

BioKnOT

Biological Knowledge through Ontologies and TFIDF

Mehmet Dalkilic*
School of Informatics
Indiana University
901 East 10th Street rm. 229
Bloomington, IN 47408-3912
*contact author
dalkilic@indiana.edu

James Costello
School of Informatics
Indiana University
901 East 10th Street rm. 235
Bloomington, IN 47408-3912
jccostel@indiana.edu

ABSTRACT

The inundation of biological data has produced enormous amounts of information in the form of research articles. The repositories for these articles have become so abundant and so excessively populated that searching for a timely and relevant article has become near impossible. BioKnOT attempts to address this issue and allows for efficient and effective information retrieval. This system implements an iterative refinement of search building upon semantic relevance, with consideration to citation frequencies. It does this by constructing ontologies from term relationships based on words determined by existing term-frequency-inverse-document-frequency (TFIDF) strategies, while including a means of comparing ontologies using scoring matrices that consider pairs of words in and among sentences. BioKnOT will address the demand for sifting through the copious amounts of present and an ever increasing number of biologically related research articles. In this paper, we discuss the theory and intuitions underlying our system and its implementation.

1. INTRODUCTION

Our ability to generate, collect, and cheaply store biological data at an ever increasing rate has created a need for tools that allow users to efficiently and effectively navigate this information to both aid (and increasingly discover) in research. Various approaches to this problem are currently and actively being studied [9] including information retrieval [2], web mining, and ontologies [11, 4, 10]. Through work on a related problem—managing protein family annotations [3]—we discovered that the kinds of information being sought by biologists who are using our system was too broad in content and was insensitive to timeliness *i.e.* the date of publication. With this in mind, we set out to create the system, **B**iological **K**nowledge through **O**ntologies and **T**fidf

(BioKnOT), that allows users to quickly and easily “drill-down” on a topic that, at the same time, will provide timely content. BioKnOT also provides a means of iterating over the set of documents allowing the user to refine the specificity of a search. Formally, we have a set of documents D and seek an ordering d_1, d_2, \dots, d_k , where d_i is more interesting and useful to the user than d_{i+1} .

Our system builds on the well-known, successful information retrieval technique of term-frequency-inverse-document-frequency (expanded upon in section 5.1), which identifies words that are likely to be significant and meaningful, denoted throughout this paper as the set of terms, T_f . Initially, the user is prompted for a Boolean search on terms. Those documents of D for which the Boolean function is true are used for T_f generation. Currently, we have set the size of T_f at 50, though this will be made adjustable in the future. From T_f a scoring matrix S_{ij} (expanded upon in section 5.3) is created that indicates numerically whether pairs of words i, j , where $i, j \in T_f$, that occur within a certain reading frame of no more than 20 words (arrived at experimentally), are likely to be present other than by chance. The value is actually a log-odds ratio comparing observed frequency to, in this case, a random model, which is defined as the set of documents that meet the Boolean search criteria. Scoring matrices are a universal tool in sequence alignments techniques that allow for disparate, though related molecules, to be substituted for one another in a sequence of molecules, and therefore, allow for non-identical sequences to be compared. What is unusual about our scoring matrix is that it is likely that $S_{ij} \neq S_{ji}$ due to the natural structure of language. The scoring matrix [7] is used to compare ontologies generated from the abstracts of the documents with an ontology constructed from user inputs.

An ontology [1] is a means of capturing relationships among entities—generally via a directed graph—where the nodes represent entities and an edge represents a kind of relationship (expanded upon in section 5.2). Our ontology is meant to capture pairs of words that occur near each other both in a sentence (intra-sentence) and among sentences (inter-sentence). Our motivation in seeking this distinction was that, for example, sentences like the ones below are very likely to have different meanings with respect to the relationship between the two words *mitochondria* and *permeability*:

“... and is not present in the *mitochondria*. *Permeability* is another...”

“... *mitochondrial permeability* is an important aspect of apoptosis...”

Though both words are important individually, to some degree, the way they are presented in and among sentences offers us better insight on their relationship to each other.

After the user is asked to choose the most relevant terms from T_f to the search, denoted T_u , where $T_u \in T_f$, he or she has the option of proceeding through BioKnOT by entering more information or simply doing a “quick search” and going directly to the results page.

If the user wants to enter more information for a more precise search, then after selecting the most relevant terms, T_u , the user will be prompted to enter a statement describing what type of article is being searched for, which will contain words from T_u that are then used to create a user defined ontology. This ontology is critical to BioKnOT because it provides a means of capturing semantically related terms *with respect to the user* and is used to order the relevant documents’ ontologies using S_{ij} . The user then is queried for how strong the relationships should be, owing to the fact that not enough information is provided by one or several sentences. All of this provided data is then used to score the articles within the article database.

The “quick search” option allows the user to bypass entering any more information after terms T_u are selected. The ontology that would have been created from the user’s input is derived by considering all term relationships from T_u with a distance of 1 (figure 5). This ontology will be used to score the articles within the article database.

Once a set of documents has been returned, the user can refine the search by selecting the documents that are most closely related to the user’s ultimate search goals. A new T_f is created and the process continues as before. Choices that the user made in the previous search are used as default values and can be, in the case of the user’s statement, either added to or modified.

Additionally we provide a kind of support through a second scoring mechanism that takes into account the number of times a document has been cited, weighting the more recently cited papers more. Thus, documents have a pair of values $[r, s]$, where r is what we call the semantic relevance taken from the scored ontologies and s the support taken from the citation information. 1

The remainder of this paper is as follows. In Section 2, we present a description of the system architecture of BioKnOT. In Section 3, a description of the program architecture and some implementation details are discussed. In Section 4, we discuss the tools and steps used to select and populate the article database. Section 5 discusses the methods used in the scoring of an article. In section 6, we summarize BioKnOT and briefly discuss future work and challenges.

2. SYSTEM ARCHITECTURE

The system architecture (figure 1) consists of four main components. First is the server, which is apache and run on the machine, B1OKDD, and is hosted at the School of Informatics at Indiana University (<http://www.informatics.indiana.edu/>). The second component is the client, which uses a web interface to talk to the server through CGI and Perl. Third is the article database, which is where all and any

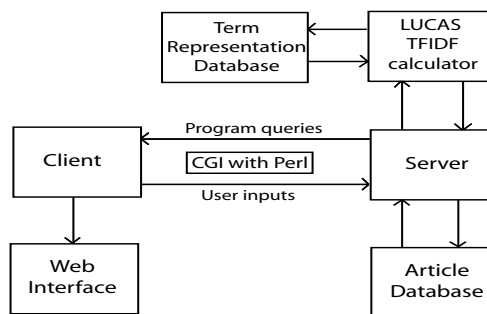


Figure 1: This system allows the client to interact with the server through a web interface. The server works directly with the client, the article database, and LUCAS. The article database holds all the article data, such as title, author(s), abstract, etc. LUCAS is a web service that does TFIDF calculations for a supplied file. LUCAS interacts directly with a term representation database, which contains millions of term relationships taken from millions of indexed web pages.

of each individual article’s information is stored. The server interacts with the database a great deal. Lastly, there is LUCAS (<http://lair.indiana.edu/research/lucas/index.html>), which talks to a term relationship database and is a piece of software developed at Indiana University’s School of Library and Information Science (<http://www.slis.indiana.edu/>) by Javed Mostafa and Yueyu Fu. This web service is called to make the TFIDF calculations (section 5.1).

3. PROGRAM ARCHITECTURE

The program (figure 2) consists of 5 core interfaces. The first is the initial query page. This is where a user can enter Boolean search terms and the program will do a search for those terms in the abstracts and titles of all the articles in the database.

After the initial set of articles has been constructed from the Boolean search, the abstracts from all these articles are placed in a text file and passed off to LUCAS (section 5.1) for the TFIDF rankings of the top 50 words.

On the next page, the user will be shown a filter, which contains the top 50 words returned by LUCAS. The user is asked to select the terms that are relevant to their search. These terms are stored and the user can then select the “quick search” option, which will bring them directly to the results page, or the user can enter more data for a more precise search. If the user wants to enter more data, then after the filter page, the user is prompted to enter several sentences on what type of article is needed.

The program then constructs an ontology of the term relationships between the selected terms in the user’s statement. This ontology will be the standard of comparison for all other articles.

When the term relationships are found, the user is then prompted to add weight to these relationships based on how important they are to the search.

After all this data has been interactively collected, ontologies of each articles’ abstract are created and then scored against the user defined term relationship ontology.

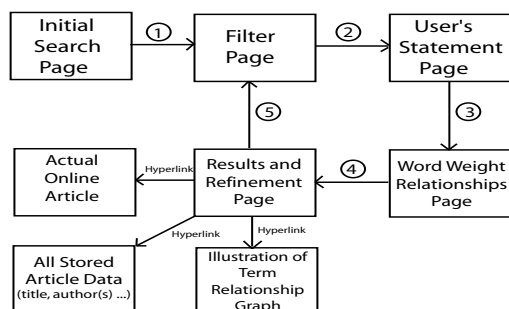


Figure 2: This is an illustration of the flow of the program. First the user enters a query on the initial search page. (1) A TFIDF calculation is done on initial searched documents and the user is asked to rank these terms on the filter page. Next (2), the user is asked to enter a few sentences stating what type of article is being looked for. (3) The scoring matrix is built and term relationships are constructed, and then the user is asked to supply these relationships with a score. (4) All the articles are scored and returned based on rank to the results page, which supplies the user with article data, illustration of the term relationships, and the URL to the article itself. The results page also serves as the refinement page, which (5) allows the user to iterate over the search with more specific data based on selected articles, instead of a random model.

The user is then shown the top ranked articles as determined by semantic score. Along with the semantic score, a citation rate is supplied to provide the user with some more input on how important an article has been over the years.

In addition to the scores, three important hyperlinks are provided. First, a link to the actual online article is given. If the online article could not be accessed, then the user is directed to the PubMed abstract page of that article. Next, a link to all the data in the database related to that article is provided, such as title, author(s), abstract, etc. Lastly, a link to the term relationship ontology is provided (figure 4). This ontology will show the user how the terms are related to each other in the abstract of the article, as well as supplies the user with a way to visually compare article semantic relationships.

The overall concept behind BioKnOT is to supply the user with an effective interactive way to find articles related to very specific search criteria. Knowing this, one of the most important features of BioKnOT is to allow the user to do an iterative search over the article database with more strictly defined search criteria.

The results page also provides the user with a means to further narrow down a search by selecting documents that are related to the user's specifications. After these documents have been selected, LUCAS is called again, but instead of the large broad sample set that was used on the first pass of the search, a very specific user selected set of documents are used to create the term filter page.

The process is then started all over again and can be run indefinitely.

BioKnOT saves the state of your previous searches and passes that information along to the scoring of the articles. The user is supplied this information and can change it at

any time during the search.

4. POPULATING THE ARTICLE DATABASE

Much consideration was given to selecting data to populate the article database. There are many sources of information available that supply portions of the data needed to run BioKnOT, but nothing that has everything. Consequently, several tools and sources were used to populate the database.

The following article related information is needed:

- title
- author(s)
- abstract
- publication date
- journal
- article URL
- citation count
- reference list
- some kind of accession number
- Mesh keywords

No single source supplied all the required fields, but ISI's Web of Science (<http://www.isinet.com/products/citation/wos/>) supplied all the fields required, with the exception of an article URL.

The article information was easily exported from ISI's Web of Science to a text file, which was then parsed to extract and insert the appropriate data into the article database.

The article URL was found by searching NCBI's PubMed database through a web bot. The bot finds the article URLs by taking the first 10 words of an article's title and executing a PubMed search. The results are then parsed, with any matches to the article title being pulled aside and the article's author(s) are checked to confirm the PubMed supplied article is the same as the article being queried.

After the search returns a positive match, the PubMed abstract page for that article is parsed for any hyperlinks that may lead to the actual online article [8]. These links are tested to see if they do in fact return data. If the links are not broken, then they are entered into the database. If there are no links to the online article found in the PubMed abstract page, then the PubMed abstracts URL for that particular article is used in place of the article URL.

Overall, ISI Web of Science supplied 7,310 articles. These 7,310 articles were then parsed and tested to see if a URL was supplied by PubMed. This narrowed the field to 2,326 usable articles.

5. DEVELOPING AN ARTICLE SCORING MODEL

The scoring model for an article is based largely on semantic term relationships with additional consideration to citation information. All semantic scores of the term relationships are constructed through user input, while the citation information is predetermined.

There are 3 main techniques and ideas that are employed in the semantic scoring of an article, which are TFIDF (5.1), Ontologies (5.2), and the Scoring Matrix (5.3). Each of these concepts are used as a major aspect of determining an article's relevance to the user's search criteria, whether it be selecting terms through TFIDF or actually scoring the article using an ontology with a scoring matrix. Each one of the concepts are expanded on below and then brought together to illustrate the article scoring function (5.4).

5.1 Term Frequency \times Inverse Document Frequency (TFIDF)

TFIDF is a way to rank words and articles based on how often a word appears in an article and how many articles contain that term. TFIDF is the product of the term-frequency of a word (TF) and the inverse-document-frequency (IDF) of that word [2].

TF states that a word is more important in describing information in a document if it occurs more often, with the exception of stop words (*e.g.*, *a*, *the*, *at* ...). [2] It is defined as the number of occurrences of the i th term in document d :

$$tf_{i,d} = \frac{|d_i|}{|\sum_i d_i|} \quad (1)$$

where $|d_i|$ is the count of term i in document d and $|\sum_i d_i|$ is the sum of all the terms i in document d .

IDF states that if a term appears in many documents, then it is less significant in describing information in a single document. [2] It is defined as the log of the total number of documents n over the total number of occurrences of term i in a document d from the total sets of documents D :

$$idf_{i,D} = \log_2\left(\frac{|D|}{|\{d_i | d_i \in D\}|}\right) \quad (2)$$

The product of TF and IDF gives a word weight, where the weight of the i^{th} word in the d^{th} document can be defined as follows:

$$weight(i, d) = \begin{cases} (1 + tf_{i,d})idf_{i,D} & \text{if } tf_{i,d} \geq 1, \\ 0 & \text{if } tf_{i,d} = 0 \end{cases} \quad (3)$$

The TFIDF calculations are done through a previously built system called LUCAS [5]. LUCAS was developed at Indiana University's School of Library and Information Science by Yueyu Fu and Javed Mostafa. The system has a term representation database that was compiled at UC Berkeley and Stanford, which contains document frequencies and ranks of 31,928,892 terms found on 49,602,191 pages on the internet (<http://elib.cs.berkeley.edu/docfreq/>).

BioKnOT supplies LUCAS with a text file of the article abstracts and a list of the top 50 ranked words are returned and then given to the user for selection.

5.2 Ontologies

"Ontology" is a term borrowed from artificial intelligence, via philosophy, that is intended to describe a particular domain of objects and their relationships [6]. For example, in the domain of sugars $\{Ribose, \alpha\text{-D-glucopyranose}, D\text{-glucose}\}$, we have relationships *Ribose is an aldopentose*, *$\alpha\text{-D-glucopyranose}$ mutarotates to $D\text{-glucose}$* , and *$D\text{-glucose}$ mutarotates to $\alpha\text{-D-glucopyranose}$* . Ontologies can and usually include instances, and are used in many areas including database, machine learning, and virtually all areas of the life sciences.

By placing constraints on the kind of relationships, one can create, say, *taxonomies* which require the relationships to collectively form a rooted tree which is the familiar Linnaean system. The National Library of Medicine Medical Subject Headings (<http://www.nlm.nih.gov/mesh/meshhome.html>) is another.

BIOKnOT is motivated by the successful, ubiquitous use of ontologies. We choose, however, to create a local, ontology for each document and compare them pairwise with an ontology from the user to provide a means of search. We establish relationships by proximity and gather the objects from the user selected terms from TFIDF calculations.

5.3 Scoring Matrix

Scoring matrices are commonly used to show how strong or weak a relationships is between a pair of something. In this case, we want to see how strongly related the words are that the user selects as being important.

An $n \times n$ matrix is constructed, where n is the total number of terms the user selects from T_f , which is the set of words taken from LUCAS's ranked words. This set of user selected terms will be called T_u . For each term intersection in the matrix a log-odds score is calculated. The log-odds score offers us a probabilistic way of determining how significant a term relationship is in relation to finding each word separately in the random model. The random model is defined as the set of all the abstracts of each article that passed the initial Boolean search and is noted as S_r . The log-odds score is calculated as follows:

$$LOD = \log_2\left(\frac{P(t_2|t_1, d)}{P(t_1) \times P(t_2)}\right) \quad (4)$$

where $t_1 \in (S_r \cap T_u)$ and $t_2 \in (S_r \cap T_u)$.

The individual term probabilities ($P(t_1)$ and $P(t_2)$) are calculated by simply taking that term's count within S_r and dividing that by the total number of words in S_r . So, each term's probabilities is calculated as follows:

$$Pt_i = \frac{|t_i|}{|\sum_i d_i|} \quad (5)$$

where $|t_i|$ is the total count of term t_i in S_r and $t_i \in (S_r \cap T_u)$ and $|\sum_i d_i|$ is the sum of all the words in S_r , including t_i .

The term relationship probabilities are calculated using the distance between terms. If t_1 is found, then a reading frame of 20 words is opened following the position of t_1 . If t_2 is found within that reading frame, then it is counted. Only the first occurrence of t_2 within the reading frame is noted and circular relationships are not considered, so an occurrence of t_1 within the reading frame is ignored. This relationship count is then divided by the total count of all term relationships within S_r . So, each term relationship in T_u that is present in S_r is calculated as follows:

$$P(t_2|t_1, \delta) = \frac{|\{t_2|t_1, \delta\}|}{|\{t_j|t_i, \delta \wedge i \neq j\}|} \quad (6)$$

where δ is the size of the reading frame after t_1 , t_1 is the first term, and t_2 is the second term.

A further refinement of the scoring matrix is then done with user input. The user was asked to score all of the term relationships that constructed the scoring matrix based on how important the relationships are to the search. This way the user can give another level of specification to the relationships that are used to score each article. The user's score is transferred to a range of $[0, 2]$, where 0 represents no

significance to the search and 2 represents the highest level of significance to the search. This new range score $r_{i,j}$ of term i and term j is a multiplier to the values that are in the scoring matrix. Negative and positive relationships have to be considered, so the final scores in the scoring matrix are calculated as follows:

$$FinalScore_{i,j} = \begin{cases} \frac{1}{r_{i,j}} S_{i,j} & \text{if } S_{i,j} < 0 \\ 0 & \text{if } S_{i,j} = 0 \\ r_{i,j} S_{i,j} & \text{if } S_{i,j} > 0 \end{cases} \quad (7)$$

where $S_{i,j}$ is the Scoring matrix value for i and j .

5.3.1 Example of LOD score

If we have a term relationship of $cell \rightarrow death$, where $cell \in (T_u \cap S_r)$ and $death \in (T_u \cap S_r)$. $Cell$ has a probability of 0.004 of appearing in S_r , $death$ has a probability of 0.009 of appearing in S_r , and $cell \rightarrow death$ has a probability of 0.003 of appearing in S_r , the log-odds equation (equation 4) of:

$$LOD = \frac{0.003}{0.004 \times 0.009}$$

would give a score of 1.92. If the user scored this relationship as extremely important to their search, then the range score multiplier value would be 2, taken from the range of [0, 2]. The LOD score would then be multiplied by the range score and we would get a score of $1.92 \times 2 = 3.84$. This value would then be entering in the matrix and the calculations are iteratively done for all term intersections in the matrix.

5.4 Putting it All Together

The semantic score for the article comes from comparing an ontology created from the user's statement to an ontology created from an article's abstract, where all of the terms in these ontologies were determined by TFIDF calculations and selected by the user. The first ontology is created from the user's input statement, which will be referred to as M , where $M = (V, E)$. In the case of the "quick search", the ontology is simply determined to be a perfect relationship of all the user selected terms from the TFIDF filter or set T_u . The second ontology is created from the article's abstract, which will be referred to as N , where $N = (V, E)$. V for both ontologies are the vertices, which is a set of words. E for both graphs is the set of directed, weighted edges, where the weight is represented by the inter-sentence and intra-sentence pair of values.

There are two values that measure distance between two terms. The first is the inter-sentence distance, which finds term relationships through out a paragraph, while ignoring sentence structure. The second is intra-sentence distance, which finds the term relationships within sentences of a paragraph, where sentence structure does matters.

If we want to find the relationship between two terms, t_1 and t_2 , the positions of all t_1 and t_2 are noted. The distances are found by subtracting the position of t_2 from the position of t_1 where there is an occurrence of t_1 before t_2 within a specified reading frame. All of these distances are then averaged to find the average distance between the terms. The intra-sentence distances only note these positions of t_1 and t_2 within sentences, which are delimited by a period.

The inter-sentence distances ignore the period and will take into consideration all term positions within a paragraph. These distances are what comprise the term relationships within the ontologies.

In order to compare the two ontologies, an adjacency matrix is derived from ontologies M and N , which captures the directed term-to-term inter-sentence and intra-sentence distances. (figure 6)

These two ontologies, M and N , which are now comparable through their matrix form are scored through the following equation, where i is the first term and j is the second term in the term relationship.

$$Score_{i,j} = \sum_{i \neq j} P_{M,N}(i,j) \times S_{i,j} \times f_{M_{i,j}}(N_{i,j}) \quad (8)$$

where $S_{i,j}$ is the score of term i and term j from the scoring matrix (section 5.3).

$$P_{M,N}(i,j) = \begin{cases} 1 & \text{if } M_{i,j} \times N_{i,j} \neq 0, \\ -1 & \text{otherwise} \end{cases} \quad (9)$$

$$f_{M_{i,j}}(N_{i,j}) = \begin{cases} 1 & \text{if } x \leq \alpha \\ 1 - \frac{1}{2} \left(\frac{x-\alpha}{\beta-\alpha} \right)^2 & \text{if } \alpha < x \leq \beta \\ \frac{1}{2} \left(\frac{\beta-x}{\gamma-x} \right)^2 & \text{if } \beta < x \leq \gamma \\ 0 & \text{if } x > \gamma \end{cases} \quad (10)$$

where α is the position of term i , β is $\alpha + 5$, γ is a distance of $\alpha + 20$, and x is the position of term j .

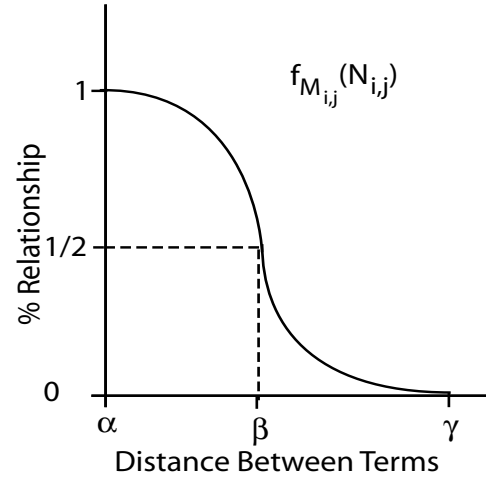


Figure 3: The abscissa represents the spacing of words, where α is the position of $M_{i,j}$, β is the equal to $\alpha + 5$, and γ is equal to 20. The ordinate represents the strength of the term relationship.

The score is based on the summation of all the term relationship products of the scoring matrix value for terms i, j , the presence of i, j in $N_{i,j}$, and the sigmoidal function of $N_{i,j}$ based on $M_{i,j}$ (figure 3). The sigmoidal function is based on intuition of term relationships. If two terms are very close together, then they are assumed to be highly correlated, but

as the distance between words increases, the correlation dramatically decreases until the distances reaches a point where there is no relationship assumed.

The user is shown two scores for each article. The first is the semantic score that is discussed above and is calculated by equation 8 and the second is the citation score. The citation score is simply the citation rate per year, which is calculated by taking the citation count and dividing it by the difference of the publication year and the present year.

The citation value offers the user a type of support to the semantic score, since citations offers us another measure of the importance of an article. These two scores are offered to the user separately, because they are independent of each other. The differences are based on how important a user finds how often an article is cited versus how relevant an article is to one's search criteria.

6. CONCLUSION AND FUTURE WORK

We have designed and implemented an information retrieval system for biological data that builds upon current well-established IR technologies, but that allows for the ability of the user to refine search iteratively using ontologies to deal with problems of "drilling-down" and timeliness.

Future work includes having much more sophisticated support by taking into account trends and how often papers are cited by other papers. Another necessary task is to do evaluation and accuracy testing to verify, indeed, whether BioKnOT is useful.

An evaluation test can be devised where we compare the results of BioKnOT to another system by simply taking the top x amount of documents from each system with the same search criteria, and determine whose system returned the most relevant documents. To get accurate results from this experiment, however, both systems would be required to have the same dataset. This would be a problem as every system has its own dataset or at least some variation of a dataset. The other problem would lie in the measure of relevance. There are many ways to measure relevance, but which one would give the most fair results based on the different search techniques that each system used. Several relevance measuring techniques need to be explored before a fair evaluation of BioKnOT can be determined.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Javed Mostafa and Dr. Sun Kim for many helpful suggestions.

8. REFERENCES

- [1] M. Ashburner, C. Ball, J. Blacke, and et al. Interoperability of biodiversity databases: biodiversity information on every desk. *Nature Genetics*, 25(1):25–29, 2000.
- [2] M. W. Berry. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer-Verlag New York, inc., New York, NY, 2004.
- [3] M. M. Dalkilic and A. Sengupta. Design and evaluation of catpa: Curation and alignment tool for protein analysis. Technical Report TR109, School of Informatics, Indiana University, 2003.
- [4] J. Edwards, M. Lane, and E. Nielssen. Interoperability of biodiversity databases: biodiversity information on every desk. *Science*, 289(5488):2312–2314, 2000.

- [5] Y. Fu and J. Mostafa. Toward information retrieval web services for digital libraries. *JCDL*, June 2004.
- [6] T. Gruber. Translation approach to portability ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [7] I. Korf, M. Yandell, and J. Bedell. *Blast*. O'Reilly & Associates, 2003.
- [8] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.
- [9] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Springs Harbor Press, 2001.
- [10] D. Oliver, D. Rubin, J. Stewart, and et al. Ontology development for a pharmacogenetics knowledgebase. *Pacific Symposium on Biocomputing*, pages 65–76, 2002.
- [11] H. Pearson. Biology's name game. *Nature*, 411(6838):631–632, 2001.

9. APPENDIX

BioKnOT may currently be accessed at the following URL:
<http://biokdd.informatics.indiana.edu/cgi-bin/jccostel/thesis/bioknot.cgi>

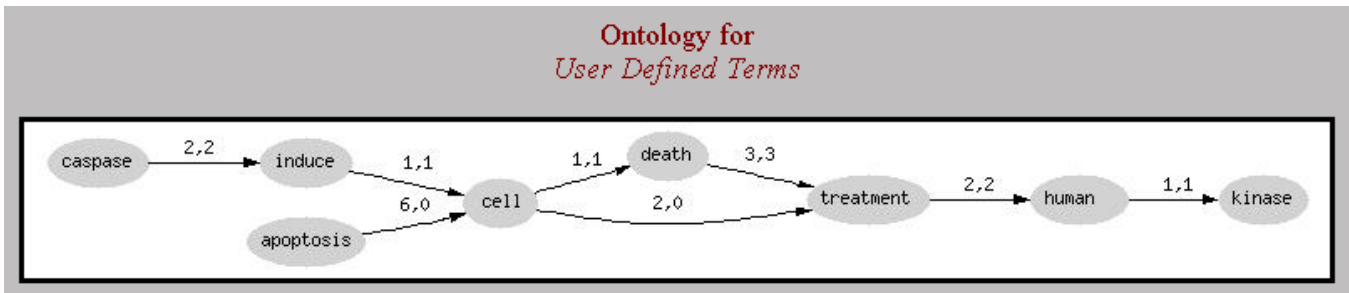


Figure 4: Screen shot of an ontology created from user defined input. The intra-sentence value is the first in the pair, while the inter-sentence value is the second in the pair.

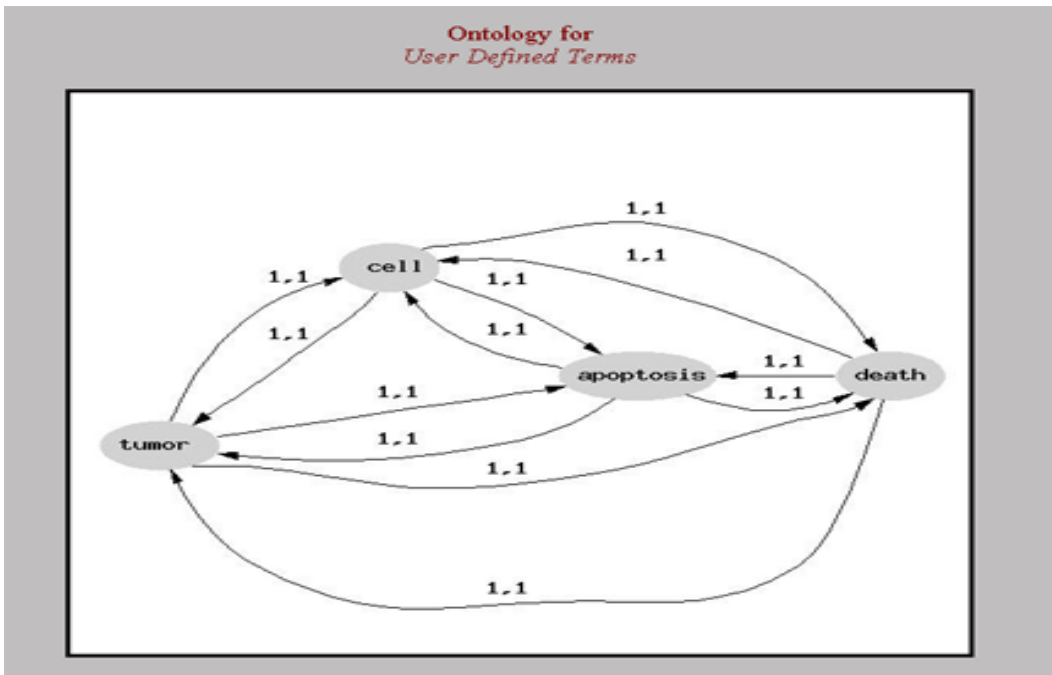


Figure 5: Screen shot of an ontology that was created for the "quick search" option. All of the relationships for each term are considered and all with an intra-sentence and inter-sentence value of 1.

Adjacency Matrix for Intrasentence words

	activation	apoptosis	inhibitor	caspase	cell	induce	induction	protein	treatment	death
activation	-1.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
apoptosis	0.00	-1.00	0.00	0.00	4.50	0.00	6.00	0.00	8.00	0.00
inhibitor	0.00	0.00	-1.00	10.00	0.00	0.00	0.00	0.00	0.00	0.00
caspase	1.00	0.00	1.00	-1.00	0.00	0.00	0.00	0.00	0.00	0.00
cell	0.00	0.00	0.00	0.00	-1.00	10.00	0.00	0.00	0.00	1.00
induce	0.00	1.00	0.00	0.00	3.00	-1.00	0.00	0.00	0.00	0.00
induction	5.00	4.00	0.00	0.00	0.00	0.00	-1.00	17.00	0.00	0.00
protein	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00	0.00	0.00
treatment	0.00	0.00	0.00	15.00	0.00	0.00	0.00	0.00	-1.00	0.00
death	0.00	0.00	8.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00

Figure 6: Screen shot of an adjacency matrix that is used to compare the user defined and the article ontologies. Intra-sentence and Inter-sentence matrices are the same structure, but with difference values for the term relationships.