Etcetera Industries

Recommendation Systems

Enhancing QUT ePrints Recommendation Systems with Pattern-Mining Techniques

Jonathan Gollagher, Alexander Waskiewicz, James Pyke, Nathan Moncrieff, Damien Horton
10-20-2013

Contents

1.0 Introduction

Being capable of compiling, synthesizing and collecting information is vital to an academic's job. This is especially true within the context of Science, where the body of knowledge is constantly being expanded upon, reinvestigated and self-moderated. As a part of an ongoing chain of research and development at QUT, Etcetera Industries has become the latest group working towards creating an efficient article recommendation system built around currently in-development data and pattern-mining algorithms.

Upgrading the recommendation system to use more complex and robust algorithms is, ultimately, an exercise in pure research and development. Regardless of the outcome, this iteration focuses on assessing the viability of new technologies, built upon

The Article Recommendation System utilizes content from QUT's ePrints, a repository of articles published by QUT, and this system currently focuses on the Science Faculty's section. Being able to quickly and concisely provide users with similar articles to the ones they are currently searching for is a useful and important tool, as sometimes a user may not know exactly what they are looking for. In addition to this, it can assist them in locating information they may not have known to look for, such as more recent studies on the same topic.

This document is designed to be an intermediary progress report, documenting the latest iteration of this project so that the next group to work on it will have an adequate basis for their own improvement of the program. Most notably, the implementation of data-mining and pattern-mining will be discussed at length, as these are the primary contributions to the program that this iteration possesses.

As this is a multi-stage development, this report largely covers the current iteration of the project, with a small section dedicated to the state of the system prior to this update. This is followed by a change log of current implementation, which is then followed by a more in-depth look at the state of this iteration. Finally, the testing and results of this iteration will be included, which will explain in detail the knowledge gained from the implementation of the iteration.

2.0 System Prior to update

The prior implemented system follows architecture with 3 levels; Data Tier, Business Logic and User interface where the top layer does not interact with the bottom layer.

The data tier used was SQL based storing data from the QUT E-prints website. Using the E-prints as data storage meant that files were easily accessible for the search, but it still needed to be much more efficient and appealing.

The business logic had 3 levels; top, middle and bottom level. The top level made to retrieve the information chosen from the database, the middle level calculated using a filtering algorithm and a user frequency algorithm to present the score of the article to ensure accuracy of recommendation and the bottom level was to calculate with a Jaccard algorithm to ensure individual profiles.

The GUI prior to its updates functionality had a simple category expanding search method where its function was to select categories to result the top 10 results searched where the data forms a profile for the user of the searches. The GUI is too simple and need to be upgraded to something more feasible and visually appealing.

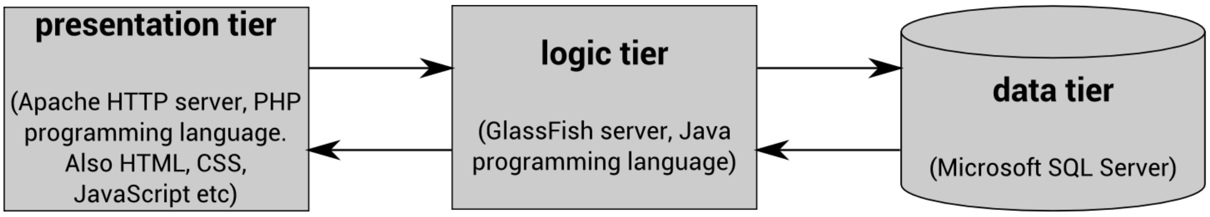
2.1 Updates

- Interface
    - Search box
    - Updated standard search method
- Mallet implementation
    - Latent Dirichlet Allocation
    - Used to categorise articles into topics
    - Assigns relevance to topic of article
- Association Rule Mining (ARM)
    - Gets topic assigned to relevance
- Web site
    - Layout
    - Replacing recommendations system

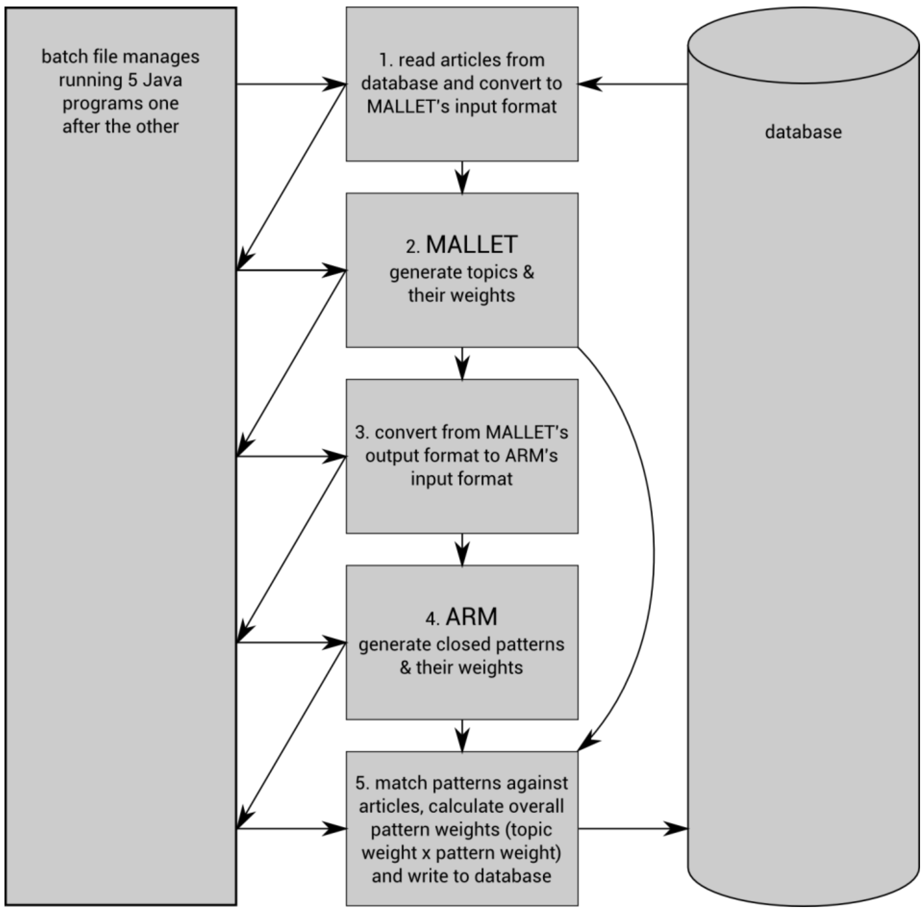
3.0 System Implementation

With the new system implementation, it still follows with the previous 3-tier architecture, which consists of; presentation tier, logic tier and data tier. The data tiers work separately with logic tier being the middleman, so there is no interaction with the presentation tier and data tier.

Presentation tier includes the web application displaying the outputs completed with various programming language, the logic tier that filters the search with Mallet to produce a sharper search and the data tier where it uses Microsoft SQL Server to hold all the articles.


presentation tier
(Apache HTTP server, PHP programming language. Also HTML, CSS, JavaScript etc)
logic tier
(GlassFish server, Java programming language)
data tier
(Microsoft SQL Server)

The data processing is launched from a batch file (Windows shell script) that in turn runs through five command-line Java apps in sequence:


batch file manages running 5 Java programs one after the other
1. read articles from database and convert to MALLET's input format
2. MALLET generate topics & their weights
3. convert from MALLET's output format to ARM's input format
4. ARM generate closed patterns & their weights
5. match patterns against articles, calculate overall pattern weights (topic weight x pattern weight) and write to database
database

1. Reads the list of categories from the database, and for each sub-category determines which top-level category it is under. Then it reads the full set of articles from the database and saves them as a text file in a folder for each top-level category the article belongs to.
2. MALLET, taken from <http://mallet.cs.umass.edu/> (not created by our team). Run multiple times to generate 20 topics for each top-level category.

3. Converts MALLET output to ARM input. Not originally created by our team, though some modifications were necessary.
4. ARM program (not created by our team). Run multiple times to generate closed patterns for each topic.
5. Final processing - reads topic distribution weights from MALLET's output (step 2), and the patterns and their support weights from ARM output (step 4). The articles (from step 1) are read in order and checked to see which patterns occur in each article. For every pattern that is matched, its pattern support value is multiplied by its topic's distribution value, to determine the pattern's overall calculated weighting for that article. This calculated value is saved along with the article's ID number and the pattern's ID back into the database.

3.1 Data Tier

The database consists of 46 tables, however at current only about 5 or so of those tables are currently being used by this software iteration.

The system's Data tier consists of an altered version of the pre-existing database, modified to accommodate the pattern-mining data-sets. These stores the patterns generated from the ARM software that exists within the Logic Tier. The information regarding the newly created table properties can be found in *section 3.1.1*.

3.1.1 Database update

The latest addition to the Capstone database, tblPatterns is a repository for the patterns identified by ARM, as it goes over the data previously generated by the LDA. The table consists of three columns:

1. **patternID** - An application generated alphanumeric identifier assigned to each individual pattern discovered during the mining process. Depending on the scope of the pattern, there may be many entries in the database associated with this pattern.
2. **ePrintID** – A numeric identifier, uniquely marking each individual QUT ePrint record.
3. **patternRating** – An objective measure of the relevance of the pattern to the overall topic the ePrint in question occupies, expressed as a decimal. The closer to 0, the less relevance the article is computed to display.

Separately, neither patternID nor ePrintID are unique. Each pattern may encompass many articles, and each article may be a part of many patterns. Together, however, they form the table's composite primary key – a unique indentifying pair. ePrintID is linked to the main ArticleInfo table, maintaining foreign key integrity.

At the time of writing, tblPatterns holds well over 800000 records, generated during the previous execution.

Properties

Property	Value
Row Count	824287
Created	01/10/13 01:25:18
Last Modified	13/10/13 07:15:09

Columns

PK	Name	Data Type	Max Length (Bytes)	Allow Nulls
Y	patternID	varchar	100	Y
Y	ePrintID	int	4	Y
N	patternRating	decimal(18,14)	32	N

Indexes

Name	Columns	Unique
PK_tblPatterns	patternID, ePrintID	Y

Foreign Keys

Name	Columns
FK_tblPatterns_tblArticleInfo	ePrintID -> [dbo].[tblArticleInfo].[ePrintID]

Uses

[dbo].[tblArticleInfo]

dbo

3.2 Logic Tier

The Logic Tier utilizes the Data tier to take the initial data provided and alter it in such a way as to be useful for implementation within the presentation tier. Utilizing Java as a programming language, the Logic Tier is powered by a GlassFish server system. The system is built from a number of pre-made systems, with intermediary code connecting them. The first system, MALLET, takes the data from the Scientific Articles database and runs it through a data-mining algorithm, finding relevant keywords within the articles' abstract texts and compiling a list of the most commonly used words. After this list is developed, the result is used as the starting point for the next module, ARM. ARM takes information generated from Latent Dirichlet Allocation (LDA) techniques to generate word patterns out of the initial list of frequently-used words. Once this new list has been generated, it can be used within the Presentation Tier, by presenting topically linked articles to articles currently being viewed.

3.2.1 User Case

The user case diagram, view in *appendix I* show the interaction between the system and the primary users of the article recommendation system; Administrator and user.

The user is able to access the interface and either search using the search box or by category. When either search is fulfilled the recommendation page shows the top 20 results where the user is either able to view the abstract or go further into search by view the related articles to the chosen article.

The administrator is able to search the database on a separate program for precision and recall testing. The search is output in a form of figures in a table to show its recall and precision testing.

3.2.2 Article recommendation generator

To generate recommendations, the other articles are assigned a score as to how similar they are to the article the user is viewing. The articles with the highest score are then recommended.

First, the reference article, which is the article the user is viewing, to generate recommendations from is checked in the database to see which patterns it contains. These patterns are then looked up to find a list of all the other articles that contain at least one of them. The articles' scores are then calculated as follows, for every pattern they have in common with the reference article:

$$score = \frac{1}{n}\sum_{i=1}^{n}(td_i \times ps_i)$$

Figure 1

td_i is the topic distribution weighting for the article in the pattern's topic which is generated by Mallet and ps_i is the pattern's support weighting. Each article's score is the mean of the topic distribution multiplied by the pattern support, for each pattern it shares with the reference article.

Implementing this algorithm, it is decided to move the multiplication of the topic distribution and pattern support to the initial data processing phase rather than by the website, as it simplifies the database storage and related queries because the values are only needed as their product and never independent of each other. The product of topic distribution and pattern support could be referred to as the "calculated weighting".

The originally algorithm had used the sum of calculated weightings instead of the mean:

$$score = \sum_{i=1}^{n}(td_i \times ps_i)$$

Figure 2

This was changed due to a concern that the results may be biased towards longer articles, which inherently contain more patterns. Using the mean may not be disadvantage, however, it may still be better to give more weighting to those articles with more pattern matches and thus presumably more relevance.

3.3 Presentation Tier

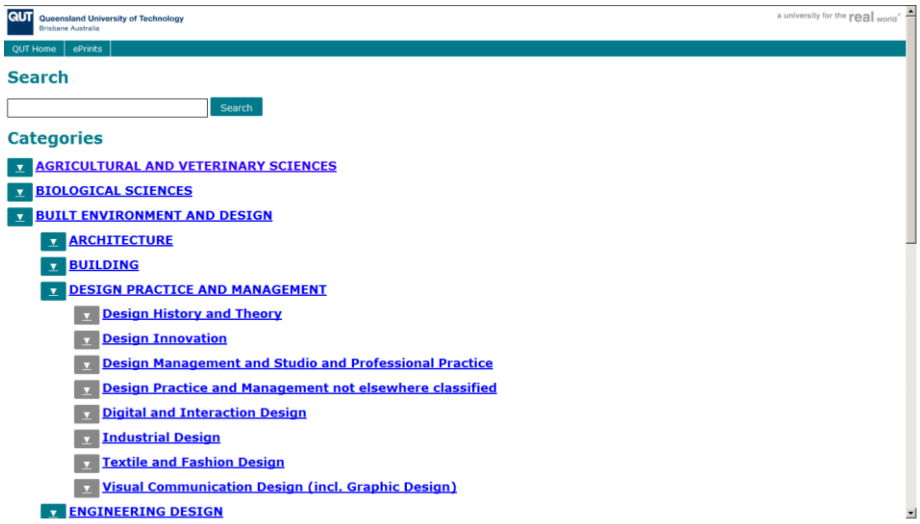
The Presentation Tier is divided into two separate sections: information provided to the end user, such as recommended articles and search functions, and the mechanical information for administrator, allowing administrators to access technical specifications of the system. The foundation of both sections is based within an Apache HTTP server, and the core website is constructed from PHP, with html, css and javascript coding used where necessary. The current iteration contains a new user interface, which will be discussed at length below.

3.3.1 Graphical User Interface

The user interfaces implemented is coded using JavaScript where it uses the fundamental logic to be moved across QUT E-prints web site. With the logic written in Java, the whole web application interface was entirely updated to keep a consistent flow to its new improvements.

The updated web application consists of three new functions; one to display the search functions to the user, one to display the recommended article results and abstracts and one to see more related articles from the chosen article. The design of the web application was similar to the previous page except this page will incorporate new functions of search that will be more appealing and simplified to the user.

The following interfaces are the updated interfaces designed according to the requirements of the user's needs.

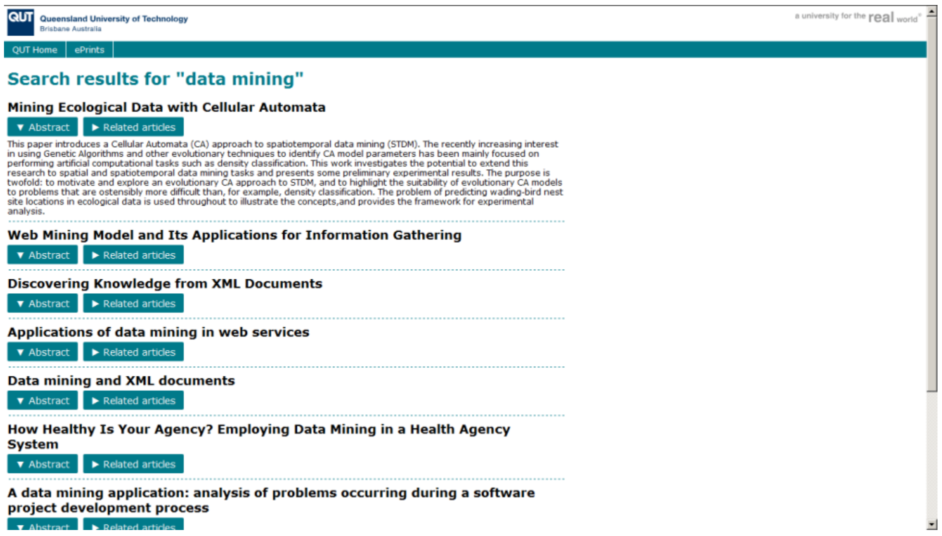

QUT Queensland University of Technology
Brisbane Australia
a university for the real world
QUT Home
ePrints
Search
Search

Categories

- AGRICULTURAL AND VETERINARY SCIENCES
- BIOLOGICAL SCIENCES
- BUILT ENVIRONMENT AND DESIGN
  - ARCHITECTURE
  - BUILDING
  - DESIGN PRACTICE AND MANAGEMENT
    - Design History and Theory
    - Design Innovation
    - Design Management and Studio and Professional Practice
    - Design Practice and Management not elsewhere classified
    - Digital and Interaction Design
    - Industrial Design
    - Textile and Fashion Design
    - Visual Communication Design (incl. Graphic Design)
  - ENGINEERING DESIGN

Figure 3: home page

A new search function is implemented to create a specific and a sharper search of articles followed with a new and improved category search, it is implemented in order to display the category types in a multilevel view point allowing you to see your path creating a much simplistic and sharper interface surpassing the previous, Figure 3: home page. When the high level categories are chosen it will include the search of all the sub categories shown in figure 4: Recommendations.


QUT Queensland University of Technology Brisbane Australia
a university for the real world
QUT Home
ePrints
▼ Abstract
► Related articles

Search results for "data mining"

Mining Ecological Data with Cellular Automata

This paper introduces a Cellular Automata (CA) approach to spatiotemporal data mining (STDM). The recently increasing interest in using Genetic Algorithms and other evolutionary techniques to identify CA model parameters has been mainly focused on performing artificial computational tasks such as density classification. This work investigates the potential to extend this research to spatial and spatiotemporal data mining tasks and presents some preliminary experimental results. The purpose is twofold: to motivate and explore an evolutionary CA approach to STDM, and to highlight the suitability of evolutionary CA models to problems that are ostensibly more difficult than, for example, density classification. The problem of predicting wading-bird nest site locations in ecological data is used throughout to illustrate the concepts,and provides the framework for experimental analysis.

Web Mining Model and Its Applications for Information Gathering

Discovering Knowledge from XML Documents

Applications of data mining in web services

Data mining and XML documents

How Healthy Is Your Agency? Employing Data Mining in a Health Agency System

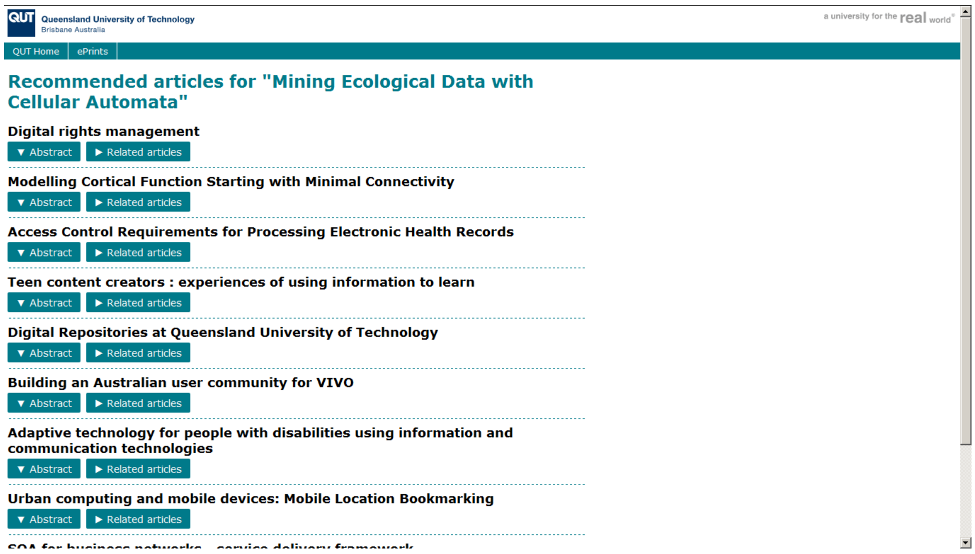
A data mining application: analysis of problems occurring during a software project development process

Figure 4: Recommendations

The recommendation result page keeps the functionality simple being able to list top 20 results; it is also updated with an abstract and recommendation article button.

When articles are recommended, the output identifies the information about the articles which include; the abstract and related articles to the recommendation so the user can either view the article or view more articles that are related. When the abstract button is pressed, the abstract is shown, see figure 4 for example and when related articles is pressed it takes you to a new search of related articles.

These upgrades are to benefit the user in identifying the wanted articles or browsing further into more articles that are related. Data is able to be saved in a form of user profile to ensure a good performance.


QUT Queensland University of Technology Brisbane Australia
a university for the real world
QUT Home
ePrints
▼ Abstract
► Related articles

Recommended articles for "Mining Ecological Data with Cellular Automata"

Digital rights management

Modelling Cortical Function Starting with Minimal Connectivity

Access Control Requirements for Processing Electronic Health Records

Teen content creators : experiences of using information to learn

Digital Repositories at Queensland University of Technology

Building an Australian user community for VIVO

Adaptive technology for people with disabilities using information and communication technologies

Urban computing and mobile devices: Mobile Location Bookmarking

Figure 5 Related articles

When related articles are pressed, it will take you to a new page, *Figure 5*, with the topics of the related articles. You also have an option to go back to the previous topic.

3.4.1 Data pre-processing

The five applications used to generate the topics and patterns needed by the website, only the first and last were created for this project.

3.4.1.1 DB2Mallet

The first of the five applications; reads articles from the database and saves them as text files, organised into different directories for top-level categories.

- **DB2Mallet**: main class
    - **main**: main method containing most functionality
- **ConnectMSSQL**: database handler class
    - **ConnectMSSQL**: constructor method, opens a connection to the database
    - **closeConnection**: closes connection to the database
    - **getCategoryParents**: reads the list of categories and their parent category from the database
    - **getArticles**: reads articles from the database

3.4.1.2 PatternCounter

Last of the five applications; reads data output by the 1st, 2nd & 4th steps, finds patterns within articles, generates their calculated weightings, and saves back into the database.

- **Main**: simple wrapper class to allow running the PatternCounter class as a new instance
    - **main**: main method
- **PatternCounter**: class containing most functionality
    - **countPatterns**: public method called by the Main class
- **ClosedPattern**: simple data structure used by the PatternCounter class
- **ConnectMSSQL**: database handler class
    - **ConnectMSSQL**: constructor method, opens a connection to the database
    - **closeConnection**: closes connection to the database
    - **clearPatterns**: wipes old pattern data from the database before new data is inserted
    - **putPattern**: inserts a pattern into the database

3.4.1.3 Batch file

The Windows batch file runs the five Java applications in sequence. To run the data pre-processing, simply run the file processing.bat and follow the prompts.

When installing the system on a new computer, it may be necessary to edit the variables in the first few lines of processing.bat to configure settings such as file paths and the database password.

3.4.2 Website

Though substantial changes were made to the website, most were implemented as changes to existing methods rather than creation of new ones. The exception to this is the search bar functionality, added as the new search.php file in the presentation tier, which in turn calls the following methods in the logic tier:

- **RecommendWS** class, **recommendBySearchTerm** method, which calls:
- **ArticleRecommender** class, **recommendBySearchTerm** method, which calls:
- **ConnectMSSQL** class, **queryArticlesBySearchTerm** method, which queries the database.

3.4.3 Testing

The testing program can be launched from the bench.bat Windows batch file, which will launch the Benchmark Java class. When the program has completed, the output data can be found in the benchresults.csv file, which can be opened in a spreadsheet program such as Microsoft Excel or LibreOffice Calc (using semicolons as field delimiters).

- **Benchmark**: main class containing most functionality
  - **main**: main method
- **ConnectMSSQL**: database handler class
  - **ConnectMSSQL**: constructor method, opens a connection to the database
  - **closeConnection**: closes connection to the database
  - **getUserIDs**: reads a list of user IDs from the database, where each user has viewed the required minimum number of articles
  - **getViewedArticles**: reads a list of articles viewed by a given user
  - **getRecommendedArticles**: recommends a list of articles similar to a given set of articles

4.0 Testing and Evaluation

As with the previous year's collaborative filtering project, the recommendations were tested against the provided user data to determine two different values; precision and recall. Precision measures the proportion of the recommended articles that are deemed to be relevant. Recall measures the proportion of relevant articles that were successfully recommended.

4.1 Methodology

Unlike the collaborative filtering project's tests, the users were not split into different groups for the different tests; all tests were run on all users. One reason is that there was no need for "training" data because this project's algorithm does not rely on user data to make its recommendations. On top of this, it was decided the testing dataset was too small to dilute between the different tests for more accurate results. This also had the advantage that more tests could be run than previously.

First the users were filtered so that only those who had viewed at least three articles were used. Then for each user, two of their viewed articles were picked at random, and used to generate recommendations (with scores based on the average calculated weighting of patterns matched in either article, not necessarily both). These recommendations were tested against the other articles the user had actually viewed, to calculate the precision and recall. These tests were run several times for the top recommendation, top 2, top 5, top 10, and so on. When all users had been tested, the average precision and recall for each test.

Like with the collaborative filtering project's addition of the new, "non-binary" algorithm test, the testing for this project added further tests with an additional modification - use the original algorithm using the sum of the calculated weights rather than the mean.

4.2 Results

	Precision		**Recall**	
Algorithm	**Average**	**Sum**	**Average**	**Sum**
Top 1	0.0000%	0.0063%	0.0000%	0.0519%
Top 2	0.0000%	0.0095%	0.0000%	0.0606%
Top 5	0.0047%	0.0190%	0.0081%	0.0938%
Top 10	0.0079%	0.0300%	0.0083%	0.3240%
Top 15	0.0158%	0.0490%	0.1366%	0.4337%
Top 20	0.0285%	0.0585%	0.2078%	0.5421%
Top 25	0.0332%	0.0696%	0.3042%	0.6872%
Top 30	0.0395%	0.0838%	0.3566%	0.8909%
Top 35	0.0459%	0.0965%	0.3735%	1.0408%
Top 40	0.0522%	0.1154%	0.4133%	1.1181%
Top 45	0.0585%	0.1360%	0.4344%	1.3482%
Top 50	0.0633%	0.1471%	0.4456%	1.5839%
Top 55	0.0696%	0.1660%	0.4555%	1.8609%
Top 60	0.0806%	0.1787%	0.4984%	2.0219%

5.0 Learning Outcomes

5.1 Average vs Sum

Using the sum of the calculated weightings gave noticeably better results for both precision and recall. Unless there was a flaw in the existing ePrints site that meant the user data is itself biased towards longer articles, it appears that the benefit of weighting towards more pattern matches may outweigh the potential problem of favouring longer articles, and so using the sum is the overall better algorithm. This is only speculation however, as the pattern mining for this project was only run on the articles' abstracts - the issue of longer articles would presumably be worse with full length articles which may have more variety of lengths.

5.2 Pattern Mining vs Collaborative Filtering

When compared to the previous year's collaborative filtering project (see appendices), our pattern mining project has noticeably worse results, particularly for precision. However it is encouraging to see that the results aren't many orders of magnitude apart; so the pattern mining project still seems to be on the right track. Possible explanations for the poor performance could include:

- Only using abstracts - the collaborative filtering algorithm didn't rely on article text at all for its recommendations, giving it a possible advantage over pattern mining, which may have performed better if used with the full article texts as it is designed to.
- Flaws in existing ePrints site - given that the pattern mining project is new and has not been implemented on the live site, the test users had to use other means to find the articles they found relevant. We can only speculate but perhaps if they had been using our system they would have found its recommendations useful and it would have tested much better as a result. The collaborative filtering system, on the other hand, doesn't suffer from this problem quite as much, since its recommendations are generated by the same set of users used to test its results - both may be skewed by whatever quirks the existing site may have, that could favour some articles over others, for example.
- Small sample size - both projects were limited in terms of the testing data available; pattern mining for example could only test on a little over a thousand users. This small size increases the likelihood of user randomness skewing the results in unpredictable ways.

The reason the precision and recall were so low at all is because many of the tests came up with zero matches between the user's viewed articles and the recommendations. Given that the collaborative filtering results weren't far ahead, it can be expected that its tests suffered from the same problem though to a lesser extent.

5.3 Precision Decline

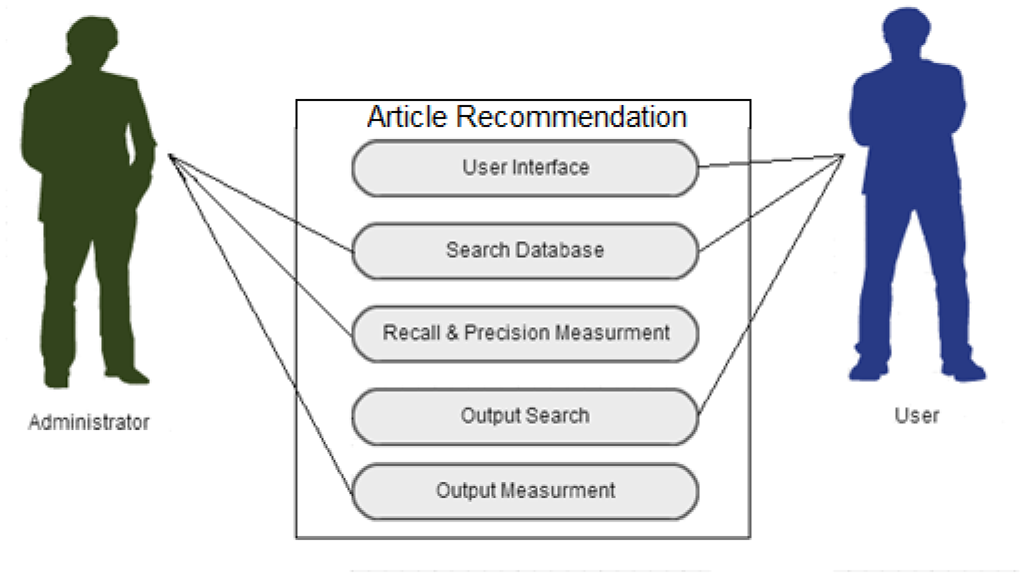
As the number of results increases, precision could be expected to level off and then decline, as reportedly happened with the collaborative filtering system. This is because each test user only viewed a limited number of articles; a hypothetical ideal recommendation algorithm that perfectly matches every test user's habits would drop off in precision once it has already recommended everything they viewed, as it can only recommend "irrelevant" articles after that. This did not occur in the tests ran on the pattern mining algorithm, as the initial recommendations were inaccurate enough that it continued to have room to improve and find more relevant articles at higher numbers of recommendations. Precision could still be expected to decline at some higher number, but our testing did not go far enough.

6.0 Conclusion

With the ever-changing landscape of Information Technology continuing to shape disciplines around it, it is important to apply new techniques to pre-existing solutions, in order to remain at the cutting edge of research and development. Implementing new pattern-mining algorithms into an article recommendation system allows for new opportunities to arise within the field, regardless of whether the actual result is fruitful. In essence, there is a wealth of knowledge available for the next stages of development, as the system continues to grow and evolve.

7.0 Appendix

Appendix I: use case diagram



Appendix II: Precision Tests

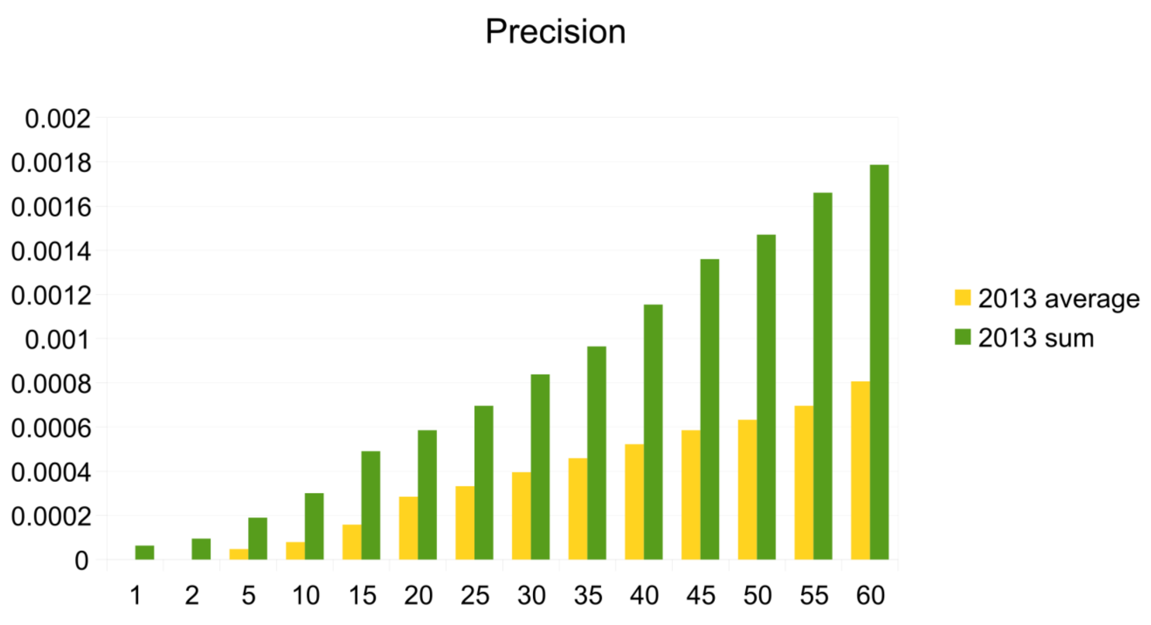


Figure 6 Precision test

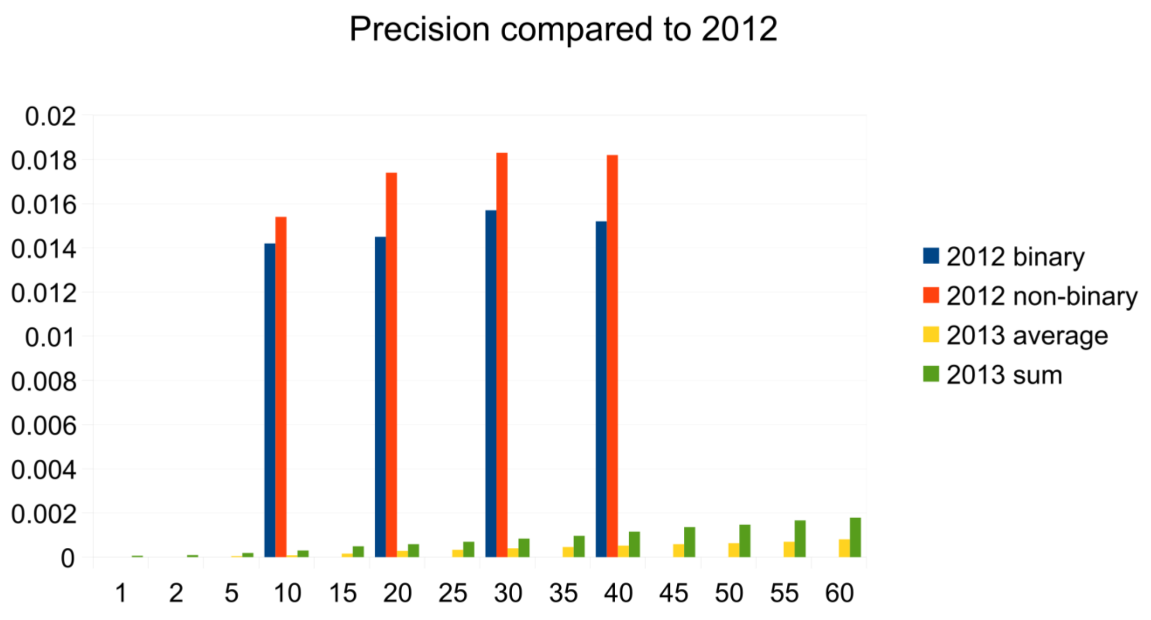


Figure 7 Precision test comparison

Appendix III: Recall Tests

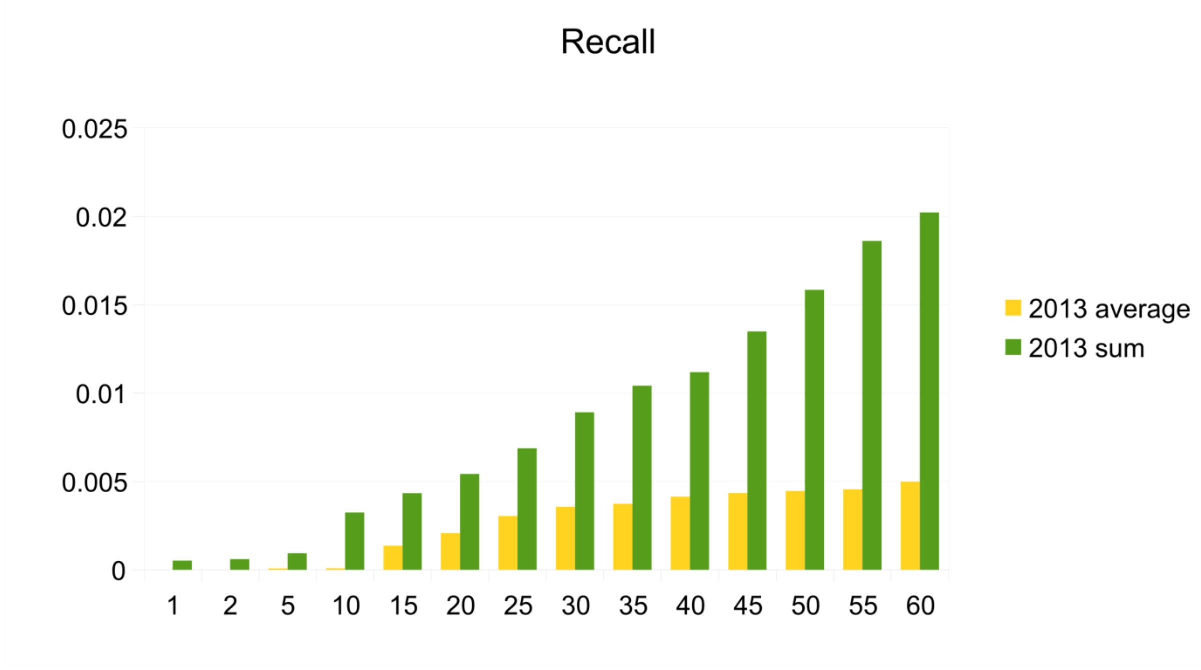


Figure 8 Recall test

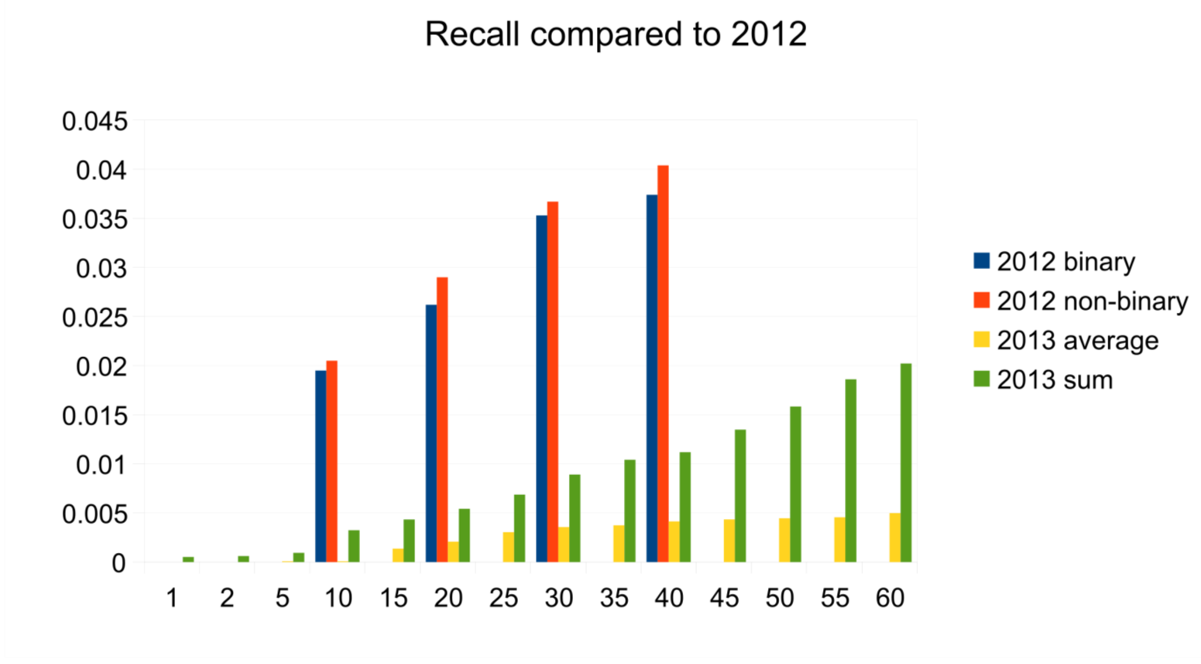


Figure 9 recall test comparison

Appendix IV Installation Guide

Most of the systems are designed for a Windows machine, though some can be adapted for other operating systems. However the database requires Windows and at this stage it is probably easiest to have all systems running on the same machine.

Database

Requires Microsoft SQL Server (we used the 2012 version) to be installed on a Windows machine. QUT IT students can download MS SQL Server and Windows free from Microsoft's DreamSpark (MSDNAA) service.

1. If SQL Server has not yet been installed, do so now, including installing the SQL Server Management Studio. You will be prompted during the install to set a name for the database server, which should be recorded for later (this can also be found later in SQL Server Management Studio as the top level item in the Object Explorer)
2. Connect to the new database server using SQL Server Management Studio. Expand the server tree, and right click on 'Databases'. Select 'Restore Database'.
3. Set 'Source' to 'Device', then click the '...' button. Click 'Add', and then browse to the**database\capstone.bak** file. Select it, and press OK to add it, and again to confirm it.
4. Click OK. Restore will proceed.

Exporting the database to install elsewhere

To pass the database on to another team for the next stage of the project's iterative cycle:

1. Create a folder to hold the backup. For the purposes of this example, we will use**C:\SQLServerBackups**
2. Open SQL Server Management Studio. Create a new query, making sure the database in the dropdown box on the top menubar is set to 'capstone'.
3. Paste in the following and then click Execute Query:
4. **BACKUP DATABASE capstone**
5. **TO DISK = 'C:\SQLServerBackups\capstone.bak'**
6. **WITH FORMAT;**
   GO

Website - logic tier

Requires the GlassFish server software, which comes bundled with the NetBeans IDE.

1. Download and install NetBeans from <https://netbeans.org/downloads/> (the Java EE bundle is recommended, though if you want to use the same IDE to edit the presentation tier you can either get the All bundle or stick with the Java EE bundle and download the PHP package later in the Plugin Manager)
2. Copy the **website\ArticleRecommendWS** folder to the desired location. In NetBeans, go to **File**->**Open Project** and browse to the **ArticleRecommendWS** folder in its new location.
3. In the Project Browser on the left, click through to **ArticleRecommendWS**->**Source Packages**->**com.qut.eprints**->**ConnectMSSQL.java**. Starting at about line 22 should be four database settings; adjust these as necessary (if you installed the database as

above, you will probably only need to change the **databaseServer** entry with your new server name).

4. To start the GlassFish server, click the Run button (green arrow in the Run toolbar at the top) or press F6. (There are other ways to start the GlassFish server but during development it's probably easiest to launch from within NetBeans).

Website - presentation tier

Requires a web server program such as Apache HTTP Server, with support for the PHP programming language. These instructions assume the use of the WampServer bundle in Windows, though other means of install are possible with your own research.

1. Download the WampServer bundle from <http://www.wampserver.com/en/> and install it.
2. Though WampServer is the quickest way to set up a PHP server in Windows, it is a bit wasteful as it comes with MySQL which we don't need. Click on the WampServer icon in the system tray (stylised W in a rounded square) and go to **MySQL**->**Service**->**Remove Service**.
3. Click the WampServer tray icon and click **www directory**. Copy the files and subfolders inside the**website\quteprints** folder into the folder that is opened (defaults to **C:\wamp\www**).
4. Click the WampServer tray icon and **PHP**->**PHP extensions**. Make sure the **php_soap** extension is enabled.
5. Open your web browser and go to **http://localhost/** - the full website should now be operational.

Data preprocessing & benchmarking components

Requires Windows to run the batch files as they are, though they could potentially be rewritten as Bash scripts etc to run under Linux or OS X (particularly the benchmarking one as it's quite short).

Copy the **processing** and **benchmarking** folders to the desired location(s). Each contains a Windows batch file that may need to be edited (any text editor will do) - to adjust the "set" commands near the top of the file with the database settings (if you installed the database as above, you will probably only need to change the **ARSdbHost** entry with your new server name).

To run the data preprocessing tools or benchmarking simply run the respective batch file.