DBLP

Project Report

Summary Of the Project

We have 5 java classes Created.

1. GUI.java

2.Author.java

3.Publication.java

4.Main.java

5.EntityResolution.java

EntityResolution.java

For Parsing we used SAX Parser. We created a SAX Parsing Handler Factory interface which creates a SAX Parser. It basically creates the logic of parsing/ parsing methods using the default handler. This implements a MakeSAXParser function where we read the file and parse it according to the required Query.

```
@Override
public void makeSAXParser()
{
    try
    {
        File inputFile = new File("dblp.xml");
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        saxParser.getXMLReader().setFeature("http://xml.org/sax/features/validation", true);
        saxParser.parse(inputFile, handler);
    }
    catch (FileNotFoundException e) {e.printStackTrace();}
    catch (Exception e) {e.printStackTrace(); }
}
```

Parsing is being done in 4 ways :

Parse WWW:

```
    public UserHandlerWWW(Map<String, Author> givenMap){
     public void startElement(String uri,String localname,String qname,Attributes attributes) throws SAXException{

     public void characters(char ch[],int start, int length)throws SAXException{

     public void endElement(String uri,String localname,String qname) throws SAXException {
 }
 //! Handler class for Authors and Publications
```

The above snippet shows the UserHandler created for authors. Here, we create a map as shown:
Now the XML Parser calls back for the StartElement function where we check the start Variable and mark the corresponding boolean value. Next the characters function follows which we have a ch[] , character array storing all the text within opening and closing tag in the given file. Also, Match the author with its corresponding names and create the author instance. After this the SAXParser calls back for the endElement function where we are storing the Author created in the map.

```
public void parseWWW()
{
    DefaultHandler handler = new UserHandlerWWW(this.ourMainMap);
    SAXHandlerFactory factory = new SAXHandlerFactory(handler);
    factory.makeSAXParser();
}
```

This parsing is performed while preprocessing/reading the XML file. Here, we do parsing only on <WWW> (Person Records), and retrieve all the author names of the same Author and store them in the form of a HashMap as < String(the particular author name), Author(all names of the same author) >.

Parse All Authors:

```
    //!Constructor
    public UserHandlerAuthors(Map<String, Author> givenMap) {

    public void startElement(String uri,String localname,String qname,Attributes attributes) throws SAXException{

    public void characters(char ch[],int start, int length)throws SAXException{

     public void endElement(String uri,String localname,String qname) throws SAXException {
 }
```

The above snippet shows the UserHandler created for authors. Here, the XML Parser calls back for the StartElement function where we check the start Variable and mark the corresponding boolean value. Next the characters function follows which we have a ch[] , character array storing all the text within opening and closing tag in the given file. Also, Match the author of the publication with the query author and create the author instance. After this the SAXParser calls back for the endElement function where we are storing the Author created in the map.

We add all the publications of the author Names and Group them incase they are the names of the same Author. Now we map Author Name to the Author Object which contains the Author Names and his number Of Publications.

```
public void parseAllAuthors()
{
    DefaultHandler handler = new UserHandlerAuthors(this.ourMainMap);
    SAXHandlerFactory factory = new SAXHandlerFactory(handler);
    factory.makeSAXParser();
}
```

Parse Publications By Author Name:

This is parsing for Publications pertaining to the Given author. This parsing is done at the time of the Query, when we need to print Publications by AuthorName.

Here, the XML Parser calls back for the StartElement function where we check the start Variable and mark the corresponding boolean value. Next the characters function follows which we have a ch[] , character array storing all the text within opening and closing tag in the given file. Also, Match the author of the publication with the query author and create the publication instance. After this the SAXParser calls back for the endElement function where we are storing the Author created in the map.

```
public List<Publication> parsePublicationByAuthor(String queryAuthorName)
{
    List<Publication> publicationsFound = new ArrayList<Publication>();
    Author queryAuthor = this.ourMainMap.get(queryAuthorName);
    DefaultHandler handler = new UserHandlerAuthorPublication(queryAuthor, publicationsFound);
    SAXHandlerFactory factory = new SAXHandlerFactory(handler);
    factory.makeSAXParser();
    return publicationsFound;
}
```

Parse Publications By Title Tags:

This is also done at the time of the Query, when we are asked to print Publications by TitleName. In this we are taking a List of TitleTags as the parameter and returning the TreeMap of numberOf matches to the list of publications with that number of matches.

```
public Map<Integer, List<Publication> > parsePublicationByTitle(List<String> listTitle)
{
    Map<Integer, List<Publication> > publMap = new TreeMap<Integer, List<Publication> >(new Comparator<Integer>(){
                public int compare(Integer a, Integer b){
                    return b.compareTo(a);}
            });
    DefaultHandler handler = new UserHandlerTitlePublication(listTitle, this.ourMainMap, publMap);
    SAXHandlerFactory factory = new SAXHandlerFactory(handler);
    factory.makeSAXParser();
    return publMap;
}
```

Here, the XML Parser calls back for the StartElement function where we check the start Variable and mark the corresponding boolean value. Next the characters function follows which we have a ch[] , character array storing all the text within opening and closing tag in the given file. We are matching the title of the publication with the query title and store the number of matches. Next, on call back for the EndElement function , when the end element is article, inproceedings etc (publications), then store the created publication in the map with the key as number of matches.

PATTERNS USED:

FactoryPattern. : for SAXHandler

Singleton Pattern : for Entity Resolution.

MVC pattern : Models : Author, Publication class

View : GUI class

Control : Main class

Publication.java

This contains all the attributes of the Publications. Its functions consist of the Setters, Getters, and toString()

Author.Java

This contains functions for Incrementing NumberOfPublications, get NumberOfPublications, retrieving the AuthorNames from a set of strings, adding AuthorNames to the set. Also, a CompareTo function has been implemented to sort the author by their NumberOfPublications in decreasing order.

GUI.java

This java class has the User Interface implementation which is being done by GroupLayout, the ActionListeners and ActionHandlers for the events.

Main.java

This is basically the controller class where we call the required function for the queries and accordingly sort them according to year or relevance.

Prediction

For prediction we used the two-point linear regression (on non-zero data points) which calculates the relationship between two variables by fitting a linear equation to observed data such that the absolute error is minimised. Also we calculated the average number of publications over years with non-zero publications. With these two we calculated the weighted mean. And hence our result .

Task Distribution

Initially we divided the task into two parts. Akarsha worked on GUI implementation and Kushagra worked on XML Parsing.

We did the integration part together.