2009

Clustering Algorithm for Enhanced Bibliography Visualization

Sriram Krishnan
San Jose State University

Clustering Algorithm for Enhanced Bibliography Visualization

A Writing Project
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial fulfillment
Of the Requirements for the Degree
Master of Science

By
Sriram Krishnan
May 2009

SAN JOSÉ STATE UNIVERSITY

The Undersigned Writing Project Committee Approves the Writing Project Titled

Clustering Algorithm for Enhanced Bibliography Visualization

by

Sriram Krishnan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Soon Tee Teoh, Department of Computer Science	05/01/2009
Dr. Chris Pollett, Department of Computer Science	05/01/2009
Dr. Mark Stamp, Department of Computer Science	05/01/2009

ACKNOWLEDGEMENT

I am taking this opportunity to thank my Project Advisor Dr. Soon Tee Teoh for his wonderful Idea and guiding me for developing this unique project. I would also like to thank my Committee members – Dr. Mark Stamp and Dr. Chris Pollett for their invaluable guidance on this project.

Table of Contents

List of Figures

List of Tables

Abstract

A Bibliography is a list of books, publications, journals etc., with details such as authors and references. Visualization could be used as a data analysis tool to represent various types of data, analyze huge chunks of data easily and arrive at interesting results. The idea of this project is to provide a medium which eases the combination of bibliography with visualization. Though there are many sources of bibliographic data like the Digital Bibliography and Library Project (DBLP), Citeseer, Google Scholar, none of these data could be used directly for deducing relations between various entities or for visualizing the relationship between related entities.

This project aims at providing a web-service that takes user queries as input and retrieves the corresponding data from a local database. Then the web-service applies a clustering algorithm to the retrieved data and then presents the clustered data as XML to the requestor. The user of the system could be any automated program that aims at providing a visual interface to bibliography. One of the main outcomes of this approach would be bringing out the hidden relationship between various related bibliographic entities and making the relationships more obvious and readable than the existing systems.

Introduction

Clustering is the science of classifying objects into groups [1]. Clustering of objects is done to make the analysis of huge volumes of data easier. The outcome of this project is a simple and easy to use web-service that provides a simple XML interface for Bibliographic data analysis and visualization. This tool caters to many basic needs of a visualization tool that could present bibliographic data.

Our web-service does author centric clustering, paper-centric clustering and also provides a clustering algorithm based on Self organizing maps algorithm [12] to place related papers on a map. This web service also provides a real time keyword-based paper search capability powered by the Yahoo search API. In addition to these a paper ranking algorithm based on the page rank algorithm [8] has been implemented to rank all the papers in the database based upon their level of importance.

The author centric clustering takes the author name as the query and gives the overall weight of the author, the papers he has authored, the co-authors with whom the particular author has worked with etc.

The paper-centric view tries to do a similar clustering with paper search, by providing details about the searched paper such as author names, paper weight, references, papers referring this paper etc.

The third clustering algorithm retrieves related papers of a particular paper (both references and referrers) and calculates the possible distances between every possible

paper pair and gives the X, Y co-ordinates for all the papers to be plotted on a map. This also provides color codes to papers based upon their importance.

By using this web-service as a backend engine developers can develop interesting online applications that involve bibliographic data analysis and visualization.

Bibliographic Data Sources

In order to do the data analysis the first step is to identify the source of data. There are lots of sources for the bibliographic data like DBLP, Citeseer [18], Google Scholar, IEEE portal, ACM Portal. After analyzing all these data sources, I was decided maintain focus on DBLP and Citeseer. The other data sources did not provide a metadata format for their bibliographic database.

DBLP stands for Digital Bibliography and Library Project. The DBLP website is hosted and maintained by the Universität Trier, in Germany. It contains a listing of more than one million articles in computer science. The databases tracked by DBLP include VLDB, IEEE, ACM and some technical conferences. Complete Search DBLP and Faceted DBLP are some known variants of DBLP. DBLP provides an XML file which contains the entire DBLP database, for programmers who work on bibliographic data.

Citeseer is an autonomous indexing system which serves as a public search engine for digital and academic publications. The original goal of Citeseer was to crawl and index publication autonomously. This website was developed and is being maintained by researchers at Pennsylvania State University. Citeseer has over 700,000 documents in its

database. Citeseer also indexes DBLP and ACM portals. Due to some technical difficulties the database was not updated after 2005.CiteseerX is the new variant of Citeseer that is available on the World Wide Web. CiteseerX has a undergone a lot of changes in early 2009 and it has various interesting features like personal bookmarking, social networking etc. Citeseer provides an archive file that consists of 72 XML dump files, for programmers who work on the bibliographic data. Citeseer also has an online interface where developers can query bibliographic data. As this interface has consistency issues and is not very simple we choose to use the archive file and replicate the environment in the development machine. The table below serves as a comparison chart between DBLP and Citeseer.

Table 1: DBLP Vs Citeseer

	DBLP	Citeseer
Co-Authors	Yes	Yes
References	Yes	Yes
Referred-By	No	Yes
Key Words	No	No
Metadata in XML	Yes	Yes
Paper Identifiers	No	Yes

Table1:

Though Citeseer had data only till 2006 it does provide more information than DBLP, because the Citeseer indexes DBLP itself. One important factor that DBLP has failed to address is the References section; this is available in the Citeseer XML dump. Citeseer also takes care of certain important issues like identifying same citation in different format. All these features made Citeseer a more appropriate data provider for this project.

Clustering Techniques

Clustering could be defined as grouping objects with similar properties together. Since objects can have various properties, it is quite possible to group the same set of objects in various ways. Clustering is one of the basic method used to ease the process of data mining, some times it could also be used to bring out the results of a data mining experiment. In this project we use a custom defined algorithm for certain clustering techniques, this algorithm is more similar to the hierarchical clustering [1]. One of the most trivial features of any clustering algorithm is to determine the distance metric. The distance metric is the one that determines how similar two entities are and how closer they need to be represented. We take into consideration the Euclidean distance as our distance measure and we use a custom algorithm to determine the similarity or closeness between any two entities. This web service groups the data in three different ways and each way has its own algorithm and distance metric. The following sections discuss in detail about each individual method.

Author Centric Clustering

This is the first type of clustering that is provided by our web service. In order to retrieve the cluster based upon an author, the system expects an ‘author id’ as its input. In order to get this the user has to search for the author and locate the author. Then he has to pass the author id of the particular author whose details he wants to view.

After taking in the author id as input the method retrieves the author centric view details for that author from the database. Note that all the computations on the database side take place dynamically; there are no pre-computed results that are being stored and retrieved. The results contain the author name, id and weight. Weight of an author indicates how important that particular author is .This will be decided by an algorithm that checks how important this author's papers are. We will discuss in brief about how the weight of an author is calculated in the following section.

The next part of the results contains the list of Co-authors along with their name, id, weight and bond. Bond stands for the distance between the queried author and a particular co-author, i.e., bond between two authors mean how close they are. This is one of the most important parameter that could represent the cluster. The next part of the result contains the list of publications done by this author, each publication's paper id, title, description, authors and weight. Weight of a paper stands for the number of papers/articles inside this database that have cited this paper as a reference. The final part of the result contains the author bond matrix. This matrix contains the details of the bond between each and every author (including the queried author) specified in the co-author list. This enables a visualization tool to place the authors at appropriate distance from each other.

The following headings throw light on how individual components of the author centric clustering are computed

Author Weight

The weight or importance of each author is calculated as follows

Weight Wa= (Σ $R(Pa)/\Sigma$ Pa)

Where Wa is the weight of the Author ‘a’

R(Pa) is the Rank of the Paper P written by author ‘a’

In this formula we calculate the weight of the author by calculating the average weights of papers written by him. So when there are a group of authors and their weights fetched out from a query then a percentile system of ranking may be locally implemented at the client side base upon the global ranks of each author.

Author Bond

Author bond is the distance metric of the Author centric cluster. Author bond is the parameter that decides how close or nearby a pair of authors are. This is calculated using an algorithm that depends upon various factors.

Let the two authors whose bond to be calculated be Author ‘a’ and Author ‘b’

Common papers

C(a,b)= COUNT(P(a) ∩ P(b))

P(x) is the list of papers authored/co-authored by author x.

Mutual Citations

MC(a,b)= COUNT(((Ref(a) ∩ P(b)))+ ((Ref(b) ∩ P(a))))

Mutual citations of 2 authors are the sum of number of papers of author ‘a’ cited by author ‘b’ and vice versa.

Both C(a,b) and MC(a,b) are normalized by using the formula

C(a,b)= C(a,b)/ /COUNT(P(a)+P(b)-C(a,b))

MC(a,b)= MC(a,b)/ /COUNT(P(a)+P(b)-C(a,b))

The final author bond is calculated by taking average of these 2 values and then converting it to a percentage value by multiplying it with 100.

Paper-centric Clustering

This is the second clustering algorithm that is offered by our web-service .In order to retrieve the cluster based upon a paper; the system expects a 'paper id' as its input. In order to get this the user has to search for the paper and locate the paper and its id. Then he has to pass the author id of the particular author whose details he wants to view. The search for a paper is not as straight forward as a search for an author. Because we cannot expect a user to search a paper with some words that are present in the title, the system also needs to display certain papers that are relevant to the search term. For this purpose the database computes the keywords of all the papers based upon their description and stores it. When the user searches for some paper, we check both the title and keywords and return the results. As a result of the previous search there would be more number of papers returned; this means more strain on the user to search for the paper.

In order to bring in important papers upfront a paper rank algorithm was implemented and this ranks all the papers in the data base based upon its importance and the importance of the papers that site this paper. This algorithm is based upon the famous page rank algorithm. Finally a ranked paper set would be returned as a result of the user query.

After retrieving the paper id, the id is supplied as input to the paper centric view method available in the web service; the procedure sends this paper id to the database and retrieves the paper-centric view details from the database. Once again all computations are dynamic and no result is pre-computed.

The first part of the result contains the paper id, title, description and weight. Weight stands for rank of the paper that is generated by the page rank algorithm. This serves as the overall ranking of the paper (used while comparing 2 or more papers). The details of the page rank algorithm will be discussed in the framework section of this report

The next section of the result contains the list of author ids, names, weights of authors who have authored the particular paper.

The next section of the result is the references section. This section contains the list of references cited by this paper. This includes paper id, title description and the author ids for each reference.

The Final section is the referred by section, this section is similar to the references section except for the fact that the papers in this section refer to the papers that cite this paper (the paper that is being queried) as its reference.

Self Organizing Map Based Paper Clustering

This is the third type of clustering provided by our webservice .Self organizing maps are a type of artificial neural network which can train themselves without any supervision and produce a low dimensional representation of the input sample. Since we are representing our data in 2 dimensions SOM would be a perfect choice for clustering our

data. SOM is more like multidimensional scaling and is often referred to as Kohonen maps (named after the person who found this method). [12]

Let us see an example of a 2D- scaling done implemented using SOM.

ITEM	Protein	Carb	Fat
Apples	0.4	11.8	0.1
Avocado	1.9	1.9	19.5
Bananas	1.2	23.2	0.3
Beef Steak	20.9	0	7.9
Big Mac	13	19	11
Brazil Nuts	15.5	2.9	68.3
Bread	10.5	37	3.2
Butter	1	0	81
Cheese	25	0.1	34.4
Cheesecak	6.4	28.2	22.7
Cookies	5.7	58.7	29.3
Cornflakes	7	84	0.9
Eggs	12.5	0	10.8
Fried Chic	17	7	20
Fries	3	36	13
Hot Choco	3.8	19.4	10.2
Pepperoni	20.9	5.1	38.3
Pizza	12.5	30	11
Pork Pie	10.1	27.3	24.2
Potatoes	1.7	16.1	0.3
Rice	6.9	74	2.8
Roast Chic	26.1	0.3	5.8
Sugar	0	95.1	0
Tuna Steak	25.6	0	0.5
Water	0	0	0

Figure 1: SOM Example – Multi dimensional scaling

In the figure above lists items like apple rice sugar etc and their corresponding protein, carbohydrate and fat levels. We can see the clustered output on the left hand side of the image. We can see that items with high protein are nearer to Tuna steak, water that does not have any of these ingredients is pushed towards the bottom right away from others. All fat rich items are placed near butter. The plot graph and the list represent the same data, but we are able to make more meaning easily out of the plot graph than the chart. So

we are going to apply the same strategy for clustering a specific group of papers. As an extension of the paper-centric view we for a group of papers by considering a set of papers that are either references or cite of a particular paper being queried. Now that we have a set of papers we can proceed to cluster them using SOM.

As a first step towards this approach we need to define the distance metric between any 2 papers. In the context of grouping we define distance between any 2 papers by using various criteria.

1) Common authors
2) Common keywords
3) Common References
4) Number of Papers citing both these papers as references
5) Either paper1 refers paper2 or vice versa.

All of the 5 parameters contribute significantly to the paper bond that is calculated between two papers. The following section explains in detail, how each parameter affect the bond value and see how they are computed.

Common Authors

Common authors of 2 papers is given by

CA(p1,p2)=A(p1) ∩ A(p2) (where A(x) is the list of authors of paper x)

Common Author Bond Co-Efficient of a paper-paper bond is given by

CAB=(COUNT(CA(p1,p2))/COUNT(A(p1)+A(p2)-CA(p1,p2)))*100

In order to normalize the value of the common authors we divide them by the total number of authors of both the papers. And then we derive a percentage value by multiplying with 100.

Common Keywords

Common keywords of 2 papers is given by

CK(p1,p2)=K(p1) ∩ K(p2) (where K(x) is the list of keywords of paper x)

Common Keyword Bond Co-Efficient of a paper-paper bond is given by

CKB=(COUNT(CK(p1,p2))/COUNT(K(p1)+K(p2)-CK(p1,p2)))*100

In order to normalize the value of the common keywords we divide them by the total number of keywords of both the papers. And then we derive a percentage value by multiplying with 100.

Common References

Common References of 2 papers is given by

CR(p1,p2)=R(p1) ∩ R(p2) (where R(x) is the list of references of paper x)

Common References Bond Co-Efficient of a paper-paper bond is given by

CRB=(COUNT(CR(p1,p2))/COUNT(R(p1)+R(p2)-CR(p1,p2)))*100

In order to normalize the value of the common References we divide them by the total number of references of both the papers. And then we derive a percentage value by multiplying with 100.

Common Papers citing both the papers

Common papers citing the 2 papers is given by

CC(p1,p2)=C(p1) ∩ C(p2) (where C(x) is the list of papers citing of paper x as its reference)

Common Citation Bond Co-Efficient of a paper-paper bond is given by

CCB=(COUNT(CC(p1,p2))/COUNT(C(p1)+C(p2)-CC(p1,p2)))*100

In order to normalize the value of CC we divide it by the total number of papers that cite paper1 plus total number of papers citing paper2. And then we derive a percentage value by multiplying with 100.

Mutual Reference

Mutual Reference Bond Co-Efficient of a paper-paper bond is set to 100 when either paper1 refers paper2 or vice versa.

Paper-Paper Bond=(CAB(p1,p2)+CKB(p1,p2) +CRB(p1,p2) +CCB(p1,p2)+MR)/5

Now there is a list of a group of a paper related to a particular paper say paper1 and a known algorithm to calculate the distance between two papers. So bonds are calculated between all possible paper pairs in this group and then we have a tabular data of distances similar to the table in the SOM example figure. One of the key things to note here is that we substitute the protein, carb and fat on the Example SOM table with distances to papers with distances to the respective papers as shown below. The out put will be X,Y Co-ordinates on a 2 dimensional plane. As a multi dimensional data is scaled down to two

dimensions this technique is also known as multi dimensional scaling. The step by step approach is given in the pictures below.

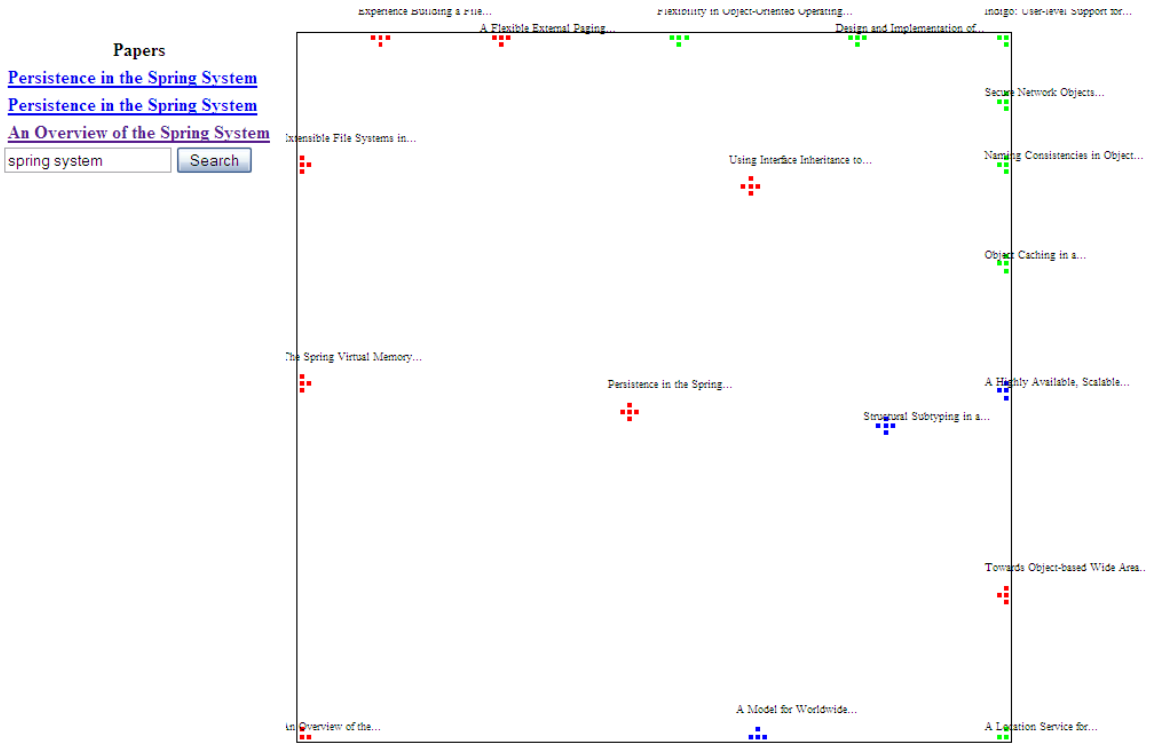

Figure 2: Sample SOM Output

In addition to placing the papers on a map with distances related to the bond strength, we also color code each paper according to its weight. So the out put now becomes useful for much complex and in depth visualization. Using such visualization one can not only find the most relevant papers to the paper you search, but also find the most important papers in that area.

Project Framework

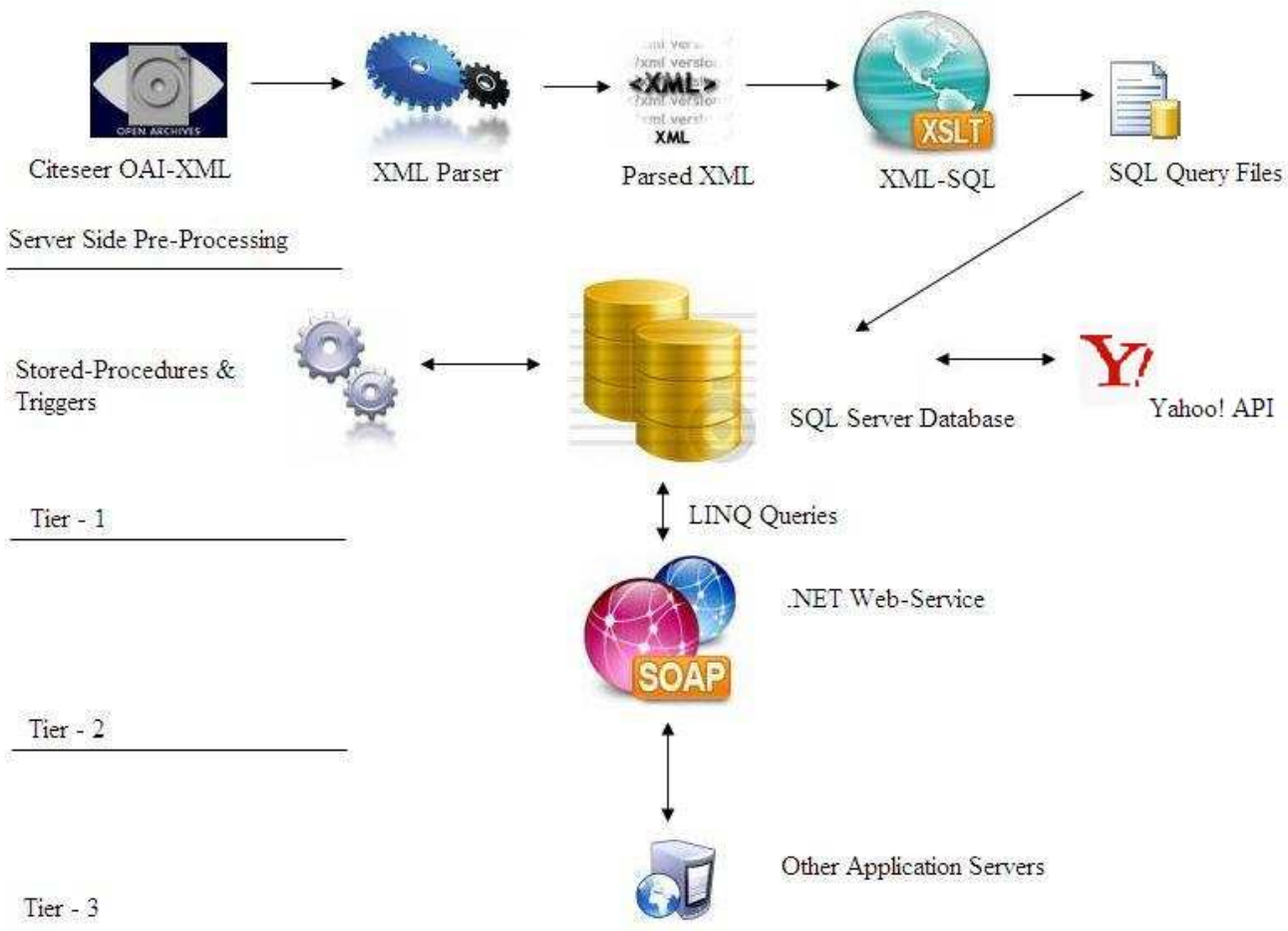

Figure 3: Project Framework

This figure clearly displays the overall view of the project. As illustrated the setup needs some server side pre-processing before the actual development can start. This pre-processing is done in order to import the data provided by the Citeseer XML and then dump it inside our server's database on which the clustering is to be done. Triggers, Stored procedures and User defined Functions (UDF) that are written on the database tables facilitate easy import of data. The stored procedures and functions also perform the

main processing for the clustering. An important point to note here is that the clustering is done on an ad-hoc manner rather than pre-computing the entire clustering result beforehand. This makes the application to behave dynamically and helps in avoiding redundant clustering after every data load .At tier-2 a web-service is created to access the data inside the database and expose it over the internet. The web-service has been programmed using Microsoft.NET technology. The web-service exposes certain methods such as SearchAuthors, SerachPapers, AuthorCentricView and PaperCentricView. All users at Tier -3 can consume this web-service and manipulate the XML data it provides in response to the queries posted. Tier – 3 users may consist of third party Application Servers, Database Servers or Web Servers. The communication between tier - 3 and tier – 2, takes place over HTTP (or HTTPS). Tier - 2 communicates with Tier - 1 that is the database layer by sending SQL queries. The communication between these 2 entities is facilitated by Microsoft.NET's LINQ (Language Integrated Query). The programmer can specify the QUERY in LINQ and this gets translated into an equivalent and efficient SQL query and sent to the database. The results are acquired via LINQ interface as well.

Server Side Pre-Processing

This section deals with all the processing that was done to import the Citeseer data inside the local database.

Citeseer OAI-XML

OAI stands for Open Archives Initiative. OAI compliance indicates that it is compatible with Open Archives Initiative's Protocol for Metadata Harvesting, often known as OAI-

PMH. This protocol allows metadata harvesters to locate and gather data from different data sources, thereby providing a diverse collection of datasets and the flexibility that they need not be stored locally. The data dump provided by Citeseer is compliant with OAI standards. There are 72 such dump files available from Citeseer. A sample of a single Citeseer record is provided in the Appendix.

XML Parser

Though the fact that the data provided by Citeseer is OAI complaint sounds good but this is rather an unnecessary overhead in this project because, the captured data is going to be stored and manipulated at a local server using a different format. The namespaces used by OAI are also quite a hindrance while performing XSL transforms. Thus the namespaces and namespace references of `oai_citeseer` and `dc` are stripped off each record. The other things that are stripped off are `copper.ist.psu.edu` URL references and its corresponding namespace and certain invalid characters that occur occasionally in the data. Also the XML has no root node so each file is appended with a root node for that file .In order to speedup the find and replace process the Parsing is faked. That is the XML is not actually parsed as an XML, instead the entire file is read as a whole and then simple string replacement is done. Sample XML after parsing might look like the parsed XML shown in the Appendix.

XML to SQL

Now that we have a proper XML that is easier to parse node by node and validate, an effective method should be found out to Move the contents inside the SQL SERVER

database. The easiest method of data upload is bulk insert queries. We use an XML Language Transformation (XSLT) and some small .NET script to transform the XML data into .SQL files containing the appropriate insert queries. By using XSLT instead of using some high level language code to parse the XML we gain a lot of performance, that is a very meager the time taken to generate the SQL files. The XSLT code that is used to transform the XML to SQL and a sample transformed SQL query are given in the Appendix.

Tier – 1

This layer is the database layer. This section discusses in brief how the database is structured and explains all the available tables, functions, procedures and triggers.

As the web-service was to be implemented in .NET, and the fact that .NET gels well along with Microsoft products MS SQL SERVER 2008 was chosen as the database. The figure below gives the structure of the database tables. The [master] table is the landing table. That is all the data from the SQL file are inserted to this table. Each insert statement corresponds to one row being added to [master]. To map the data from [master] to [papers],[authors] and [mappings] a stored procedure “sp_push_from_master_to_all” was written. Then a new database Trigger “update_all_from_master” is inserted on [master]. This trigger is fired when ever new data is added to [master]. This trigger maps all the fields of the new row(s) added to the parameters of

"sp_push_from_master_to_all" so that the other three tables are populated automatically whenever [master] gets a new row.

SQL SERVER Database

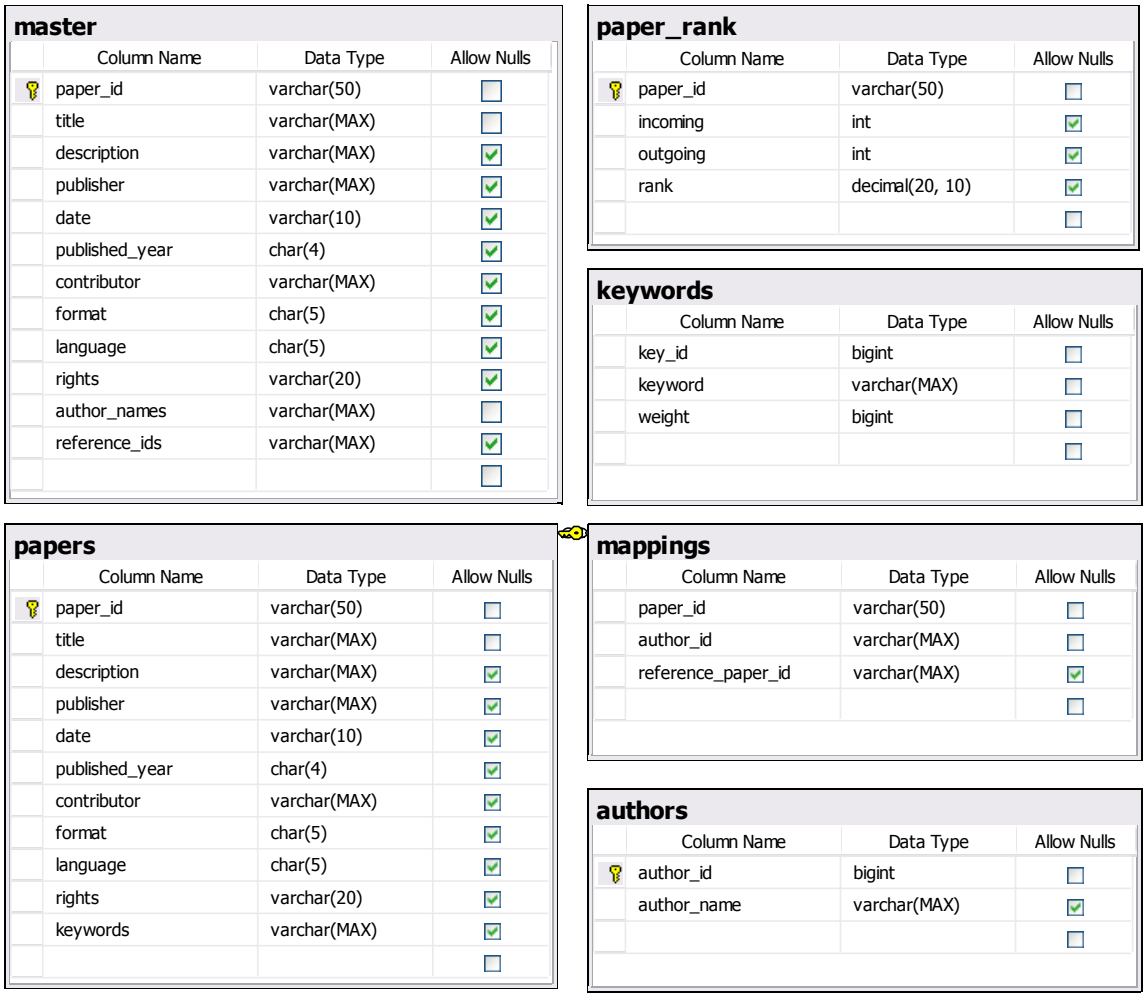

Figure 4: Database Table Layout

During this time a new row is added to [papers] and then a the author names are validated against the existing [authors].[author_name] in [authors] and then a new author is inserted inside the database if needed. Every new author is assigned a new id ([authors].[author_id]). Then the trigger proceeds to insert the data inside the [mappings]

table, which has a foreign key relation with [papers]. Here for every paper a single row is inserted with the corresponding [papers].[author_id] and [papers].[reference_paper_id]. The author id's inserted in [papers].[author_id] are separated by a colon(:). Similarly the reference paper id's inserted in [papers].[reference_paper_id] are separated by a colon(:). The fact to be noticed here is that the table [mappings] is not normalized. [mappings] has not been normalized taking into consideration the huge volume of data and the extent of data covered in each database transaction. There are about 750,000 records that needs to be inserted , on average if each paper is authored by 3 authors and each paper references 5 other papers then for each paper there will be 5 x 3= 15 rows in [mappings], if [mappings] was normalized. So for 750,000 records there will be 11250000 rows in [mappings], this is only a lower estimate. So each select query will try to lock 15x rows and all the database transactions would be very slow. Thus a non normalized version of the database is more suited for this application (only one row per paper).

Stored-Procedures & Triggers

There are eleven stored procedures that have been coded for this project

1. sp_push_from_master_to_all
2. sp_authorBonds → Distance between two authors
3. sp_paperCentric_Detail → Clustering result for paper query
4. sp_authorSearch → Search all authors matching a search string
5. sp_paperSearch → Search all papers matching a search string
6. sp_authorCentric_Detail → Clustering result for author query
7. sp_paper_paper_bond → Distance between two papers

8. sp_relatedPapersDetail → Retrieve related papers of a given papers and bonds between all possible paper pairs in the retrieved set of papers.
9. sp_insertKeywords → Insert new keywords in the keyword database and update count for existing keywords
10. sp_populate5000PapersWithKeywords → populates keywords in first 5000 paper entries in the database that do not have keywords processed
11. sp_rank_papers → Calculate the rank of all the papers based upon page rank [8] algorithm

There is one Trigger on the [master] table that was described in the pervious section

1. Update_all_from_master

There are twelve User Defined Functions (UDFs) supporting the stored procedures .

1. author_author_bond →Distance between two authors
2. fnAuthorsOfPaper → Returns a list of authors for a given paper
3. fnAuthorWeight → Returns the overall rank of the given author
4. fnCoauthors → Returns all Co-authors for a given author
5. fnGetKeywords → Calls Yahoo API and returns keywords for text input
6. fnKeywordsOfPaper → Returns keywords for a given paper
7. fnPapersOfAuthor → Returns all papers published by an author

8. fnPapersReferring → Returns all papers referring a given paper
9. fnPaperWeight →Returns the overall rank of a given paper
10. fnReferencesOfPaper → Returns all the references of a given paper
11. fnReferencesOfPapers→ Returns all the references of a given papers (: seperated)
12. fnSplit → Splits a string “a:b:c” and returns a table with rows a,b,c

As we have discussed the algorithm for most of the procedures and functions we will just concentrate on how key words are got and filled inside the papers table and we will also see the implementation of the paper rank algorithm.

Keyword Generation Using Yahoo Web Service

You many recall that we have used a keyword-based search to retrieve the list of papers as a part of the paper search query. This section discusses about how we pass the description of a paper to the Yahoo API and retrieve the corresponding keywords. These keywords can also be used as a parameter to weigh the bonds between two papers. The main challenge here is to generate the keywords from the description of a paper. The description of the paper is stored inside the papers table of the BIBLIO database. Upon some internet research I found that Yahoo! Offers a web service that takes a long text as input and gives out a list of keywords as the output. This is a REST [10] based web service offered by Yahoo! on its developer network [20]. This web service is called the yahoo contextual search API [14].

There are two methods the call to this web service could be done , The first method is to write a web proxy using any server side programming language and then call the web service. The main overhead in this method is that we need to query the database and pull out the description text for which we are generating the keywords and then call the web service, retrieve the results and then connect to the database and push the results inside the appropriate rows. Microsoft SQL server 2008 is much more flexible in this regard. Since the .NET environment has a common language runtime (CLR), it is possible to write a database stored procedure or a user defined function in C#.NET and then register it in SQL Server. In order to support the CLR code some configuration needs to done on the SQL server.

As a first step we need to set the TRUSTWORTHY property [16] on the SQL server, This could be done by using the following SQL statement

```
ALTER DATABASE biblio SET TRUSTWORTHY ON
```

This statement is used to make the SQL server trust that this database (biblio) and its contents can be trusted. Since we are about to add external CLR code this is a required step.

In order to test the Yahoo! web-service call, I wrote a test .NET program that calls the web-service by supplying a sample text as a parameter and then prints out the result on a

browser. I did an internet search on some existing code for calling a web service using a .NET code. I found out two useful articles [2], [14]. The article "Accessing HTTP Web Services" [14] gave a clear idea on how to call their web service. In order to use Yahoo!'s web services extensively, I registered for a Yahoo! application Id, that could be used to make Yahoo! web service calls.

The following snippet of code was used to call the web service and receive the response

```
protected void termExtractionProxy()
{
Uri address = new
Uri("http://api.search.yahoo.com/ContentAnalysisService/V1/termExtracti
on");
HttpWebRequest request = WebRequest.Create(address) as HttpWebRequest;
request.Method = "POST";
request.ContentType = "application/x-www-form-urlencoded";
string appId = "sample app id";
string context = "This paper addresses the problem of trading-off
between the minimization of program  and data memory requirements of
single-processor implementations of… ";
string query = "";
StringBuilder data = new StringBuilder();
data.Append("appid=" + HttpUtility.UrlEncode(appId));
data.Append("&context=" + HttpUtility.UrlEncode(context));
data.Append("&query=" + HttpUtility.UrlEncode(query));
byte[] byteData = UTF8Encoding.UTF8.GetBytes(data.ToString());
request.ContentLength = byteData.Length;
using (Stream postStream = request.GetRequestStream())
{
    postStream.Write(byteData, 0, byteData.Length);
}
using (HttpWebResponse response = request.GetResponse() as
HttpWebResponse)
{
StreamReader reader = new StreamReader(response.GetResponseStream());
Console.WriteLine(reader.ReadToEnd());
}
}
}
```

The response from the Yahoo! Web-Service would look some thing like this.

```
<?xml version="1.0"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:cate" xsi:schemaLocation="urn:yahoo:cate
http://api.search.yahoo.com/ContentAnalysisService/V1/TermExtractionRes
ponse.xsd">
  <Result>digital signal and image processing</Result>
  <Result>memory optimization</Result>
  <Result>appearance schedules</Result>
  <Result>optimization step</Result>
  <Result>buffer memory</Result>
  <Result>program memory</Result>
</ResultSet>
```

The next step is to create a SQL server project using Visual Studio 2008 and then add a new User Defined Function to that project. Now the visual studio IDE creates a template for the UDF specified. There are some DLL's that need to be registered on the SQL server in order to make the C# code work. One such package is the System.Net class which contains utilities such as `HttpUtility` class. In order to register that package we need to execute the following code on the SQL server as a 'sa' (Super Admin).

```
CREATE ASSEMBLY [System.Web]
AUTHORIZATION dbo
FROM
'C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.dll'
WITH PERMISSION_SET = UNSAFE;
```

This code registers the System.Web DLL inside the SQL Server. If there are errors in executing the assembly registration try using the following statement on SQL server and then run the above query.

```
EXEC sp_changedbowner 'sa'
```

This will make the super admin the database owner. Now right click the SQL server project in the Visual Studio IDE, go to properties and set the permission level in the database tab to unsafe. Now open the Assemblyinfo.cs file inside the project and add the following lines of code to it.

```
using System.Security;
[assembly: AllowPartiallyTrustedCallers]
```

This will allow the UDF to be called by some partially trusted users (such as IIS user). Once this is done I copied the code from my test project to the SQL Server project and set the return value of the UDF as the string from the webservice, now the UDF is ready to be used.

I wrote a wrapper stored procedure that calls this UDF for the top 5000 rows where the keywords are NULL and then dumps the result back inside the appropriate rows. As there are no calls from the front end to the database, this method is quicker that the first method, however the web-service call takes the same amount of time.

Paper Ranking Algorithm

As you may recall the paper rank algorithm is used to rank all the papers in the database and is used in ordering the results of a paper search query and also in calculating weight of each paper and author. The paper ranking algorithm is used to rank all the papers in the database and these ranks would be used in computing the importance of every paper and

there by help in computing the importance of every author as well. This paper ranking algorithm is based upon the Page Rank algorithm [8]. PageRank is a trademark of Google. Google uses PageRank to rank all the pages on the World Wide Web, and use its result for its world famous search engine. The PageRank algorithm gained its fame largely after the huge success of Google's web site. The PageRank is an iterative algorithm that calculates the rank of a page based upon the ranks of the pages that link to this page. There fore the rank of a page not only depends upon how many pages connect to this page but also upon how important those pages are.

The formula for the rank of a page is given by.

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right).$$

Figure 5: Page Rank Formula [8]

Where PR (A) is the page rank of the page A, the pages B, C, D etc are the pages that link to the page A , L(X) is the total number of pages that are being linked from page X , N is he total number of pages considered in the universe of pages and 'd' is the damping factor ,which is the probability that a person will continue browsing. It is usually set to 0.85 as per the PageRank algorithm. When pages without links are encountered, they are considered to be linked to all the other pages in the universe.

A Similar approach could be applied to this project in order to rank all the papers inside the database. In order to do this we need to decide upon a criteria or a metric based upon which the papers are ranked. In the page rank the rank was decided upon how many

pages link to a particular page, but in case of ranking papers we decide the rank by checking how many papers cite this paper and how important those papers are. There is a reason behind choosing the paper rank of the papers that cite a particular paper as the ranking metric, the reasoning is as follows. If a paper is being cited by lot of other papers then it is obvious that that paper has good recognition and hence this algorithm gives such papers a high rank. Some paper could be referenced by papers that have a higher level of importance themselves in this case the base paper would be assigned a much higher level of importance. The page rank also depends upon the number of other citations used by the papers citing this paper, so that we give a fair rank to all the papers. For the purpose of paper ranking the PageRank formula shown in the above figure should be used by redefining all the terms in the formula with respect to papers, references and citations.

PR (A) becomes the PaperRank of the paper A. Papers B, C, D etc cite paper A as their reference. L(X) is the total number of papers cited by paper X. N is the total number of papers in our database. We maintain the same value for the damping factor here.

Usually the Google always keeps on ranking the page as there are new page getting added to WWW every moment, hence it would usually take 30-40 iterations for a page rank to be more accurate. But in our case the number of papers is more or less fixed and there are not going to be many frequent additions and hence the papers would be ranked app in 3-5 iterations.

Tier – 2

This layer consists of the web service component which gives the actual outcome of all these computations. This layer acts a medium that receives queries and flushes the final result.

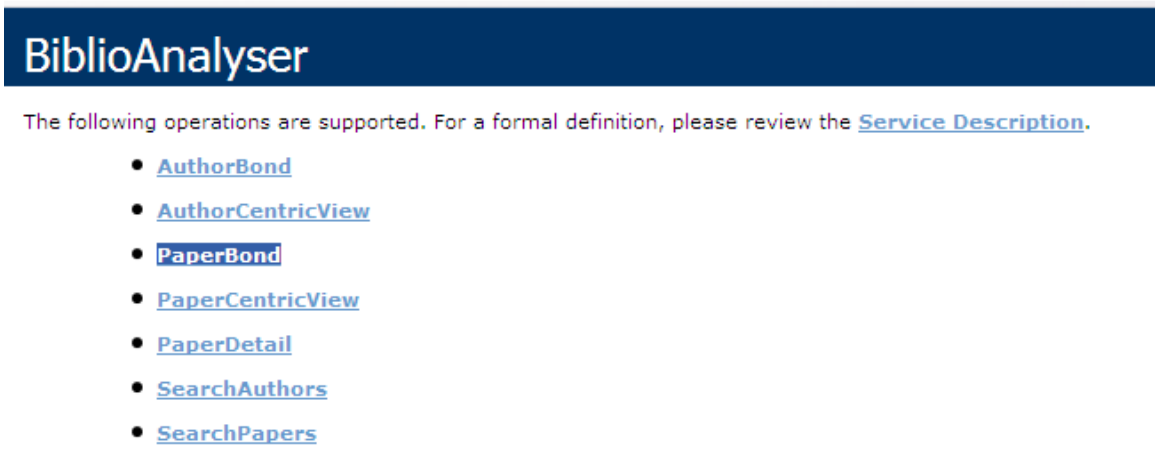

Figure 6: Webservice Snapshot

.NET Web-Service

A web-service is a programming interface that can be invoked over HTTP. The first attempts to standardize such services made use of things like WSDL, SOAP, UDDI, etc. The server that consumes the web-service must implement a proxy for the web-service and build its code base over that proxy. The methods exposed by the web-service are called through the Proxy on the server consuming the service. This function call happens as a remote procedure call to the server providing the web-service. The data communication takes place via XML.

The connectivity between the web-service at Tier - 2 and the database at Tier - 1 is done using SQL queries. In order to have a quick connection and response time from the database, Microsoft's LINQ (Language Integrated Query) technology is used for calling the stored procedures in the database and retrieving the results returned by the stored-procedures. The usage of LINQ further facilitates easier organizing of the retrieved data at the server-side code. Since the database is used to do majority of the clustering the processing is quicker, using LINQ to connect to the database makes the connectivity faster and as a result the end result reaches the consumer much quickly than what is expected out of a heavy weight web-service call.

The web-service created for this project is "BiblioAnalyser". This web-service exposes seven methods that could be invoked over HTTP. These methods are

1. SearchAuthors
2. SearchPapers
3. AuthorBond
4. PaperBond
5. AuthorCentricView
6. PaperCentricView
7. PaperDetail

SearchAuthors

This method takes in the author name or a part of the author name as input and returns all the author names and Ids that match with this name from [authors] table in the database. After receiving the response, end user selects one author from the returned result set and provides it as input to the AuthorCentricView. Sample output can be found in the appendix.

SearchPapers

This method takes in the paper name or a part of the paper name as input and returns all the paper names and paper Ids of the papers whose keywords or title match the search query. The values are fetched from the [papers] table in the database and ordered by the papers page rank such that the most important papers are shown first. After receiving the response, end user selects one paper from the returned result set and provides it as input to the PaperCentricView. A sample XML output can be found in the appendix.

Author Bond

This method takes in two author ids as input and returns the bond value between them. A sample XML response is available in the appendix.

Paper Bond

This method takes in two paper ids as input and returns the bond value between them. A sample XML response is available in the appendix.

Author Centric View

This method takes the author id as input. The author id can be retrieved as a result of the SearchAuthors procedure. After taking in the author id as input the method retrieves the author centric view details for that author from the database. The results contain the author name, id and weight. The next part of the results contains the list of Co-authors along with their name, id, weight and bond. The next part of the result contains the list of publications done by this author, each publication's paper id, title, description, authors and weight. The final part of the result contains the author bond matrix. This matrix contains the details of the bond between each and every author (including the queried author) specified in the co-author list. This enables a visualization tool to place the authors at appropriate distance from each other. A sample XML response for the author centric view is available in the appendix.

PaperCentricView

This method takes the paper id as input. The pape rid can be retrieved as a result of SearchPapers procedure. After retrieving the paper id, it is supplied as input to this procedure; the procedure sends this paper id to the database and retrieves the paper-centric view details from the database. The first part of the result contains the paper id, title, description and weight. The next section of the result contains the list of author ids, names, weights of authors who have authored the particular paper. The next section of the result is the references section. This section contains the list of references cited by this

paper. This includes paper id, title description and the author ids for each reference. The Final section is the referred by section, this section is similar to the references section except for the fact that the papers in this section refer to the papers that cite this paper (the paper that is being queried) as their reference. A sample XML response for the paper-centric view can be seen in the appendix.

Paper Detail

This method takes in a paper id as its input and it then computes all the papers relevant to this paper (papers that are directly related).This algorithm then tries to get the bonds between all the possible paper pairs available in the fetched set of papers. This data is fed into a self organizing map program that automatically trains the input nodes (paper ids) and gives two dimensional co-ordinates (X,Y) for each paper such that the distances in the map are relative to the bond between papers. This form of representation is also called multi dimensional scaling. A sample XML response is available in the appendix.

Conclusion

The outcome of this project can be used to serve multiple purposes, it can be used by any bibliographic data visualization tool to visualize all these information in different ways and bring out many more hidden relations present in the data. This information could also be used as a formatted metadata for websites tracking bibliographic records and trying to implement custom search techniques.

As a means to test our webservice I tried using this webservice on a bibliographic visualizer that is currently being build by one of the students of Dr.Teoh . Still we did not get a chance to test the Self-organizing maps based clustering , so in order to test that I developed my own algorithm to plot the nodes on a 2D graph (papers) at the X,Y co-ordinates that are retrieved as the output. A sample is shown in the figure labeled Figure-2.

As my future work on this project I would like to load this application on a real-time web server and implement various methods of caching to speed up all the queries and speed up the response from the server. I would also like to buy an unrestricted license from Yahoo-inc in order to make unlimited calls to their webservice every day; this would help me to generate keywords for a greater amount of papers in a short span of time. I would also like to buy a server and a domain and server and host my project.

Appendix

Citeseer OAI-XML

```
<record>
  <header>
    <identifier>oai:CiteSeerPSU:225109</identifier>
    <datestamp>1970-01-01</datestamp>
    <setSpec>CiteSeerPSUset</setSpec>
  </header>
  <metadata>
    <oai_citeseer:oai_citeseer
xmlns:oai_citeseer="http://copper.ist.psu.edu/oai/oai_citeseer/"
xmlns:dc ="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://copper.ist.psu.edu/oai/oai_citeseer/
http://copper.ist.psu.edu/oai/oai_citeseer.xsd ">
      <dc:title>Sample Paper Title to avoid turnit in issues</dc:title>
      <oai_citeseer:author name="Paul Patrick">
      <address>
      Gower Street Gower Street; London WC1E 6BT London WC1E 6BT
      </address>
      <affiliation>
      Department of Computer Science Department of Computer Science;
      University College London University College London
      </affiliation>
      </oai_citeseer:author>
      oai_citeseer:author name="White Jon Crowcroft">
      <address>
      Gower Street Gower Street; London WC1E 6BT London WC1E 6BT
      </address>
      <affiliation>
      Department of Computer Science Department of Computer Science;
      University College London University College London</affiliation>
      </oai_citeseer:author>
      <oai_citeseer:author name="England England">
      </oai_citeseer:author>
      <dc:subject>
      Paul Patrick,White Jon Crowcroft,England England Sample Paper
      Title to avoid turnit in issues </dc:subject>
      <dc:description>
      Sample Paper Description to avoid turnit in issues
      Sample Paper Description to avoid turnit in issues
      Sample Paper Description to avoid turnit in issues
      </dc:description>
      <dc:contributor>
      The Pennsylvania State University CiteSeer Archives
      </dc:contributor>
      <dc:publisher>unknown</dc:publisher>
```

```
      <dc:date>1970-01-01</dc:date>
      <oai_citeseer:pubyear>1998</oai_citeseer:pubyear>
      <dc:format>ps</dc:format>
      <dc:identifier>http://citeseer.ist.psu.edu/225109.html
      </dc:identifier>
      <dc:source>ftp://cs.ucl.ac.uk/darpa/WWWcos.ps.Z</dc:source>
      <dc:language>en</dc:language>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:54612</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:36088</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:9741</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:2342</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:507336</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:32808</oai_citeseer:uri>
      </oai_citeseer:relation>
      <oai_citeseer:relation type="References">
        <oai_citeseer:uri>oai:CiteSeerPSU:226021</oai_citeseer:uri>
      </oai_citeseer:relation>
      <dc:rights>unrestricted</dc:rights>
    </oai_citeseer:oai_citeseer>
  </metadata>
</record>
```

Parsed XML

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <record>
    <header>
      <identifier>1</identifier>
      <datestamp>1993-08-11</datestamp>
      <setSpec>CiteSeerPSUset</setSpec>
    </header>
    <metadata>
      <oai_citeseer>
        <title> Sample Paper Title to avoid turnit in issues </title>
        <author name="Gabriele Scheler">
          <address>80290 Munchen , Germany</address>
          <affiliation>
            Institut fur Informatik; Technische Universitat Munchen
            </affiliation>
        </author>
```

```
        <subject> Sample Paper Title to avoid turnit in issues
</subject>
        <description>
          Sample Paper Description to avoid turnit in issues
          Sample Paper Description to avoid turnit in issues
          Sample Paper Description to avoid turnit in issues
        </description>
        <contributor>The Pennsylvania State University CiteSeer
Archives</contributor>
        <publisher>unknown</publisher>
        <date>1993-08-11</date>
        <format>ps</format>
        <identifier>http://citeseer.ist.psu.edu/1.html</identifier>
        <source>ftp://flop.informatik.tu-muenchen.de/pub/fki/fki-179-
93.ps.gz</source>
        <language>en</language>
        <rights>unrestricted</rights>
      </oai_citeseer>
    </metadata>
  </record>
</root>
```

XML to SQL: XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:xdt="http://www.w3.org/2005/xpath-datatypes"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <xsl:output version="4.0" method="text" indent="no" encoding="UTF-8"
doctype-public="-//W3C//DTD HTML 4.0 Transitional//EN" doctype-
system="http://www.w3.org/TR/html4/loose.dtd"/>
  <xsl:template match="/">
    <xsl:for-each select="root">
      <xsl:for-each select="record">
        <xsl:text>INSERT INTO [dbo].[master]
([paper_id],[title],[description],[publisher],[date],[published_year],[
contributor],[format],[language],[rights],[author_names],[reference_ids
]) VALUES</xsl:text>
        <xsl:text>(&apos;</xsl:text>
        <xsl:for-each select="header">
          <xsl:for-each select="identifier">
            <xsl:apply-templates/>
          </xsl:for-each>
        </xsl:for-each>
        <xsl:text>&apos;,</xsl:text>
        <xsl:for-each select="metadata">
          <xsl:for-each select="oai_citeseer">
            <xsl:text>&apos;</xsl:text>
            <xsl:for-each select="title">
              <xsl:apply-templates/>
```

```
      </xsl:for-each>
      <xsl:text>&apos;,LEFT(&apos;</xsl:text>
      <xsl:for-each select="description">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,8000),&apos;</xsl:text>
      <xsl:for-each select="publisher">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,LEFT(&apos;</xsl:text>
      <xsl:for-each select="date">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,10),LEFT(&apos;</xsl:text>
      <xsl:for-each select="pubyear">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,4),&apos;</xsl:text>

      <xsl:for-each select="contributor">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,LEFT(&apos;</xsl:text>
      <xsl:for-each select="format">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,5),LEFT(&apos;</xsl:text>
      <xsl:for-each select="language">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,5),LEFT(&apos;</xsl:text>
      <xsl:for-each select="rights">
        <xsl:apply-templates/>
      </xsl:for-each>
      <xsl:text>&apos;,20),&apos;</xsl:text>
      <xsl:for-each select="author">
        <xsl:for-each select="@name">
          <xsl:value-of select="string(.)"/>
        </xsl:for-each>
        <xsl:text>:</xsl:text>
      </xsl:for-each>
      <xsl:text>&apos;,&apos;</xsl:text>
      <xsl:for-each select="relation">
        <xsl:for-each select="uri">
          <xsl:apply-templates/>
        </xsl:for-each>
        <xsl:text>:</xsl:text>
      </xsl:for-each>
      <xsl:text>&apos;</xsl:text>
    </xsl:for-each>
  </xsl:for-each>
  <xsl:text>)</xsl:text>
  <xsl:text>&#10;&#10;&#10;</xsl:text>
</xsl:for-each>
```

```
      <br/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Generated SQL

```
INSERT INTO [dbo].[master]
([paper_id],[title],[description],[publisher],[date],[published_year],[
contributor],[format],[language],[rights],[author_names],[reference_ids
]) VALUES('2','Sample Paper title to avoid turnit in issues',LEFT('
Sample Paper description to avoid turnit in issues. Sample Paper
description to avoid turnit in issues. Sample Paper description to
avoid turnit in issues. Sample Paper description to avoid turnit in
issues. Sample Paper description to avoid turnit in issues. Sample
Paper description to avoid turnit in issues. Sample Paper description
to avoid turnit in issues. Sample Paper description to avoid turnit in
issues. Sample Paper description to avoid turnit in issues. Sample
Paper description to avoid turnit in issues. Sample Paper description
to avoid turnit in issues. Sample Paper description to avoid turnit in
issues. Sample Paper description to avoid turnit in issues. Sample
Paper description to avoid turnit in
issues...',8000),'unknown',LEFT('1997-11-01',10),LEFT('1991',4),'The
Pennsylvania State University CiteSeer
Archives',LEFT('ps',5),LEFT('en',5),LEFT('unrestricted',20),'Xianshu
Kong:Hazel Everett:Godfried Toussaint','97473:154288')
```

Author Search: Sample XML

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <authors>
    <author>
      <id>1260</id>
      <name>Johanna D. Moore</name>
    </author>
    <author>
      <id>1703</id>
      <name>J. R. Jagannatha Rao</name>
    </author>
    <author>
      <id>1848</id>
      <name>Viktor K. Prasanna</name>
    </author>
    <author>
      <id>2222</id>
      <name>Kannan Panchapakesan</name>
```

```
    </author>
    <author>
      <id>4171</id>
      <name>Anna Lubiw</name>
    </author>
    <author>
      <id>4214</id>
      <name>Mihalis Yannakakis</name>
    </author>
    <author>
      <id>4336</id>
      <name>Sanjeev Khanna</name>
    </author>
  </authors>
</root>
```

Paper Search: Sample XML

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <papers>
    <paper>
      <id>10025</id>
      <title>Sample Paper1</title>
    </paper>
    <paper>
      <id>10167</id>
      <title> Sample Paper2</title>
    </paper>
    <paper>
      <id>10211</id>
      <title> Sample Paper3</title>
    </paper>
    <paper>
      <id>10265</id>
      <title> Sample Paper4</title>
    </paper>
    <paper>
      <id>10340</id>
      <title> Sample Paper5</title>
    </paper>
    <paper>
      <id>10343</id>
      <title> Sample Paper6</title>
    </paper>
    <paper>
      <id>10575</id>
      <title> Sample Paper7</title>
    </paper>
    <paper>
      <id>10601</id>
```

```
      <title> Sample Paper8</title>
    </paper>
    <paper>
      <id>10723</id>
      <title> Sample Paper9</title>
    </paper>
  </papers>
</root>
```

Paper Bond: Sample XML

```
<?xml version="1.0" encoding="utf-8"?>
<root>
<bonds>
  <bond paper1="10018" paper2="18450" >0</bond>
</bonds>
</root>
```

Paper Bond: Sample XML

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <bonds>
    <bond author1="39798" author2="39797" >100</bond>
  </bonds>
</root>
```

Author Centric View: Sample XML

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <author>
    <id>28295</id>
    <name>S. Resnick</name>
    <weight>2</weight>
    <coauthors>
      <author>
        <id>28293</id>
        <name>C. Davatzikos</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>28294</id>
        <name>M. Vaillant</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>28296</id>
```

```
        <name>J. L. Prince</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>28297</id>
        <name>S. Letovsky</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>39796</id>
        <name>J. Geluk</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>39797</id>
        <name>L. De Haan</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
      <author>
        <id>39798</id>
        <name>C. St Aric A</name>
        <weight>1</weight>
        <bond>1</bond>
      </author>
    </coauthors>
    <publications>
      <paper>
        <id>99618</id>
        <title>Sample Paper title to avoid turnitin issues</title>
        <description>
          <![CDATA[. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues. Sample
```

```
paper description to avoid turnit in issues. Sample paper description
to avoid turnit in issues. ...]]>
        </description>
        <weight>0</weight>
        <authors>
          <id>28293</id>
          <id>28294</id>
          <id>28295</id>
          <id>28296</id>
          <id>28297</id>
        </authors>
      </paper>
      <paper>
        <id>12236</id>
        <title> Sample Paper title to avoid turnitin issues </title>
        <description>
          <![CDATA[.Sample paper description to avoid turnit in issues.
Sample paper description to avoid turnit in issues. Sample paper
description to avoid turnit in issues. Sample paper description to
avoid turnit in issues. Sample paper description to avoid turnit in
issues. Sample paper description to avoid turnit in issues...]]>
        </description>
        <weight>0</weight>
        <authors>
          <id>39796</id>
          <id>39797</id>
          <id>28295</id>
          <id>39798</id>
        </authors>
      </paper>
    </publications>
  </author>
  <bonds>
    <bond author1="28295" author2="28293" >1</bond>
    <bond author1="28295" author2="28294" >1</bond>
    <bond author1="28295" author2="28296" >1</bond>
    <bond author1="28295" author2="28297" >1</bond>
    <bond author1="28295" author2="39796" >1</bond>
    <bond author1="28295" author2="39797" >1</bond>
    <bond author1="28295" author2="39798" >1</bond>
    <bond author1="28293" author2="28293" >1</bond>
    <bond author1="28293" author2="28294" >1</bond>
    <bond author1="28293" author2="28296" >1</bond>
    <bond author1="28293" author2="28297" >1</bond>
    <bond author1="28293" author2="39796" >0</bond>
    <bond author1="28293" author2="39797" >0</bond>
    <bond author1="28293" author2="39798" >0</bond>
    <bond author1="28294" author2="28294" >1</bond>
    <bond author1="28294" author2="28296" >1</bond>
    <bond author1="28294" author2="28297" >1</bond>
    <bond author1="28294" author2="39796" >0</bond>
    <bond author1="28294" author2="39797" >0</bond>
    <bond author1="28294" author2="39798" >0</bond>
    <bond author1="28296" author2="28296" >1</bond>
```

```
    <bond author1="28296" author2="28297" >1</bond>
    <bond author1="28296" author2="39796" >0</bond>
    <bond author1="28296" author2="39797" >0</bond>
    <bond author1="28296" author2="39798" >0</bond>
    <bond author1="28297" author2="28297" >1</bond>
    <bond author1="28297" author2="39796" >0</bond>
    <bond author1="28297" author2="39797" >0</bond>
    <bond author1="28297" author2="39798" >0</bond>
    <bond author1="39796" author2="39796" >1</bond>
    <bond author1="39796" author2="39797" >1</bond>
    <bond author1="39796" author2="39798" >1</bond>
    <bond author1="39797" author2="39797" >1</bond>
    <bond author1="39797" author2="39798" >1</bond>
    <bond author1="39798" author2="39798" >1</bond>
  </bonds>
</root>
```

Paper-centric View: Sample XML

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <paper>
    <id>978</id>
    <title> Sample Paper title to avoid turnitin issues </title>
    <description>
      Sample Paper Description to avoid turnit in issues. Sample Paper
Description to avoid turnit in issues. Sample Paper Description to
avoid turnit in issues. Sample Paper Description to avoid turnit in
issues. Sample Paper Description to avoid turnit in issues. Sample
Paper Description to avoid turnit in issues...
    </description>
    <weight>0</weight>
    <authors>
      <author>
        <id>3022</id>
        <name>Nir Shavit</name>
        <weight>8</weight>
      </author>
      <author>
        <id>3023</id>
        <name>Dan Touitou</name>
        <weight>2</weight>
      </author>
    </authors>
    <references>
      <paper>
        <id>14421</id>
        <title> Sample Paper Title to avoid turnit in issues.</title>
        <description> Sample Paper Description to avoid turnit in
issues. Sample Paper Description to avoid turnit in issues. Sample
Paper Description to avoid turnit in issues. Sample Paper Description
to avoid turnit in issues. Sample Paper Description to avoid turnit in
```

```
issues. Sample Paper Description to avoid turnit in
issues.</description>
        <weight>32</weight>
        <authors>
          <id>3554</id>
          <id>40634</id>
          <id>41473</id>
          <id>25008</id>
          <id>5983</id>
          <id>41474</id>
          <id>3553</id>
          <id>41475</id>
          <id>38473</id>
        </authors>
      </paper>
    </references>
    <referredby></referredby>
  </paper>
</root>
```


Paper Detail: Sample XML


```
<?xml version="1.0" encoding="utf-8"?>
<locations>
  <p>
    <paper>
      <id>10017</id>
      <title> Sample Paper Title to avoid turnit in issues.</title>
      <rank>00001.1057140682</rank>
      <color>0xFF8D39</color>
      <X>99</X>
      <Y>0</Y>
    </paper>
    <paper>
      <id>11543</id>
      <title> Sample Paper Title to avoid turnit in issues.</title>
      <rank>00000.1976601305</rank>
      <color>0xFFFFFF</color>
      <X>20</X>
      <Y>62</Y>
    </paper>
    <paper>
      <id>12363</id>
      <title> Sample Paper Title to avoid turnit in issues.</title>
      <rank>00000.2744855273</rank>
      <color>0xFFFFFF</color>
      <X>0</X>
      <Y>24</Y>
    </paper>
    <paper>
      <id>12606</id>
```

```
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>26</X>
  <Y>99</Y>
</paper>
<paper>
  <id>13158</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>10</X>
  <Y>90</Y>
</paper>
<paper>
  <id>13230</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.2690887020</rank>
  <color>0xFFFFFF</color>
  <X>51</X>
  <Y>46</Y>
</paper>
<paper>
  <id>14645</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>0</X>
  <Y>50</Y>
</paper>
<paper>
  <id>18130</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>12</X>
  <Y>99</Y>
</paper>
<paper>
  <id>1840</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.4755982381</rank>
  <color>0xFFFFFF</color>
  <X>99</X>
  <Y>99</Y>
</paper>
<paper>
  <id>18450</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.1233744162</rank>
  <color>0xFFFFE9</color>
  <X>0</X>
  <Y>7</Y>
</paper>
```

```
<paper>
  <id>187573</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>99</X>
  <Y>66</Y>
</paper>
<paper>
  <id>188199</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.1131201723</rank>
  <color>0xFFCA83</color>
  <X>50</X>
  <Y>0</Y>
</paper>
<paper>
  <id>5030</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>0</X>
  <Y>99</Y>
</paper>
<paper>
  <id>8727</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0200730819</rank>
  <color>0x00FFFF</color>
  <X>41</X>
  <Y>99</Y>
</paper>
<paper>
  <id>92692</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0472252223</rank>
  <color>0x00FCFF</color>
  <X>48</X>
  <Y>83</Y>
</paper>
<paper>
  <id>9407</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0473119524</rank>
  <color>0x0078FF</color>
  <X>74</X>
  <Y>99</Y>
</paper>
<paper>
  <id>96621</id>
  <title> Sample Paper Title to avoid turnit in issues.</title>
  <rank>00000.0000030000</rank>
  <color>0x00FF00</color>
  <X>0</X>
```

```
      <Y>76</Y>
    </paper>
    <paper>
      <id>97414</id>
      <title> Sample Paper Title to avoid turnit in issues.</title>
      <rank>00000.1277266740</rank>
      <color>0xFFFFFF</color>
      <X>21</X>
      <Y>0</Y>
    </paper>
  </p>
</locations>
```

References

[1] *Cluster analysis.* (2009). http://en.wikipedia.org/wiki/Data_clustering

[2] *Getting started with the API.*http://developer.constantcontact.com/node/35

[3] Kuldeep, C. (2008). *Accessing REST based web services using SQL CLR.* Retrieved 06/18, 2008, from http://blogs.msdn.com/sqllive/archive/2008/06/18/accessing-rest-based-web-services-using-sql-clr.aspx

[4] Lawrence, S., Giles, L. & Bollacker, K. (2005). *Citeseer.*http://citeseer.ist.psu.edu/oai.html

[5] Ley, M. (2008). *DBLP.*, 2008, from http://www.informatik.uni-trier.de/~ley/db/

[6] Microsoft Corporation. (2008). *LINQ: .NET language-integrated query.*, 2008, from http://msdn.microsoft.com/en-us/library/bb308959.aspx

[7] Microsoft Corporation. (2008). *SQL server programming overview.*, 2008, from http://msdn.microsoft.com/en-us/library/ms166342(SQL.90).aspx

[8] *PageRank.* (2009). http://en.wikipedia.org/wiki/PageRank

[9] Refsnes, H., Refsnes, S. & Refsnes, J. E. (2008). *XSLT tutorial.*, 2008, from http://w3schools.com/xsl/default.asp

[10] *Representational state transfer.* (2009). Retrieved 04/19, 2009, from http://en.wikipedia.org/wiki/Representational_State_Transfer

[11] *Search web services: Contextual web search.* (2009). http://developer.yahoo.com/search/web/V1/contextSearch.html

[12] *Self- organizing map.* (2009). http://en.wikipedia.org/wiki/Self-organizing_map

[13] Shen, Z., Ogawa, M., Teoh, S. T., & Ma, K. (2004). BiblioViz: A system for visualizing bibliography information.

[14] *Silverlight: Accessing HTTP web services.* (2009). http://developer.yahoo.com/dotnet/silverlight/2.0/webservice.html

[15] Strahl, R. *Creating and using web services with the .NET framework and visual studio.net.*http://www.west-wind.com/presentations/dotnetwebservices/DotNetWebServices.asp

[16] *TRUSTWORTHY database property.* (2009). http://msdn.microsoft.com/en-us/library/ms187861.aspx

[17] Wakefield, J. (2008). *Self-organizing-map (SOM).* Retrieved 11/16, 2008, from http://dynamicnotions.blogspot.com/2008/11/c-self-organising-map-som.html

[18] *Wikipedia - Citeseer.* http://en.wikipedia.org/wiki/CiteSeer

[19] *Wikipedia- digital bibliography & library project.*

http://en.wikipedia.org/wiki/DBLP

[20] *YDN: Your gateway to yahoo! services.* (2009).

http://developer.yahoo.com/everything.html