

PaperScout

(A multi-agent system based publications tracking tool for researchers in Pakistan).

By

Osama Misbah 2009-NUST-BIT-272

Sheikh Sarosh Bilal 2009-NUST-BIT-276



School of Electrical Engineering & Computer Science (SEECS)

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

2012-2013

Certificate

It is certified that the contents of this report of Proposal Defense for “PaperScout- A multi-agent system based publications tracking tool for researchers in Pakistan” submitted by Osama Misbah and Sheikh Sarosh Bilal is total contribution of their knowledge in domain of their project. Any help from other resources are referenced in the report.

It has been found satisfactory for the requirements of Final Year Project’s Proposal Defense and also fulfill the requirements of advisor and co-advisor.

Advisor: _____

(Dr. Peter Charles Bloodsworth)

Co-Advisor: _____

(Dr. Nauman Qureshi)

DEDICATION

To Allah the Almighty

&

To Our Family, Faculty and Friends
Who have always been there for Us

ACKNOWLEDGEMENTS

First of all we are thankful to Almighty Allah for giving us the strength and ability to complete this project and we are extremely thankful to our advisor and co-advisor, Mr. Dr. Peter Charles Bloodsworth, and Mr. Dr. Nauman A.Qureshi respectively for helping us throughout the course in accomplishing our final year project. Their guidance, support and motivation encouraged us in achieving the objectives of the project. It is an honor and pleasure for us to work on this project under their supervision.

Secondly, we acknowledge our parents; their love, prayer and trust in us helped us a lot in completing this task so we are also very thankful to them

Table of Contents

CHAPTER 1	8
1. Introduction	8
1.1. Problem statement	8
1.2. Motivation:	8
1.3. Domain knowledge:	8
1.4. Goals:	9
1.5. Solution	9
1.6. Basic Architecture	9
CHAPTER 2:	10
2. Literature Review	10
2.1. Why multi-agent system?	10
2.2. Why AMAZON EC2?	10
2.3. Research Aspects	11
2.3.1. Data integration	11
2.3.2. Agent-cloud interaction	11
2.3.3. Artificial intelligence (how intelligent our agents are)	11
2.4. Challenges	11
2.5. Business point of view:	12
2.6. Technologies used:.....	12
2.7. Comparison:.....	12
2.8. Comparison of Servers:.....	13
2.9. End product:.....	13
CHAPTER 3:	14
3. Functionality and design	14
3.1. Major Features:.....	14
3.1.1. Paper Matching.....	14
3.1.2. Automatic Downloading	15
3.1.3. Email Alerts	15
3.1.4. Behavior/preference based suggestions:	16
3.1.5. Better relevant results	17
3.2. Agent Architecture.....	18
3.3. Flow Diagram:	20
3.4. Basic Design:	20

3.5. ERD.....	21
CHAPTER 4	22
4. Methodology.....	22
4.1. INTEGRATION.....	23
4.2. Data Agents.....	26
4.2.1. Matching Agent.....	27
4.2.2. Priority Agent:	27
4.2.3. Download agent:.....	28
4.3. Paper Matching:.....	29
4.4. Priority Index:.....	31
4.5. Email Prompts:.....	33
4.6. Publication Insertion:	33
4.7. Automatic Downloads:.....	34
4.8. Data Retrieval:.....	36
4.9. DBLP:	37
4.10. Microsoft Academic Search API:	37
4.11. Google Scholar:	38
4.12. Data Integration Agent:	39
5. Results:.....	40
5.1. GUI Testing:.....	40
6. Conclusion and Recommendations:	41
7. References:	41

Figure 1: Basic Architecture	9
Figure 2: Paper Matching.....	14
Figure 3: Automatic Downloading	15
Figure 4: Email Alerts.....	16
Figure 5: Behavior/preference based suggestions.....	16
Figure 6: Better relevant results	17
Figure 7: Use Case Diagram	19
Figure 8: Flow Diagram.....	20
Figure 9: Basic Design.....	20
Figure 10: Entity Relationship Diagram	21
Figure 11: Home Screen	23
Figure 12: Modules	23
Figure 13: Paper Matching.....	29
Figure 14: Paper Matching.....	29
Figure 16: Automatic Download.....	34
Figure 17: Data Retrieval.....	36
Figure 18: Data Integration Agent	39

1. Introduction

1.1. Problem statement

A problem that researchers usually face is that they do not know much about papers published in the particular conferences of a certain domain similar to what he is thinking of. The conflict arises when a person submits a paper in a conference and then he realizes that he didn't contribute towards a particular domain because a similar work was being done before or during his time of research. There are different solutions available to solve this problem but still most researchers need to work for Months just to decide and ensure that no similar work/ research have been done which will be the basis of his research because the currently available solutions do not focus on the actual needs of a researcher. This project aims to facilitate the research community by making them aware of any conference paper or journal being published in the same domain or of the same topic that they were thinking to work on. The project will provide features like automatic downloads, paper similarity, email alerts and user preference based priority. All along with this we will try to improve the relevance of results by retrieving data from as much online sources as much possible. Major sources include Google, Microsoft, Dblp, Mendeley and Citeseerx.

1.2. Motivation:

The basic motivation behind doing this project is to help out MS and PhD students in writing literature reviews. Most of such people face a lot of problem while writing a literature review and waste a lot of time and energy over it that ends up adding no value. Therefore to help such struggling people, we are developing an intelligent system that helps them by taking their work load to itself and facilitate them in writing and research. We hope that the project will be much beneficial to all.

1.3. Domain knowledge:

The domain of our system falls under the category of artificial intelligence and cloud computing, artificial intelligence because we are using intelligent agents to achieve our targets. Agents are created using JADE in Java and are capable enough to perform tasks on the behalf of others. As we will be using cloud to deploy our agents, it thus also falls under the domain of cloud computing. We will be using amazon EC2 servers to host our agents in the cloud environment. As we can see that cloud computing is the future of computing that is why we have aimed working over it. We are ensuring use of latest technologies.

1.4. Goals:

Our main goal is to develop a system capable enough of helping people writing literature reviews by providing them preference based suggestions, along with this the system will provide strong supporting features like automatic downloads, similarity measure and the priority check. We have decided to develop an interface that is easy to use by a lay man and is attractive and responsive.

1.5. Solution

This project aims to facilitate the research community in Pakistan by making them aware of any conference paper or journal being published in the same domain or of the same topic that they were thinking to work on. The project involves understanding of multi-agent technologies along with usage of cloud resources. Agents will work on the user's behalf to search for relevant information and conference papers from different resources such as Google scholar, the ACM or the IEEE online

1.6. Basic Architecture

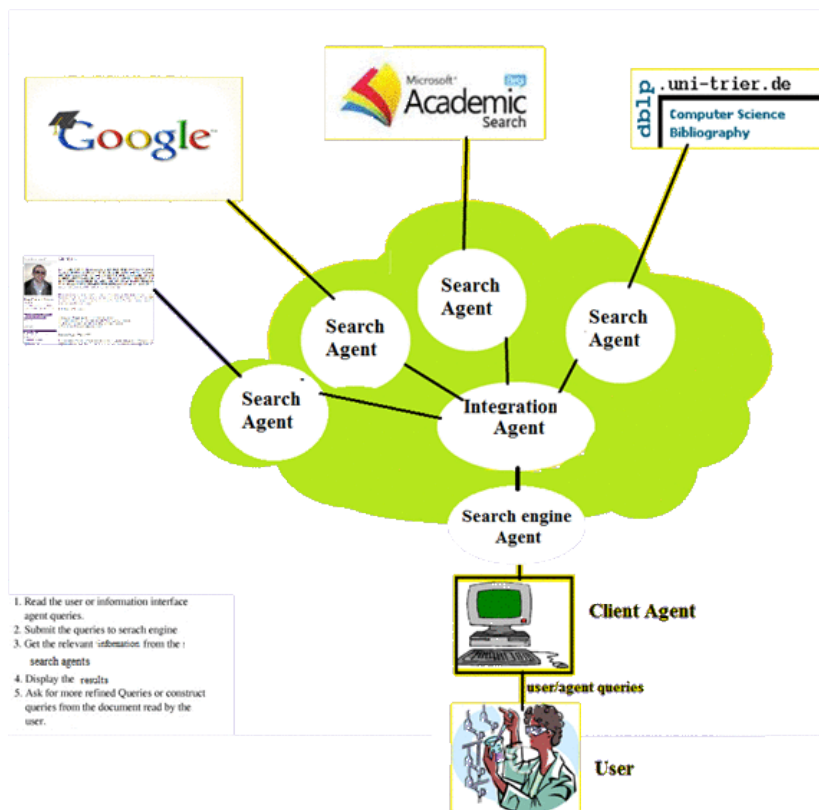


Figure 1: Basic Architecture

2. Literature Review

2.1. Why multi-agent system?

As primary focus of our application is gathering data concurrently from multiple resources, so it would be inevitable to use multi-agents. As the agents work parallel and coordinate with each other unlike individual agent or a monolithic system. Following are the advantages of developing a multi-agent system

- Efficient
- Can make agents intelligent as per our requirement for processing results and making comparisons.

Basically multi-agents form an ant like structure as the ants do their job, same is for the agents in a multi-agent system. They all go out in the environment and fetch back desired results to a common place.

2.2. Why AMAZON EC2?

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity in the cloud. It is designed to make web-scale computing easier for developers. We will be using it because it provides much better facilities than its competitors in much less price and it is more

- Elastic
- Flexible
- Reliable
- Inexpensive (comparatively)

2.3. Research Aspects

2.3.1. Data integration

It involves the development of techniques about integrating data from different sources together to provide users with a unified view of this data, and formatting data in a single proper format after being received from different sources in different formats.

2.3.2. Agent-cloud interaction

As far as agent-cloud interaction is concerned we will be developing scalable multi-agent system which will use cloud resources. As per this point of view, we will work on how an agent will interact with the cloud.

2.3.3. Artificial intelligence (how intelligent our agents are)

We will be working at how intelligent our agents are and at improved searching and comparison algorithms. The algorithms should be efficient enough to search from different sources in a few seconds.

2.4. Challenges

- How to make the system efficient?
- How to get data from different sources?
 - DBLP
 - Personal web pages of researchers
 - Microsoft academic research API
 - Mandalay
 - Cite Seer
- How to manage huge data sets?
- How to compare results from different sources and filtering of required results?

2.5. Business point of view:

- Our direct competitors are Google, Microsoft, and ZETOC etc.
- How can we make our product unique than their products, to get a niche in the market.

2.6. Technologies used:

- JADE (to use AOP)
- Amazon EC2
- JSF (for developing interface)

2.7. Comparison:

Parameter	Google Scholar	Microsoft	Zetoc	Paper Scout
Search relevance	70% relevance* found (21/30)	50% relevance* found(15/30)	Results confined to British Library resources	Will try to optimize it to the best and better than others
Alerts	Basic**	No	Proper alerts	Proper alerts
Interface	Poor	Interactive	Interactive	Interactive
Features	No advanced features	Have some advanced features	Have some advanced features	Will have advanced features***

*relevance found by searching research papers published by Dr.Peter bloodsworth on different sides out of his total research papers.

**Very basic alerts and need to be created manually by the user.

** Features like graphical representations, messages-mails, different search parameters etc.

2.8. Comparison of Servers:

Dedicated Server	Cloud based server
1. A dedicated server is its own collection of hardware.	1. It is basically software that runs on hypervisor. It is not its own collection of hardware.
2. It is a bit difficult to scale up and down a dedicated server as it requires hardware changes.	2. They can be scaled up and down easily using a software interface
3. Back up is a bit difficult in dedicated servers.	3. Back up is easy in cloud.
4. It is not easy to move dedicated servers in data centers as they need to move hardware too.	4. It is easier to move cloud servers in data centers.

2.9. End product:

A fully functional web based academic searching tool possessing interactive and efficient interface and giving better results .Features like automatic alerts, messaging service, E-mails will be implemented. A user will need to sign up before start experiencing.

3. Functionality and design

3.1. Major Features:

3.1.1. Paper Matching

This feature will help users find publications similar to the publication they provide to the application. The application will process the input semantically and finally suggesting some other publications that are semantically related to the one provided by the user.

Working:

The user will provide a PDF or word document as input to the application. The application will convert it in plain text and will trim such that only the first 2 pages remain. After words this plain text will be provided to GATE for natural language processing. GATE will find keywords related to the subject and then the synonyms of the found keywords will be determined. After all this a list of important keywords will be generated along with their semantic relation (depending on the algorithm being used). Then those keywords having maximum similarity will be found over the web and the results will be shown to the user. The steps are explained below.

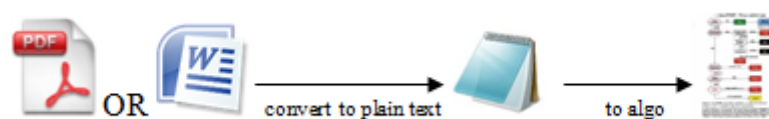


Figure 2: Paper Matching

3.1.2. Automatic Downloading

It will help user to download full text versions of publications automatically by a single click. This feature is designed keeping in view the messy work being done by people searching for the full texts of the publications over the web. Now the user will select the publications of its choice and press download. Our application will look for the full text versions of those publications and will download for the user at its machine. In the end a report will be generated describing the final download status of the publications.

Working:

To use this feature the user will click on a particular publication or may select multiple publications to download, the downloading agent will take the titles of those publications and will search over the web and will download the available publications. It will also generate a report in the end.



Figure 3: Automatic Downloading

3.1.3. Email Alerts

Emails are ubiquitous these days. Our system will be intelligent enough to detect latest happenings and will prompt the users by sending those emails immediately. User will be able to create email alerts and will be informed about that particular alert through emails. For example a person creates an alert for the publications related to 'Data mining', The application will prompt the user immediately if any new publication is published in the field of 'data mining'.

Working:

The user will be alerted if anything changes. An agent will be looking for any changes in the databases, if the agent finds any change or new publication; it will inform the users via an email immediately.



Figure 4: Email Alerts

3.1.4. Behavior/preference based suggestions:

As adaptive systems are a hot topic these days, ours will also be an adaptive system. We will keep record of user's behavior by observing his/her pattern and preferences. After this we will suggest user based upon the preferences and behavior. By this way the user will have a kind of customizable home page, related to stuff related to its preference only, ending in a better user experience.

Working:

Every time a user logs in to our application, we will start observing it behavior and activities by storing the keywords he/she enters and the publications he/she clicks at in the database. An algorithm will be implemented to decide the behavior of that particular user by taking values from the database and comparing them. Then an agent will find data related to the behavior of that particular user. When the user will log in next time, he will be able to see suggestions based on his previous visits.



Figure 5: Behavior/preference based suggestions

3.1.5. Better relevant results

Our application will allow users to search freely about any publication, subject or author. The main issue that different tools face is the relevance of results. We will try to improve the relevant of results by using intelligent algorithms.

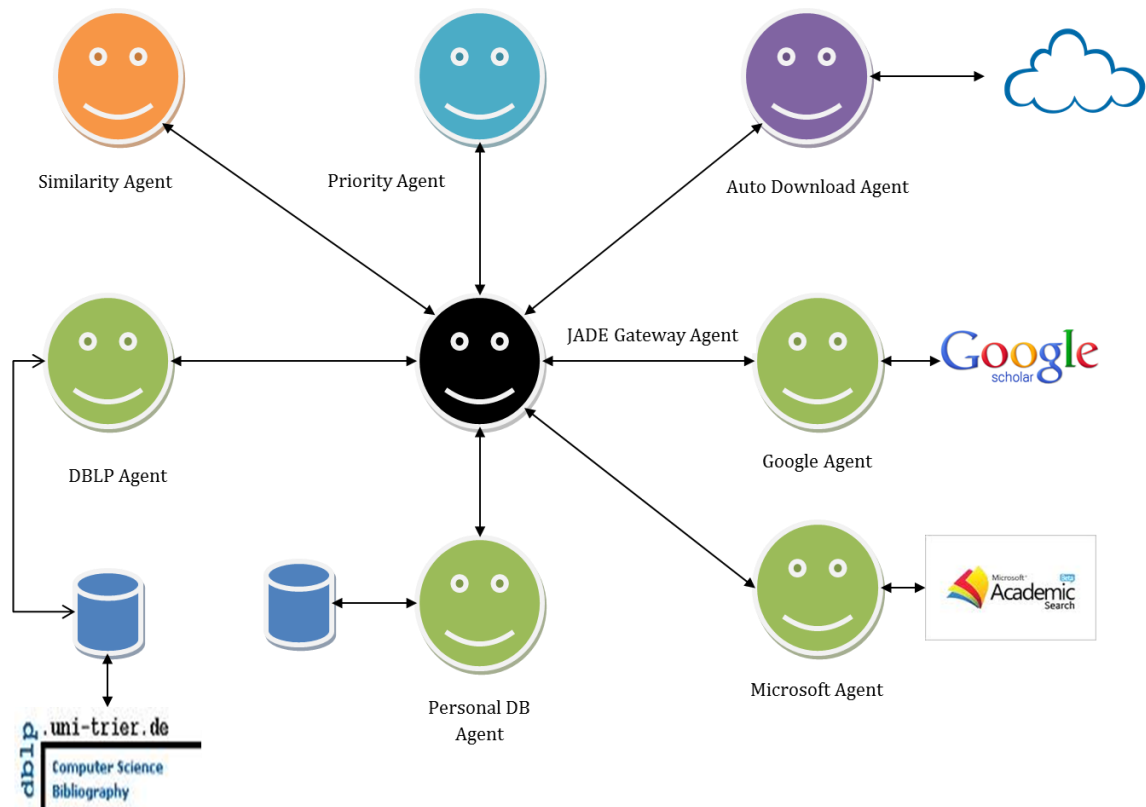
Working:

To improve the relevance and to cover most of the results we will take data from as many online sources as possible. In the end we will integrate the data thus removing data redundancy. And finally the results will be shown to the user.



Figure 6: Better relevant results

3.2. Agent Architecture



- This is the basic architecture used to retrieve results from different sources and show the results to the user. The user will enter a query and the query will be passed on to the 3 data retrieval agents namely “DBLP”, “MAS” and “home page search” agent. The data retrieval agents will pass on the results in a similar format to “data integration agent”. The Data integration agent will prepare a final list and display it to the client. The duties of the agents are as follows:

- **Sign In Agent:** Will help user sign in its account. It will check in the database if the user exists.
- **Sign up Agent:** will insert the details of the user when he/she signs up.
- **Microsoft Academic Search Agent:** Will retrieve results from Microsoft academic research API.
- **DBLP Agent:** Will retrieve results from DBLP API
- **Home Page search agent:** Will retrieve results from the home pages of the professors/researchers.

- **Data Integration agent:** Will remove data redundancy and will integrate data from the 3 data retrieval agents and it will then display the final list to the user.
- **Download & Search Agent:** This one will be responsible for searching will pdf files of research papers and download them to the client machine.
- **Display Agent:** Will be responsible to display data lists with sorting and refining features. It will communicate with download and search agent to download files and show the reports about the downloaded PDFs.
- **Informing Agent:** Will inform the email agent about any latest update in the online databases.
- **Email Agent:** Will send email to the user about any latest update.
- **Paper Matching Agent:** This agent will find similar publications to the one uploaded by the user.

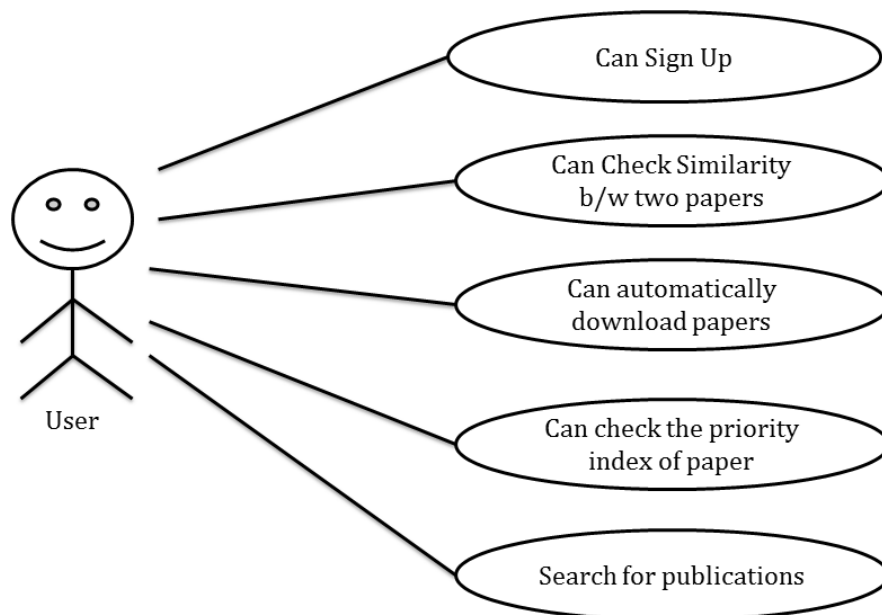


Figure 7: Use Case Diagram

3.3. Flow Diagram:

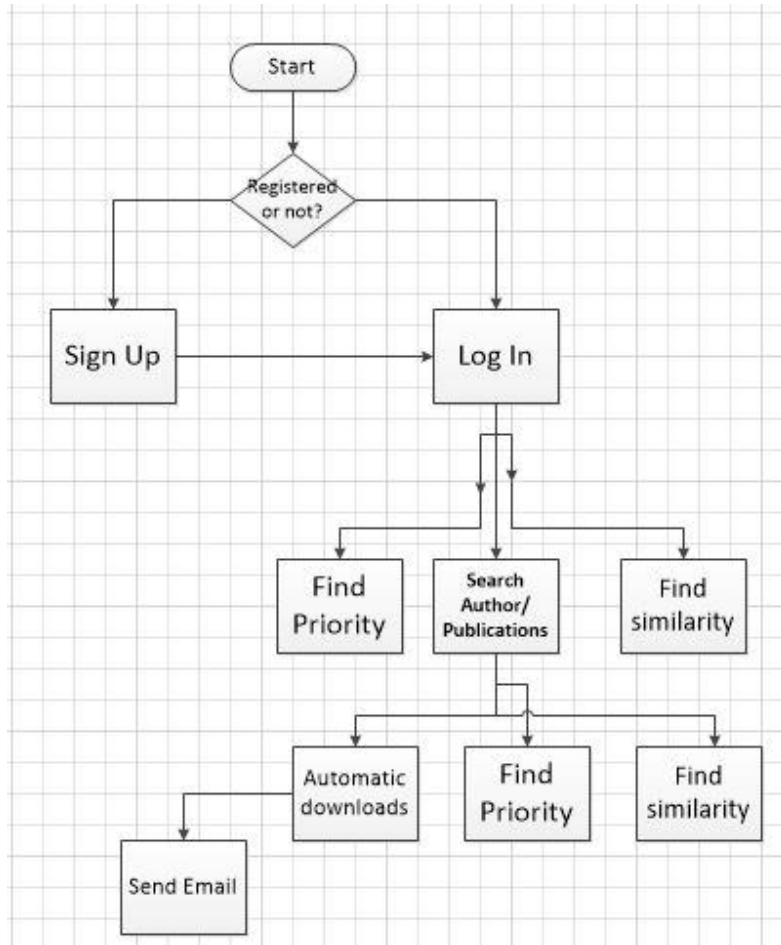


Figure 8: Flow Diagram

3.4. Basic Design:

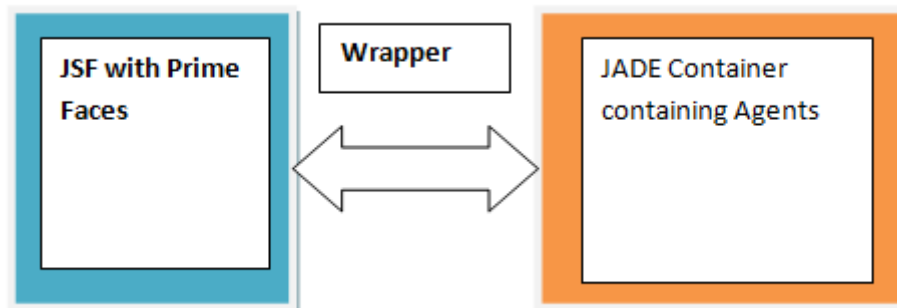


Figure 9: Basic Design

3.5. ERD

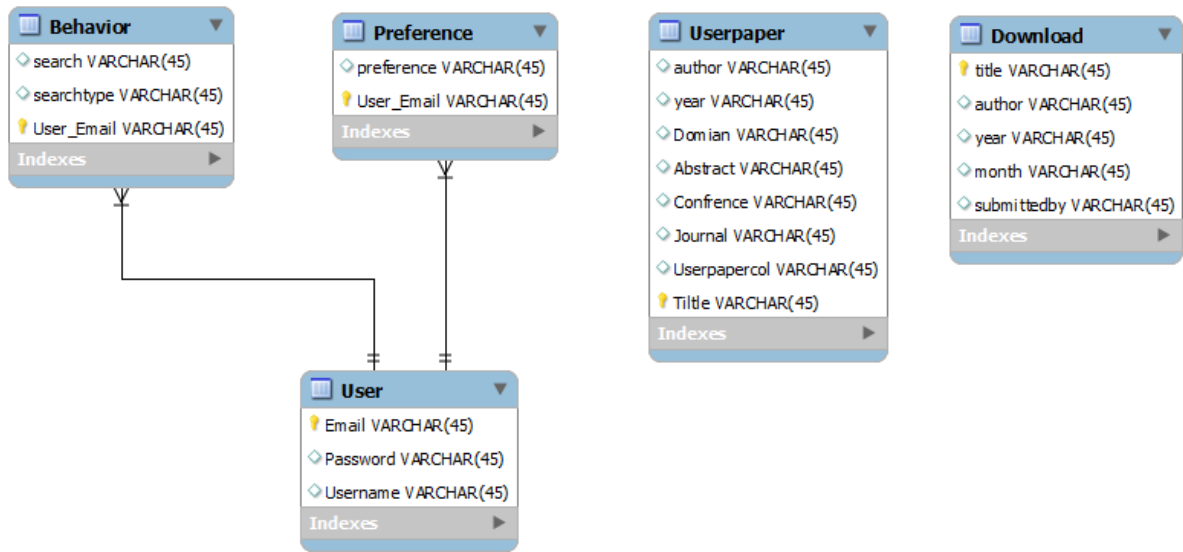


Figure 10: Entity Relationship Diagram

4. Methodology

In this part we will explain how our system works and how different modules interact with each other. Basically we have used jade to develop agents in java. The front end is developed in JSF and we have used prime faces to add extra capabilities to it. We will explain the functionality of every module separately.

JSF front end using Prime faces+ Integration with Agent O2A communication (jade gateway wrapper class)

Front end of this system is made by using various features of Prime faces; some of main features of prime faces that are used are as follows

- Charts
- Data table
- Dialog box
- File Chooser

User gets validated then allowed access to the main features of our system and he can get personalized results.

Different Pages are made to serve different features using bean at back end we had to configure our project with prime faces and to use some feature like file upload etc. we had to include various other jar files

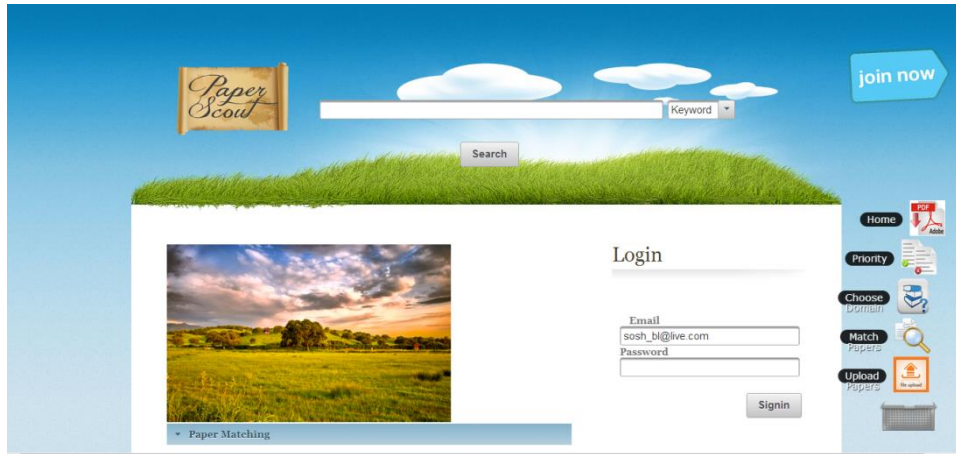


Figure 11: Home Screen

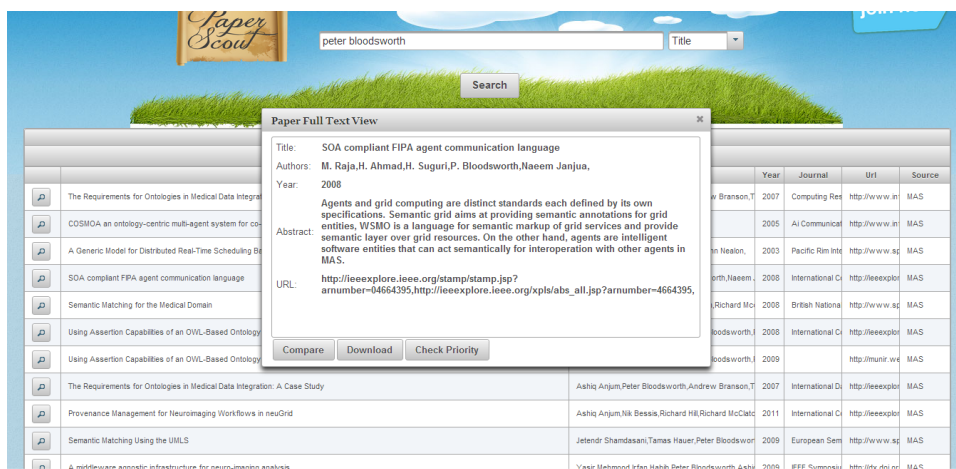


Figure 12: Modules

4.1. INTEGRATION

Integration between Agents and JSF pages is done via alteration in Jade Gateway Wrapper class. (Wrote it for JSF and tackle various Agents with different responses) which is based on O2A communication.

What we actually do is to put our message and receiver name in and object and then get these parameters from object and then send these to desired agent via jade gateway which has ability to extract parameter from object and act like an agent (O2A communication). In particular case this agent receives the blackboard object and its content will be sent to the proper agent

```

public class MyGateWayAgent extends GatewayAgent {
    BlackBoardBean board=null ;
        String ans;
    protected void processCommand(java.lang.Object obj) {
        if (obj instanceof BlackBoardBean) {
            countx=0;
            board = (BlackBoardBean)obj;
            ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
            msg.addReceiver(new AID( board.getReceiver(), AID.ISLOCALNAME) );
            if(board.getCommand().equalsIgnoreCase("searched"))
                {
                    DbIp.perInfoAll.clear();
                //msg.addReceiver(new AID( board.getReceiver1(), AID.ISLOCALNAME) );
                msg.addReceiver(new AID( board.getReceiver2(), AID.ISLOCALNAME) );
                }
            count1=0;
            msg.setContent(board.getMessage());
                send(msg);
            if(board.getCommand().equalsIgnoreCase("download"))
                {   releaseCommand(board);   }
            }
        }
    public void setup()
    {   // Waiting for the answer
        addBehaviour(new CyclicBehaviour(this)
            {   public void action() {
                ACLMessage msg = receive();
                if ( (msg!=null)&&(board!=null) )
                {   board.setMessage(msg.getContent());
                    if(msg.getContent().equals("done"))
                        {
                            countx++;
                        }
                    else
                        {
                            ans=msg.getContent();

                            String[] parts = ans.split("%#");
                            if (parts[1].equals("hello"))
                                {

```



```

        Dblp.result=parts[2];
        Dblp.key1=parts[3];
        Dblp.key2=parts[4];
        Dblp.key3=parts[5];
        Dblp.key4=parts[6];
        Dblp.key5=parts[7];
        releaseCommand(board);
    }
    else
    {
        Dblp.perInfo joe=new Dblp.perInfo(parts[0], parts[1] , parts[2], parts[3], parts[4],parts[5],parts[6]);

        Dblp.perInfoAll.add(joe);
        //Dblp.perInfoAll[count1]=new Dblp.perInfo(parts[0], parts[1] , parts[2], parts[3],
parts[4]);
        count1++;
    }
    }
    if(countx>=2)
    {
        releaseCommand(board);
    }
    } else block();
    }
    });
    super.setup();
}
}
}

```

4.2. Data Agents

Above is our Gateway Agent , it does communicate with various agent like priority ,matching, data integration and download agent and do expect a response from these agents which may be of different structure we do recognize there kind by introducing a key

We obtain data from different agents like DBLP, mas personal database and scholar and they obtain data from respective sources then it is send to our gateway agent which checks that if done keyword is received from all these agents then release command is issued and data is shown to the user .

All data is stored into custom list create from PerInfo Class

```
DbIp.perInfo joe=new DbIp.perInfo(parts[0], parts[1] , parts[2], parts[3], parts[4],parts[5],parts[6]);
```

4.2.1. Matching Agent

We do send

- Two paths
- Two online papers
- One online and one path of uploaded file

To matching agent which then in response gives us the similarity in percentage and most important key words when we receive these key words release command is issued

```
String[] parts = ans.split("%#");  
  
if (parts[1].equals("hello"))  
{  
  
Dblp.result=parts[2];  
  
Dblp.key1=parts[3];  
  
Dblp.key2=parts[4];  
  
Dblp.key3=parts[5];  
  
Dblp.key4=parts[6];  
  
Dblp.key5=parts[7];  
  
releaseCommand(board);  
  
}
```

4.2.2. Priority Agent:

Similar to matching agent path/ online paper along with selected domain and keywords is sent to user and then we get results in form of percentage and remarks and keywords that promoted this result

4.2.3. Download agent:

Online paper is sent to download agent and no response is expected from it and release command is issued as soon as online paper message is sent.

Code for setting receivers and content to be sent is as below

```
public String searchagent(){  
  
    // Dblp.perInfoAll=null;  
  
    JadeGateway.init("jadebean.MyGateWayAgent",null);  
  
    board = new BlackBoardBean();  
  
    board.setCommand("searched");  
  
board.setReceiver("db");  
  
board.setReceiver1("ps");  
  
board.setReceiver2("mas");  
  
//board.setReceiver3("mendley");  
  
board.setMessage(""+Dblp.search+"%#" +PPRBean.city);  
  
    try    {  
  
        JadeGateway.execute(board);  
  
    } catch(Exception e) { e.printStackTrace(); }  
  
    //return "response?faces-redirect=true";  
  
    return "results?faces-redirect=true";  
  
}
```

4.3. Paper Matching:

For the module of paper matching we have provided three options to the user, such as

- Check similarity between two documents
- Check similarity between one uploaded and one online selected.
- Both of the publications are online selected from the list displayed.

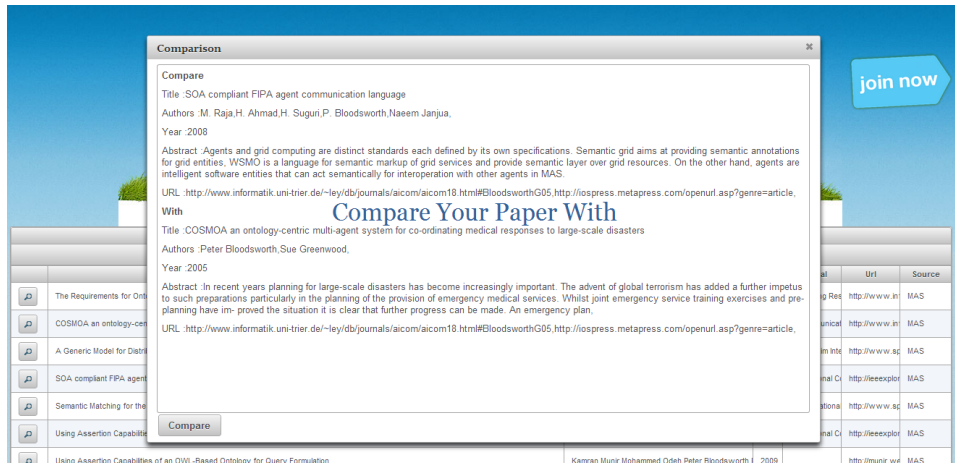


Figure 13: Paper Matching

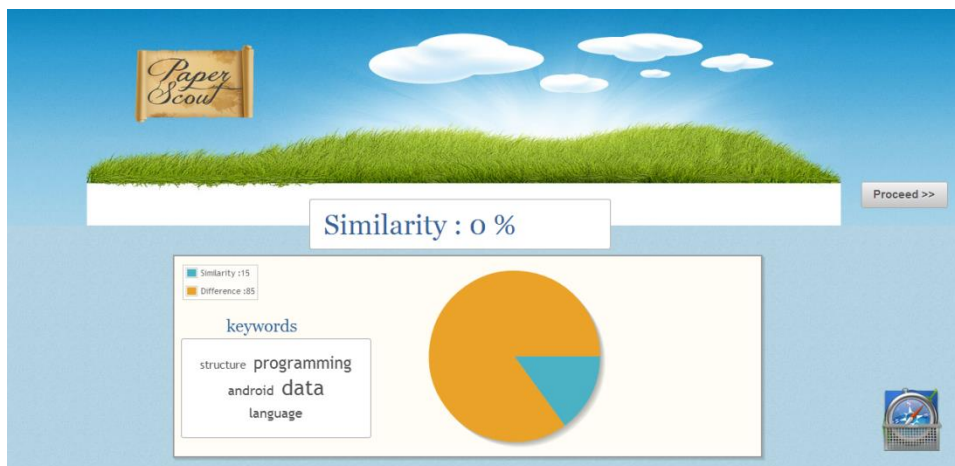


Figure 14: Paper Matching

To provide this functionality we have devised an algorithm. The algorithm works as follows:

- 1: Takes input as PDF.
- 2: Changes to plain text.
- 3: Trim the document to one page only. We take the first page only as it contains title, authors and abstract.
- 3: Removes all the 3 and less than 3 letter words from the text.
- 4: finds the cosine similarity of the document with the compared one.
- 5: The result is then multiplied with 100 and is sent back to the integration agent.

Explanation:

Now we will explain the above mentioned steps. Firstly the rich text format (in the case of document upload) is converted to plain text, we are using PDFBox by apache foundation to convert and trim pdfs. After converting into plain text we execute an algorithm to remove the 3 and less than 3 letter words, after words the two plain texts are compared to find cosine similarity. Cosine similarity is basically found mathematically and it tells us the occurrence of each word in the two documents relative to each other. After words the occurrence frequency is put in a vector and in the last calculated for the cosine index. The mathematical formula to calculate the cosine similarity is as follows.

$$\text{COS} = \frac{A \cdot B}{\text{mod}(A) \times \text{mod}(B)}$$

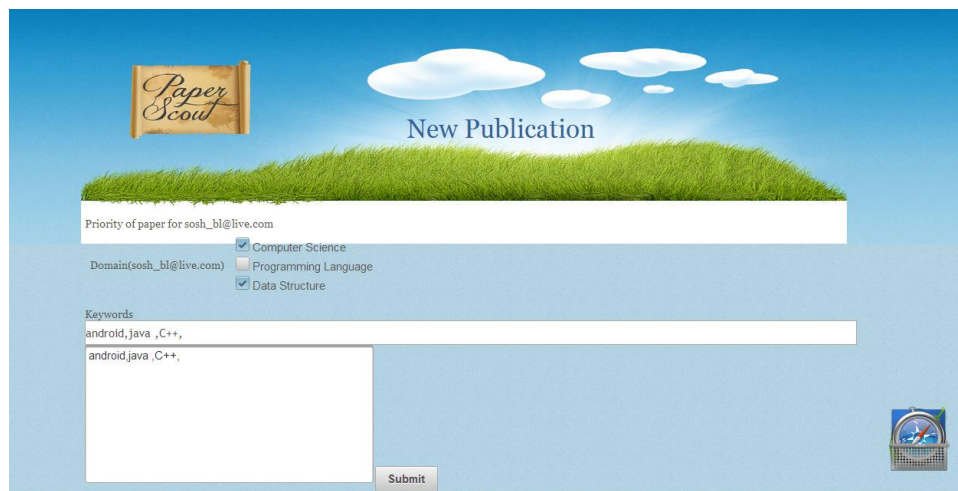
Where “.” Is the dot product of two vectors. The returned value is a numeric figure showing results in %age. The Agent message type used is “INFORM”.

4.4. Priority Index:

The primary focus of this feature is to predict priority of a particular publication w.r.t a person's own interest. For example a person is eager about cloud computing and he wants to check the priority of a particular paper. He will be told this according to cloud computing.

For this feature the user has two options:

- User can upload a document to check the priority index.
- User can select a paper from the results displayed



The screenshot shows a web application interface titled "Paper Scout" with a "New Publication" section. The background features a blue sky with white clouds and a green grassy hill. The form includes the following fields and options:

- A text input field for "Priority of paper for sosh_bi@live.com".
- A "Domain(sosh_bi@live.com)" label.
- Three checkboxes for domain categories: "Computer Science" (checked), "Programming Language" (unchecked), and "Data Structure" (checked).
- A "Keywords" section with a text input field containing "android,java ,C++," and a list box below it containing "android.java ,C++,".
- A "Submit" button at the bottom right.
- A small globe icon in the bottom right corner.

Following steps are involved in calculating the priority index.

- 1: Takes input as PDF.
- 2: Changes to plain text.
- 3: Trim the document to one page only. We take the first page only as it contains title, authors and abstract.
- 3: Removes all the 3 and less than 3 letter words from the text.
- 4: finds the cosine similarity of the document with the compared one.
- 5: The result is then multiplied with 100 and is sent back to the integration agent.

6: The final result is then graded as per our own scheme; the scheme is explained as below

- 90% and greater means “VERY HIGH”
- Between 80% and 90% means “HIGH”
- Between 65% and 80% means “Upper Medium”
- Between 50% and 65% means “Medium”
- Between 40% and 50% means “LOW”
- Between 25% and 40% means “Very Low”
- 25% and less means “considerably Low”

Explanation:

Now we will explain the above mentioned steps. Firstly the rich text format (in the case of document upload) is converted to plain text, we are using PDFBox by apache foundation to convert and trim pdfs. After converting into plain text we execute an algorithm to remove the 3 and less than 3 letter words, after words the two plain texts are compared to find cosine similarity. Cosine similarity is basically found mathematically and it tells us the occurrence of each word in the two documents relative to each other. After words the occurrence frequency is put in a vector and in the last calculated for the cosine index. The mathematical formula to calculate the cosine similarity is as follows.

$$\text{COS} = \frac{A \cdot B}{\text{mod}(A) \times \text{mod}(B)}$$

After words the results are mapped according to our grading criteria and are sent back to the integration agent. The Agent message type used is “INFORM”.

4.5. Email Prompts:

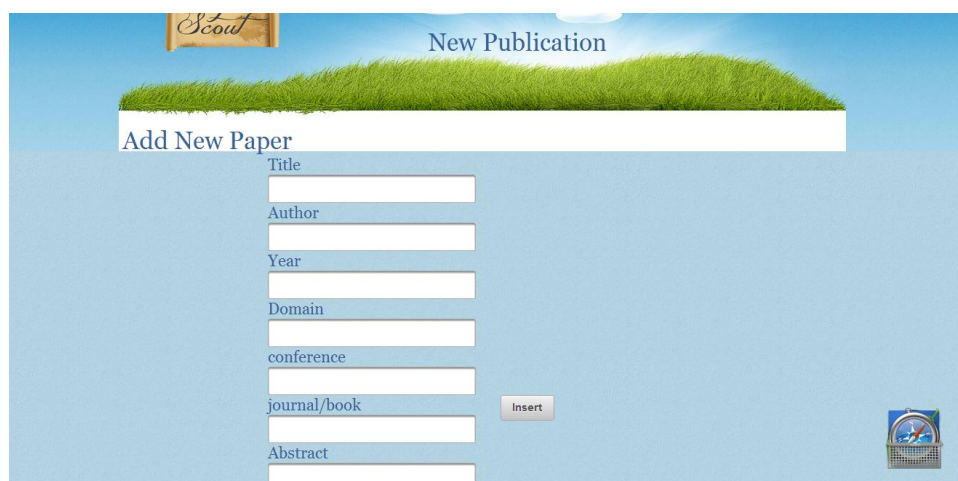
The system provides a feature of email prompts if any new publication is added to our database. The system runs an analysis and matches the publication domain along with the people signed up with us of the same domain. Afterwards all those are prompted about the latest publication.

For example a person signs up with the domain interest of artificial intelligence. After a few days a new paper related to artificial intelligence is added to our system. All people having the domain interest of artificial intelligence will be prompted.

In near future email prompts can be used to detect latest publications added to the external databases such as DBLP Microsoft etc. Currently this feature prompts only when a publication is added to our local database only.

4.6. Publication Insertion:

This feature helps the user to add publications to our local database. Our aim is to create our local database with loads of data so that we do not need to depend on external data in future. The data insertion form is created keeping in view all the necessary fields required related to a publication.



The screenshot shows a web interface for adding a new publication. At the top, there is a logo with the word "Scout" and the text "New Publication". Below this is a decorative header with a green grassy hill. The main form is titled "Add New Paper" and contains the following fields:

- Title
- Author
- Year
- Domain
- conference
- journal/book
- Abstract

An "Insert" button is located to the right of the form. A small icon of a globe is visible in the bottom right corner of the form area.

4.7. Automatic Downloads:

This feature helps to download full text research papers automatically from the web. The feature searches for the full text versions and downloads them. After downloading validates the paper and then sends the paper via email to the person. The feature algorithm works in the following way

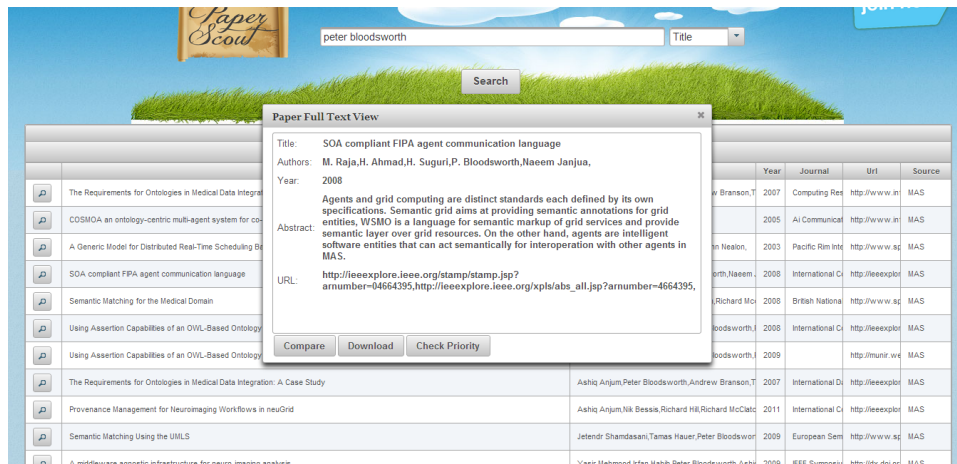


Figure 15: Automatic Download

- 1: String received to the download agent.
- 2: Parses the string and tries to download the full text version from the URL.
- 3: If successful in downloading the full text version. It sends the paper to the email id as per the session maintained at that time.
- 4: If the full text is not found it provides the title of the paper to Google custom search API. The API then finds it over the web and returns the response in an XML format.
- 5: The result is parsed and the URLs are saved in a list object.
- 6: Then the agent checks if a document is available at the URL or not. If yes it is downloaded and if no, the agent tries to connect with the next document.

7: Once the agent finds a document and validates it, if the validation returns true the full text of that paper is attached and mailed to the desired person.

8: If the validation returns false, the agent leaves it and tries to connect to the next URL in the list.

9: If the agent is unsuccessful in finding any paper or validating it, it mails the URLs of the publication to the desired person.

Explanation:

This agent is very dynamic in contrast with the other ones. It helps to download full text of publications. The agent gets its first response in XML from Google custom search API.

The XML to java conversion is being done using the SAX API. After parsing the results, they are being searched and the whole process as mentioned above is processed. For searching the web we have tried yahoo, Google, Bing and ASK search engines but the best suited was Google. That is why we preferred using Google. The results were then calculated and shown. The emails were sent via the TLS protocol.

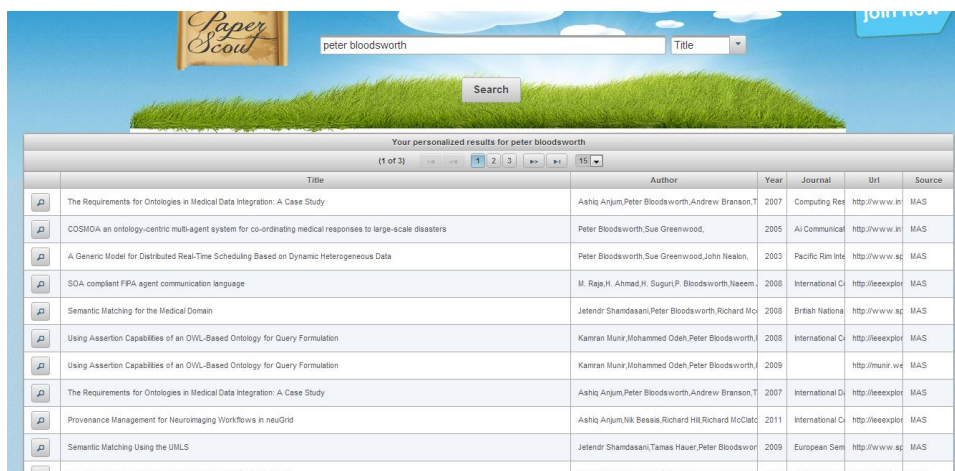
For the validation of a research paper, we check by comparing the title and author names. The system checks if particular author name and title name exists in the file then it returns true otherwise false. Such functionality provides us with 2 security checks and this method of validation is quite effective.

4.8. Data Retrieval:

We have retrieved data from different online academic databases and web services available to retrieve data. We retrieved the data from the following sources

- DBLP
- Microsoft Academic search
- Google Scholar
- Mendeley
- CiteSeerX
- Our own database

Now I will briefly explain the data retrieval from each source. We created individual agents for every source.



The screenshot shows a search interface with a search bar containing 'peter bloodsworth' and a 'Search' button. Below the search bar, there is a table of search results. The table has columns for Title, Author, Year, Journal, Url, and Source. The results are as follows:

Title	Author	Year	Journal	Url	Source
The Requirements for Ontologies in Medical Data Integration: A Case Study	Ashiq Anjum,Peter Bloodsworth,Andrew Branson,T	2007	Computing Res	http://www.in	MAS
COSMOA: an ontology-centric multi-agent system for co-ordinating medical responses to large-scale disasters	Peter Bloodsworth,Sue Greenwood,	2005	AI Communica	http://www.in	MAS
A Generic Model for Distributed Real-Time Scheduling Based on Dynamic Heterogeneous Data	Peter Bloodsworth,Sue Greenwood,John Neaton,	2003	Pacific Rim Int	http://www.sj	MAS
SOA compliant FPA agent communication language	M. Raja,H. Ahmad,H. Suguri,P. Bloodsworth,Naseem,	2008	International C	http://eeexpl	MAS
Semantic Matching for the Medical Domain	Jetendr. Shandassani,Peter Bloodsworth,Richard Mc	2008	British Nationa	http://www.sj	MAS
Using Assertion Capabilities of an OWL-Based Ontology for Query Formulation	Kamran Munir,Mohammed Odeh,Peter Bloodsworth,	2008	International C	http://eeexpl	MAS
Using Assertion Capabilities of an OWL-Based Ontology for Query Formulation	Kamran Munir,Mohammed Odeh,Peter Bloodsworth,	2009		http://munir.w	MAS
The Requirements for Ontologies in Medical Data Integration: A Case Study	Ashiq Anjum,Peter Bloodsworth,Andrew Branson,T	2007	International D	http://eeexpl	MAS
Provenance Management for Neuroimaging Workflows in neuGrid	Ashiq Anjum,NR. Bessia,Richard Hill,Richard McClatc	2011	International C	http://eeexpl	MAS
Semantic Matching Using the UMIS	Jetendr. Shandassani,Tamas Hauer,Peter Bloodswor	2009	European Sem	http://www.sj	MAS
A Middleware based Infrastructure for Semantic Image Analysis	Yusef Habboubi,Felix Habbib,Dalal Bloodsworth,Ashiq	2006	IEEE Systems	http://www.in	MAS

Figure 16: Data Retrieval

4.9. DBLP:

DBLP is a free service made available by a German university. It provides its data in two ways. The first one is via a web service and the second one is by a simple 1.1GB XML file containing millions of entries in it. We used the second way to retrieve data from DBLP. We downloaded the xml file from their website and inserted all the data in MySQL database. A grand total of 2.1 Million rows were inserted in the database. The attributes that DBLP provide are title, authors, journal, year and URL.

To read the DBLP data in java we used the SAX API streamer to read xml tags. After streaming through the xml file the system then inserted all the data in a database using a SQL query. The data is retrieved using JDBC. The result set is provided as per the query. The DBLP agent has cyclic behavior. And use the INFORM message to send and receive.

4.10. Microsoft Academic Search API:

This is one of the latest APIS to retrieve academic data. The API is provided by Microsoft and it provides its data using a web services. All the functions and the codes of the web service are provided in the API manual. One bad thing about using this API is that it allows you to get only 100 results per query. To retrieve data from Microsoft you first need to register with them by sending them a mail out. Then they reply you with an API key that you need to use while querying the web service. The Microsoft agent uses cyclic behavior and “INFORM” type ACL messages to communicate with the other system. The response from Microsoft is received in the JSON format. We convert the JSON to XML using Staxon API. Then we read the XML file using SAX parser. The results are retrieved and then sent to the integration agent. Microsoft provides a very detailed response. We have taken only those attributes that are important for us to show.

4.11. Google Scholar:

The best academic search engine up till now, having most data covered in it. It is not possible to retrieve data from Google scholar directly but yes there are indirect methods of getting data from Google scholar. One of the methods devised by us is as follows. We developed a separate agent for Google and the agent used “INFORM” messages to respond and has cyclic behavior.

Google scholar agent follows following steps

- Takes input from the front agent
- Parses the input
- Searches it over the web using Google custom search API(the search engine developed uses only scholar.google.com)
- Gets the response in XML and parses it.
- Connects with the profile URL parsed from the previous response
- Receives the response which contains list of publications, this response is in HTML
- Parses the HTML response using JSOUP html parser for java.
- Connects with every single URL to get details about every single publication.
- Sends the response to other agent.

4.12. Data Integration Agent:

The data integration agent works by combining results from all the data agents. The results from all the agents are gathered by data integration agent and the data redundancy is removed from the data. After words it is displayed in the form of a table. The agent has cyclic behavior and it waits to receive 3 “HELLO” strings for every single request. When it receives 3 “HELLO” strings it indicates that the response is completed.

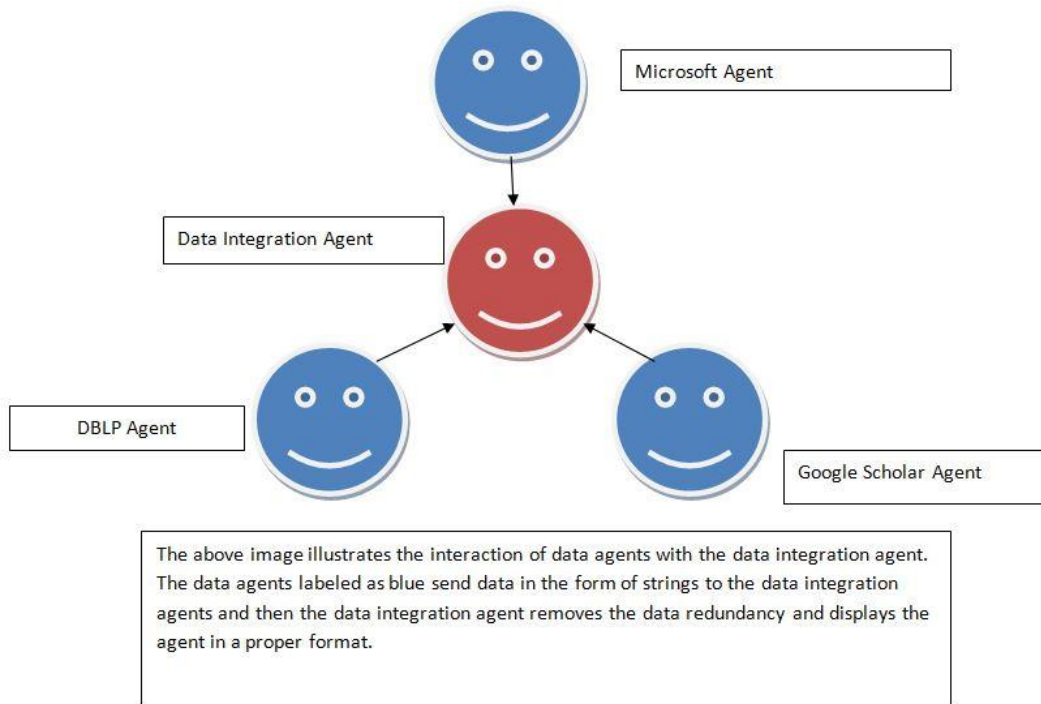


Figure 17: Data Integration Agent

5. Results:

As far as the results of our project are concerned, we completed all the claimed features. All of the features are working fine. Though we could not achieve the desired level of result relevance but still it is very good. All other features cannot be measured qualitatively but they can be measured quantitatively. The features are working well in an integrated environment. All the modules have been integrated and are performing very well.

In short according to us our project is successful, as we have accomplished our goals. Though we are working at the interface to make it more attractive and user friendly. All along with this we are trying to improve the efficiency of the system and the response time to the best.

The system performs well. Some of the tests that we executed and got result are as follows.

We tried to search publications related to Dr.Peter Charles Bloodsworth and we were able to get 80% of his publications.

After words we tested our Automatic downloading feature. The feature worked perfectly well and was able to download 60-70% of the publications. In the other part a mail regarding the URL were sent to the respective user. We have tried to provide support to our customers by providing solution in every case.

For the testing of Similarity agent we tried all the three cases of this function. All of the three cases worked perfectly fine. The result comes out in percentage and as per our analysis the results are very good. The keywords detected by the algorithm make sense and quantitative and qualitative analyses show that the results are accurate. Same results were achieved for the priority module.

The overall performance of the system is quite good and all the features are working fine.

5.1. GUI Testing:

The GUI is tested after integrating all the independent modules. The user was able to perform all the functions and see response related to the query. The components did not crash at any place and all went very well. The testing proved that the GUI is user friendly and easy to use.

6. Conclusion and Recommendations:

PaperScout is working well. All of the features are performing well under all kind of inputs. The accuracy of the system is also very good and depends on person to person use. It is an online tool that will help MS and PhD students in writing their literature reviews before they start their thesis. The tool basically helps to automate the most time consuming activities for writing the literature review. The tool helps to solve the most faced problems by a researcher.

The GUI is user friendly and lets the user add his details easily. It also helps to upload pdf documents. The GUI has been created; keeping in view the basic principles of HCI. The user is prompted of all the responses as soon as possible. The results are shown with proper graphical scales. To guide people more, about the accurate results calculated by our system. Keywords are shown along the results. The keywords are calculated on runtime.

Several improvements can be made to the tool and the results can also be improved. The auto prompt module of the tool can be improved by adding a feature of detecting latest publications at the online databases. Secondly the user can be allowed to choose sources of its choice to see data from.

Another advancement that can be made to the system is of making algorithms of priority index and paper similarity better and more efficient. By improving the algorithms, results obtained will be better accurate and more precise.

The speed of data retrieval can also be improved by improving the data retrieval algorithm.

7. References:

- JADE
 - <http://jade.tilab.com/>
- Prime Faces
 - <http://primefaces.org/>
- ZETOC
 - <http://zetoc.mimas.ac.uk/>
- Microsoft Academic Research

- <http://academic.research.microsoft.com/>
- Google Scholar
 - <http://scholar.google.com.pk/>

Turnitin Originality Report

Document Viewer

- Processed on: 22-May-2013 17:46 PKT
- ID: 331924961
- Word Count: 6059
- Submitted: 1

PaperScout By Osama Misbah

Similarity Index		Similarity by Source	
4%		Internet Sources:	3%
		Publications:	2%
		Student Papers:	3%

1% match (Internet from 29-Feb-2008)

<http://sharon.cselt.it/projects/jade/doc/tutorials/JadeGateway.pdf>

1% match (student papers from 27-Jun-2012)

[Submitted to Higher Education Commission Pakistan on 2012-06-27](#)

1% match (student papers from 27-Jun-2012)

[Submitted to Higher Education Commission Pakistan on 2012-06-27](#)

< 1% match (Internet from 18-May-2009)

http://www.univaud.com/about/news/press_2008/12022008.php

< 1% match (student papers from 25-Oct-2010)

[Submitted to School of Accounting & Management on 2010-10-25](#)

< 1% match (publications)

[Ni Lar Thein. "Ontology-Based Agent Community for Information Integration System in Semantic Web". First Asia International Conference on Modelling & Simulation \(AMS 07\). 03/2007](#)

< 1% match (student papers from 05-Apr-2012)

[Submitted to The Hong Kong Polytechnic University on 2012-04-05](#)

< 1% match (Internet from 11-Sep-2010)

<http://elm.eeng.dcu.ie/~nauphys/projects/MichaelSearles-Thesis.pdf>

< 1% match ()

<http://lists.horde.org/archives/chora.mbox/chora.mbox>

< 1% match (publications)

[Xiao-jun, He and Zhen-ying, Chen. "Excellent editors need to be good authors too". Learned Publishing. 2013.](#)

< 1% match (Internet from 30-Dec-2012)

<http://www.codefeeds.com/>

< 1% match (Internet from 22-Jun-2010)

<http://ta.ba.ttu.edu/naggarwal/Research/Thesis%20Proposal%20Single%20Spacing.pdf>

< 1% match (publications)

[Bellifemine. "The JADE Web Services Integration Gateway". Wiley Series in Agent Technology. 03/03/2007.](#)