASTA looks for exactly matching words, BLAST uses a scoring matrix (BLOSUM62 for amino acid sequences, by default) to find high-scoring words. These high-scoring hits are used as seeds for the slower dynamic programming algorithm. Analogously, text comparison can begin with differentiating keywords, which are useful in similarity judgments, from "stop words," which are not. The documents in the target database that contain enough of the keywords present in the query can then be subjected to the slower dynamic programming algorithm.

BLAST for Electronic Text (eTBLAST)

If we define a segment as a contiguous subsequence of text, and a segment pair as a pair of segments of the same length, one from each of the two texts being compared, then the task that eTBLAST performs is the identification of all pairs of similar segments whose score exceeds a given threshold. These pairs of similar segments are called "high scoring segment pairs" (HSPs). The segment pair with the highest score is called the maximal scoring segment pair (MSP); its alignment cannot be improved by extending it or shortening it.

eTBLAST is similar to the method used for document neighbor computation by the National Center for Biotechnology Information (NCBI) (see the Appendix), but can be applied not only to text abstracts contained in MEDLINE but to arbitrary text provided by the user. The keywords in the query and in the target database are identified by negation; they are the remainder after removal of terms that are contained in a given stop list. For the query, lexical variants and synonyms are generated. The resulting keyword list is compared to the extracted keywords from the target database, and the score of a single pairwise comparison is calculated either using the vector cosine method, or without normalization.

There are three major steps in the eTBLAST algorithm (Figure 1), which compares the BLAST and eTBLAST algorithms. Detail for each of the steps is as follows:

1a) Where BLAST filters low complexity regions (CA repeats, for example) and removes them from the query and generates a list of all of short sequences, or words, that make up the query, eTBLAST removes stop words (such as "a" and "the") from the query and generates a list of all keywords and their frequencies of occurrence within the query. We recognize that a single set of stop words cannot cover all possible applications or circumstances and have therefore generated several sets. One such set was generated by parsing MEDLINE abstracts and tabulating the top 5,000 most abundant words. Processing all abstracts in MEDLINE through April 2001, we extracted ~4,493,000 unique words. Titles and abstracts were read in for each record and preprocessed to eliminate delimiting punctuation (colons, commas, semicolons, and periods that were followed by spaces—to ensure the punctuation was not part of the word). Brackets, parentheses, linefeed characters, and quotes (single and double) were eliminated entirely, along with words containing no letters (i.e., purely numeric (e.g., 4, 2.5) or symbolic (e.g. ,– >, 4/5) references). Words were read into an array, sorted, merged, and summed. Both absolute word frequency and word frequency by abstract are calculated. The list was truncated at approximately 2,000 words, since inspection of the list indicated that words appearing less frequently were more likely to be useful keywords. A few of the words occurring frequently enough to be in the top 2,000 were removed from the stop list, as they were biomedically relevant keywords, like "cancer," "drug," and "patient." Other keyword lists were generated from less biologically centered sources, such as the Merriam Webster dictionary.

1b) BLAST uses a scoring matrix (BLOSUM62, by default, for amino acids) to determine all high-scoring matching words for each word in the query sequence. No gaps are allowed. The list of matches is reduced by taking only those that will score above a given threshold, called the neighborhood word score threshold. eTBLAST takes the list of keywords from the query and makes some adjustments in order to handle the situation in which keywords in the

query and the target are similar in meaning but do not match exactly, and the related situation in which keywords have the same meaning but differ in their grammatical usage. To accomplish this, we employed lexical variant (LVG) and synonym generation. One implementation of LVG is a suite of tools that can "generate, mutate and filter" lexical variants, which is part of the SPECIALIST set of lexical tools available from the Unified Medical Language System (UMLS) Knowledge Sources at the National Library of Medicine. (See http://umlsks.nlm.nih.gov.)

Since the amount of computation required to compare a single query abstract to the MEDLINE abstract database is already significant, and the addition of lexical variant generation makes it larger, we decided to minimize its impact by applying LVG to the query only. We expand each keyword in the query into its set of lexical variants, and at the same time, add keywords that are synonymous with those in the original list. The resulting (larger) list of keywords is then used in the comparison against the target database.

2) BLAST searches through the target sequence database for exact matches to the word list generated. Because BLAST has already preprocessed and indexed the databases for the occurrence of all words in each sequence in the database, this search is extremely fast. If a match is found, it is used to seed a possible alignment between the query and the databases sequences. eTBLAST searches through the target text database for exact matches to the keyword list generated. Because eTBLAST has already preprocessed and indexed the databases for the occurrence of all words in each text in the database, this search is extremely fast. If a match is

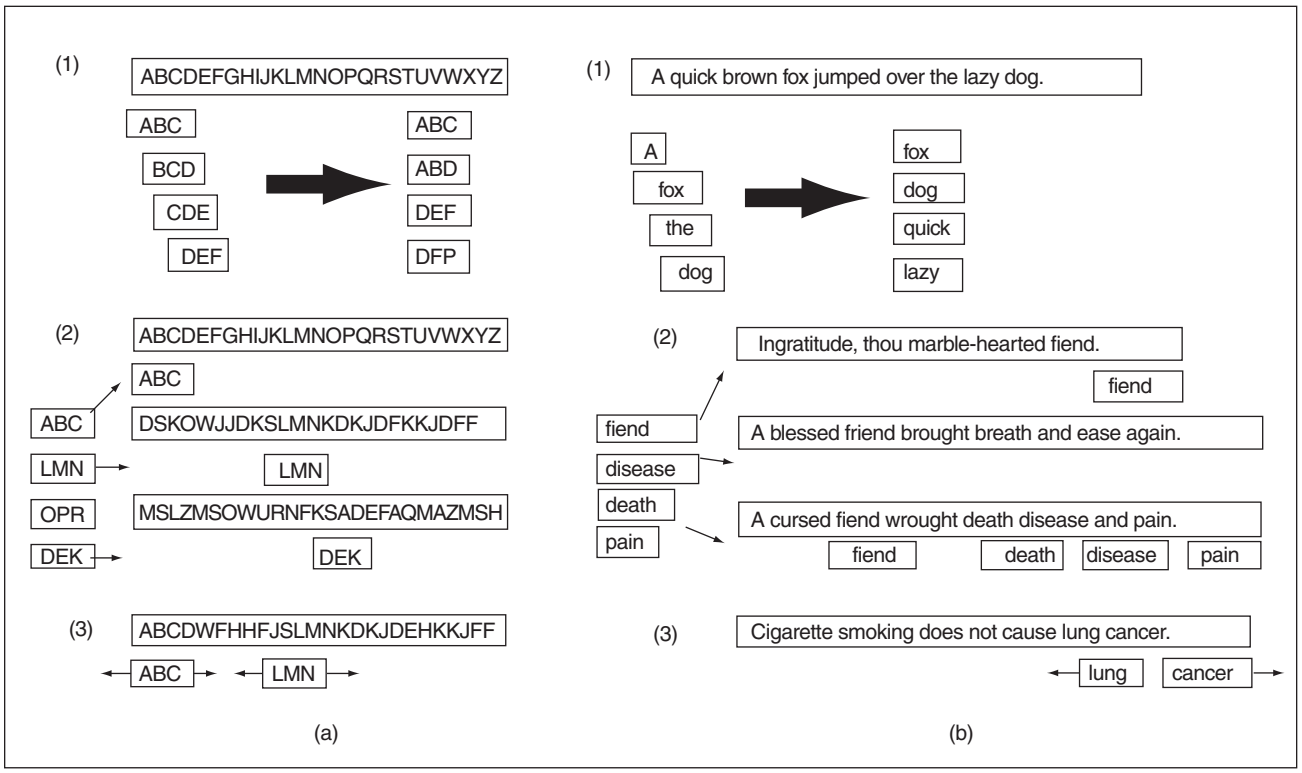

Fig. 1. (a) The BLAST algorithm. (1) Given a query sequence of length L, BLAST derives a list of words of length w, where $w = 3$ for amino acid sequences and 11 for nucleotide sequences. There are at most $L - w + 1$ such words. The word list is reduced by removing those that score less than a threshold T when scored using a scoring matrix such as PAM250 (10) or BLOSUM62 (9). For typical parameter values, this results in about 50 words per residue of the query sequence. (2) The high scoring word list is compared to the sequence database and exact matches are identified. (3) For each word match, the alignment is extended in both directions to generate alignments that score higher than the score threshold S. (b) The eTBLAST algorithm. (1) Given a query sequence of length L words, eTBLAST generates a list of keywords, removing those that appear in a given stop list. (2) The keyword list is compared to each abstract in the database and exact matches are identified. The initial score is calculated based on the frequency of co-occurrence of the keywords in the query and the target. (3) For each keyword in the high scoring abstracts, the local neighborhood of keywords is computed via a distance matrix, and the abstracts are rescored based on the distributions of keywords in the query and the target.

found, it is used to seed a possible alignment between the query and the database sequences.

3a) BLAST tries to extend the alignment from the matching words in both directions as long as the score continued to increase. The resulting alignment was called an HSP. eTBLAST simply looks for the highest value in the dynamic programming matrix and determines the optimal local alignment from that point.

3b) BLAST determines if each score found is greater than a given cutoff score, *S*, determined empirically by examining the range of scores given by comparing random sequences and then choosing a value that is significantly greater. The MSPs from the entire database are identified and listed. BLAST determines the statistical significance of each score by calculating the probability that two random sequences, one the length of the query sequence and the other the length of the database (the sum of the lengths of all of the database sequences) with the same composition (nucleotide or amino acid) could produce the calculated score. eTBLAST is not as ambitious at this level and only uses the scores from the dynamic programming step to rerank the hits returned from the keyword-based comparison.

We have also implemented an iterative process in which the user can select from the initial set of returned abstracts to create a more accurate profile for a subsequent search against the target database. As in PSI-BLAST, a profile generated from significant hits found in round i is used for round $i + 1$.

Clustering and Visualization

One of the frustrations many users have with search engines is that the results are typically returned in a ranked list form whether or not that is the most appropriate format for the data and independent of the underlying structure and relationships. For example, a Web search on the word "breast" will return hits to pages on cancer, Britney Spears, swimming, and chicken, combined in a single list, with no indication of how entries in that list are related to each other. One simple, and perhaps simplistic, approach to making the set of returned results more useful is to provide an indication of how individual results are related to each other. In the case of biomedical abstracts, this can be done by calculating a social network [17] for the set of abstracts, in which similarity scores are calculated for every pair of abstracts. The resulting distance matrix represents a set of constraints that are only guaranteed to be fulfilled in a space of dimensionality corresponding to the number of abstracts. There are several good methods (principal component analysis, correspondence analysis, or more recently, local linear embedding [18], [19]) that reduce the dimensionality of the space to something easier to visualize (like three dimensions) while maximally satisfying the constraints imposed by the distance matrix. Once the individual results are assigned points in three-dimension space, the results can be displayed using a standard viewer for biological structures, like MAGE [20], [21]. Please see http://chaos.swmed.edu/frisc/mage for an example of our application of this viewer to text similarity data.

Discussion

The real value of our implementation of text comparison based on dynamic programming and grammar induction will not come from its utility as a new document clustering (search) tool, but from its ability to identify hidden connections within the literature as a whole. This is especially true for connections discussed by different authors using different writing styles and word choice, and for connections hidden because they are linked only through other databases (acronyms, gene sequence similarity, or protein structure/function similarity).

At the moment, we only have tools for the comparison of sequences and structures, and some simple tools for the comparison of biological abstracts. We have no tools for the comparison of organelles, or cells, or organisms, although as the bioinformatics toolkit grows, we will begin to take steps in those directions. We will accomplish this by being able to look not only at sequence similarity, structural similarity, or textual similarity individually, but at the relative arrangement of similar sequences, structures, and sentences, and not only at sequence, structure, or textual similarity alone, but at all of them together.

All of this is part of a paradigm shift away from hypothesis driven research, in which we ask a single question and receive a single answer, to data driven research, in which we are given many answers at once and seek to find the hypothesis that best explains them and tells us something interesting at the same time. Instead of generating more and more complex queries to extract knowledge from these data sets, we use data mining to uncover patterns.

This gets us back to the principal computational tool of biologists, the similarity search. There are two central ideas to the idea of elucidating the local space around a given query, neighboring (which is the central idea behind BLAST [22] and related algorithms), and iteration (which is the central idea behind PSI-BLAST [23] and related algorithms). These are usually applied entirely within a single database. Data mining leads us immediately to the possibility of adding the idea of crossing database boundaries for a given search, which is promising for two reasons: what is adjacent/neighboring in one database may not be in another, and what is sparsely populated in one database may be densely populated in another. There is already some evidence that even a limited application of this idea improves the accuracy of the similarity search [24]. The only problem is that, while sequence databases are well populated, structure and function (literature) databases are much less so. This means that there may not yet be sufficient information to provide large gains in accuracy. At the same time, the philosophy that biology can now afford to adopt is a Bayesian one, that includes all available information, of whatever nature, because even an incremental increase in our ability to assess similarity will bring understanding that much sooner.

Acknowledgments

This work was supported by the P.O'B. Montgomery Distinguished Chair in Developmental Biology and the Hudson Foundation. We wish to thank Pieter Adriaans, for helpful discussions, and Natalie Prikhodko and Shuping Ai, for programming assistance.

Address for Correspondence: Harold R. Garner, Eugene McDermott Center for Human Growth and Development, University of Texas Southwestern Medical Center, 5323 Harry Hines Boulevard, Dallas, Texas 75390, USA.

References

[1] M. Galperin, "The molecular biology database collection: 2004 Update," *Nucleic Acids Res.*, vol. 32, pp. 3–22, 2004.
[2] R.J. Roberts, et al., "Information access. Building a "GenBank" of the published literature," *Science*, vol. 291, pp. 2318–2319, 2001.
[3] W.E. Pearson and D.J. Lipman, "Improved tools for biological sequence comparison," *Proc. Nat. Acad. Sci. USA*, vol. 85, pp. 2444–2448, 1988.
[4] T. Madej, J.F. Gibrat, and S.H. Bryant, "Threading a database of protein cores," *Proteins*, vol. 23, pp. 356–369, 1995.
[5] J.F. Gibrat, T. Madej, and S.H. Bryant, "Surprising similarities in structure comparison," *Curr. Opin. Struct. Biol.*, vol. 6, pp. 377–385, 1996.
[6] L. Holm and C. Sander, "Dali: A network tool for protein structure comparison," *Trends Biochem. Sci.*, vol. 20, pp. 478–480, 1995.
[7] C.J. van Rijsbergen, *Information Retrieval*. Butterworths, London, 1990.
[8] A. Pertsemlidis, and J.W. Fondon, III "Having a BLAST with bioinformatics (and avoiding BLASTphemy)" *Genome Biol.,* vol. 2, 2001, REVIEWS- 2002.
[9] S. Henikoff and J.G. Henikoff, "Amino acid substitution matrices from protein blocks" *Proc. Nat. Acad. Sci. USA*, vol. 89, pp. 10,915–10,919, 1992.
[10] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt, in *Atlas of Protein Sequence and Structure*. M.O. Dayhoff, Ed. Washington, DC: National Biomedical Research Foundation, 1978, pp. 345–352.
[11] D.J. States, W. Gish, and S.F. Altschul, "Improved sensitivity of nucleic acid database searches using application-specific scoring matrices," in *METHODS: A Companion to Methods in Enzymology*, vol. 3, 1991, pp. 66–70.
[12] P. Adriaans and E. Haas, in *Proc. 1st Workshop Learning Language Logic*, J. Cussens Ed., Bled, Slovenia, 1999, pp. 117–127.
[13] S.B. Needleman and C.D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, pp. 443–453, 1970.
[14] T.F. Smith and M.S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.
[15] D.A. Benson, et al. "GenBank," *Nucleic Acids Res.*, vol. 28, pp. 15–18, 2000.
[16] A. Bairoch and R. Apweiler, "The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000," *Nucleic Acids Res.*, vol. 28, pp. 45–48, 2000.
[17] L. Freeman, "Visualizing social networks," *J. Social Structure*, vol. 1, pp. 1–15, 2000.
[18] S.T. Roweis and L.K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2336, 2001.
[19] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2322, 2001.
[20] D.C. Richardson and J.S. Richardson, "The kinemage: A tool for scientific communication," *Protein Sci.*, vol. 1, pp. 3–9, 1992.
[21] D.C. Richardson and J.S. Richardson, "Kinemages—simple macromolecular graphics for interactive teaching and publication," *Trends Biochem Sci.*, vol. 19, pp. 135–138, 1994.
[22] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, pp. 403–410, 1990.
[23] S.F. Altschul, et al. "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.
[24] J.T. Chang, S. Raychaudhuri, and R.B. Altman, "Including biological literature improves homology search," in *Pac. Symp. Biocomput*, 2001, pp. 374–383.
[25] W.J. Wilbur and Y. Yang, "An analysis of statistical term strength and its use in the indexing and retrieval of molecular biology texts," *Comput. Biol. Med.*, vol. 26, pp. 209–222, 1996.
[26] G. Salton, *Automatic Text Processing*. Reading, MA: Addison-Wesley, 1989.

Appendix

Computation of Related Articles (NCBI)

NCBI eliminates 310 common words from consideration, does some stemming, and categorizes each word as text words, title words, or MeSH terms. Of course, as NCBI recognizes, "not all words are of equal value." Each word is assigned a global weight based on the number of documents in the database that contain the word, and an estimate of the importance of the word in producing relationships in the database [25]. For a given pairwise comparison of two documents, each word is also assigned a local weight, based on the number of times the term appears in a given document. The overall weight of a word, for the purpose of comparing two documents, is the product of the global weight and the local weight in each of the documents. The overall similarity of two documents is calculated by adding up all of the weights of the words that the two documents have in common, and normalizing by the product of the lengths of the two documents, producing what is called a vector cosine score [26]. For more detail, see "Computation of related articles" at http://www.ncbi.nlm.nih.gov/entrez/query/static/overview.html.