XBRL browser

Siyi Kong:G00329103 Xuezhen Zeng:G00307186
Supervisor: John Healy
April 30, 2015

Galway-Mayo Institute of Technology, Ireland
http://www.gmit.ie

Abstract. XBRL is abbreviation of Extensible Business Research Language, which is known as an open standard that is built to suit the financial requirement and sharing business reports across the world, which mainly has two versions, one is IFRS(International Financial Reporting Standards), the other is GAAP, all of which not only shares loads of attributes from XBRL, but contain bunch of other attributes based upon the requirements for our own use. Our project is XBRL browser, it can be used to parse XBRL format file and search the financial terms in the file. There are also heuristic evaluation for the similarity between the words you input and the word from the original text. Finally, when you select the words in the list, there are descriptions shown on the right corner of the panel and you can choose to save it to a new file.

1 Problem Statement

The problems we faced were: How to design a proper user interface. How to design an relatively efficient algorithm to figure out the heuristic. During the process we designed this software, we used Java Swing as a framework to design the UI and we tried to find the solution to store some specific datas and ensured those datas are updated and displayed properly as users constantly make changes to the terms they want to search.

2 Objective

Our goal is to design a XBRL browser that running on desktop. The functionality should at least look like a search engine, which, we import sufficient terms in form of xml, into the browser. Then we starting typing letters and a drop-down list appears. Every changes we make to the searching bar should result in different options popped out from combobox. When press the search button, all the relevant words should be shown. Finally, we can see the similarity and save the item selected in the list to a new file.

3 Literature Review

The core part in this browser is design the algorithm for heuristic evaluation. The algorithm I used is Hamming Distance, which figure out number of positions at which the corresponding symbols are different But some people would also like to use such algorithms as Edit distance which is to compute the minimal time for transformation from one string to another. Both of them have some advantages.

4 System Architecture

Java swing is a very flexible framework which is based upon Awt(Abstract Window Toolkit) but is more flexible and has more features and functionalities to use. So considering the fact that we should make our UI looks more neatly and friendly, we decided to use swing. For decision-making of algorithm, we were thinking about which algorithm is best suited in this project. The reason we choose Hamming distance as its algorithm is because it purely figures out the number of different letters between two strings. First of all, we took the total length for each string and minus them, so that we get the difference in the total length between them . Then I transfer these two strings into char type, which we can compare the letter one by one, if there is any difference between them, we counted it by plus one to a variable. Finally, we add the variable and the difference in total length together, and reversed it, so we can get the similarity. We simply found this method very straight forward.

5 UML Diagram

See final page

6 Screen shot

See final page

7 Performance Evaluation

Our project has successfully achieved the following functionality:
1,Parse xml and xsd file properly and store all the relevant words and description in the buffer.
2,Everytime users input a letter, the Xbrl browser will update options shown in combobox.
3,When users press the button Search, all the possible results will be shown in the list, along with corresponding heuristic values.
4,Switch to dynamic mode, and change the original text and save it.
5,Pick up one or more items in the list, and save it to a new file. So you can

check it by Preview.
Algorithm aspect: In terms of efficiency, it runs fast and while doing search work, it is quiet straight forward and quickly get the result for you.

8 Conclusion

This project gave us a deep understanding of Object-oriented programming and some very important design patterns that will make the project looks better and code looks more neatly. It also polish our knowledge of using Parser to parse xml files and store relevant elements in the buffer for use in future, and above all, stimulate our interest to design effective algorithms to achieve our goal.

During the process of designing this project , we took much effort to think what the best way is to achieve required functionality and to avoid any possible bugs hidden, especially under the circumstance of knowing nothing about financial world. But we get the basics swiftly and diverted most of our energy onto the logic of this project and its implementation as well.

We have also learnt how important XBRL is in financial world and how it helps business of all kinds to save more time, money and laborer effort to get the business onto the nest stage. We really appreciate that we have chance to learn this part of knowledge.

9 Appendix

Installation instructions:
Installation of jdk1.4(or higher) and setting the variable environment are necessary.

Code listing:

```
public String hamming(String userInput,String Xbrltext)
float similarity=0.0f;
float count=0.0f;
String simi;
char[] user=userInput.toCharArray();
char[] xbrl=Xbrltext.toCharArray();
    int shortest=Math.min(user.length,xbrl.length)
int longest=Math.max(user.length,xbrl.length);
    for(int i=0;i¡shortest;i++)
if(user[i]!=xbrl[i])
count+=1;

    count+=longest-shortest;// after figuring out, add the difference between
two strings
    if(count==0.0f)
return ”100

    similarity=(Xbrltext.length()-count)/Xbrltext.length();
NumberFormat format=NumberFormat.getInstance();
format.setMaximumIntegerDigits(2);
format.setMaximumFractionDigits(2);
    simi=format.format(similarity*100)+”
    return simi;
```

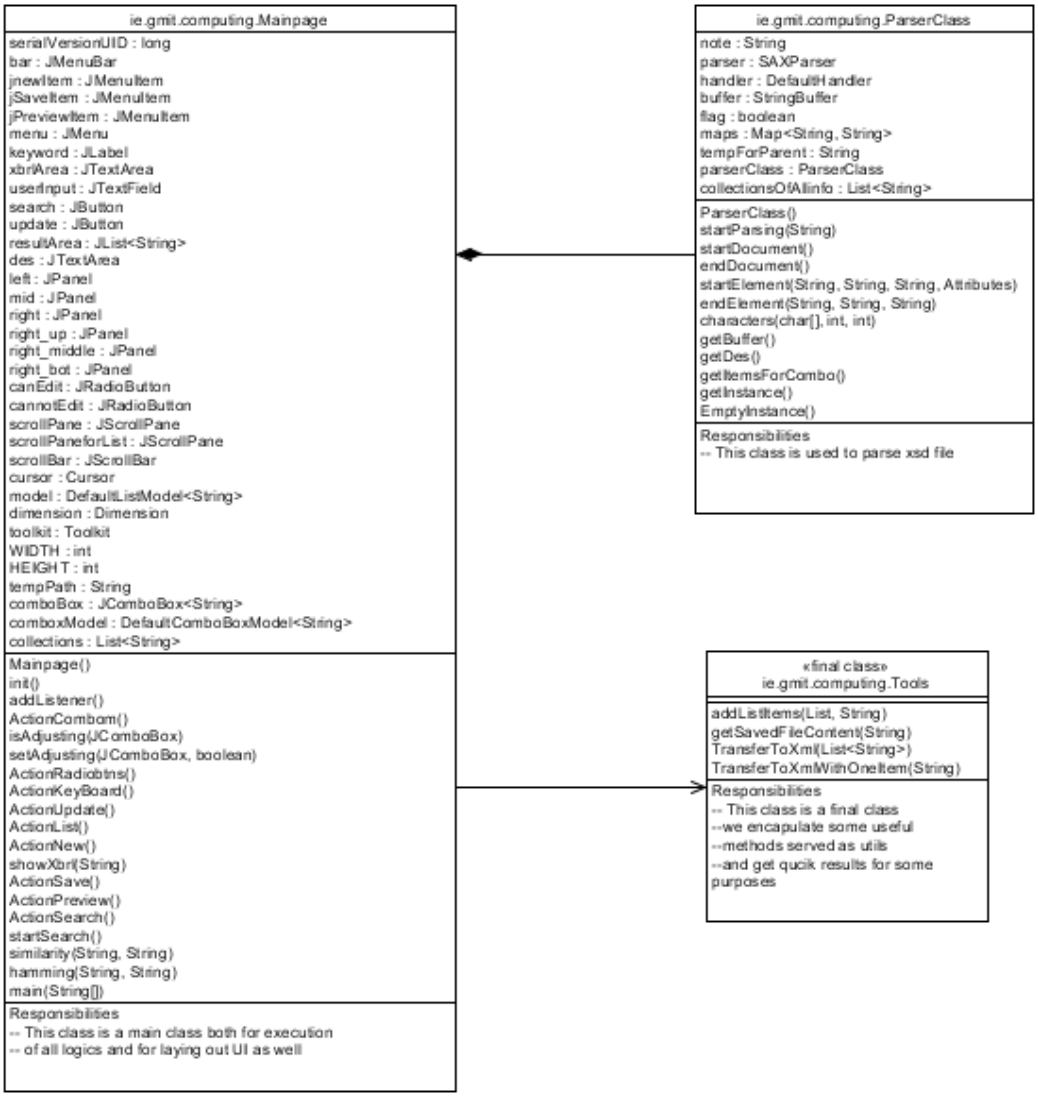
ie.gmit.computing.Mainpage
serialVersionUID : long
bar : JMenuBar
jnewItem : JMenuItem
jSaveItem : JMenuItem
jPreviewItem : JMenuItem
menu : JMenu
keyword : JLabel
xbrlArea : JTextArea
userInput : JTextField
search : JButton
update : JButton
resultArea : JList<String>
des : JTextArea
left : JPanel
mid : JPanel
right : JPanel
right_up : JPanel
right_middle : JPanel
right_bot : JPanel
canEdit : JRadioButton
cannotEdit : JRadioButton
scrollPane : JScrollPane
scrollPaneforList : JScrollPane
scrollBar : JScrollBar
cursor : Cursor
model : DefaultListModel<String>
dimension : Dimension
toolkit : Toolkit
WIDTH : int
HEIGHT : int
tempPath : String
comboBox : JComboBox<String>
comboxModel : DefaultComboBoxModel<String>
collections : List<String>
Mainpage()
init()
addListener()
ActionCombom()
isAdjusting(JComboBox)
setAdjusting(JComboBox, boolean)
ActionRadiobtns()
ActionKeyBoard()
ActionUpdate()
ActionList()
ActionNew()
showXbrl(String)
ActionSave()
ActionPreview()
ActionSearch()
startSearch()
similarity(String, String)
hamming(String, String)
main(String[])
Responsibilities
-- This class is a main class both for execution
-- of all logics and for laying out UI as well
ie.gmit.computing.ParserClass
note : String
parser : SAXParser
handler : DefaultHandler
buffer : StringBuffer
flag : boolean
maps : Map<String, String>
tempForParent : String
parserClass : ParserClass
collectionsOfAllinfo : List<String>
ParserClass()
startParsing(String)
startDocument()
endDocument()
startElement(String, String, String, Attributes)
endElement(String, String, String)
characters(char[], int, int)
getBuffer()
getDes()
getItemsForCombo()
getInstance()
EmptyInstance()
Responsibilities
-- This class is used to parse xsd file
«final class»
ie.gmit.computing.Tools
addListItems(List, String)
getSavedFileContent(String)
TransferToXml(List<String>)
TransferToXmlWithOneItem(String)
Responsibilities
-- This class is a final class
--we encapulate some useful
--methods served as utils
--and get qucik results for some
purposes

Fig. 1. UML

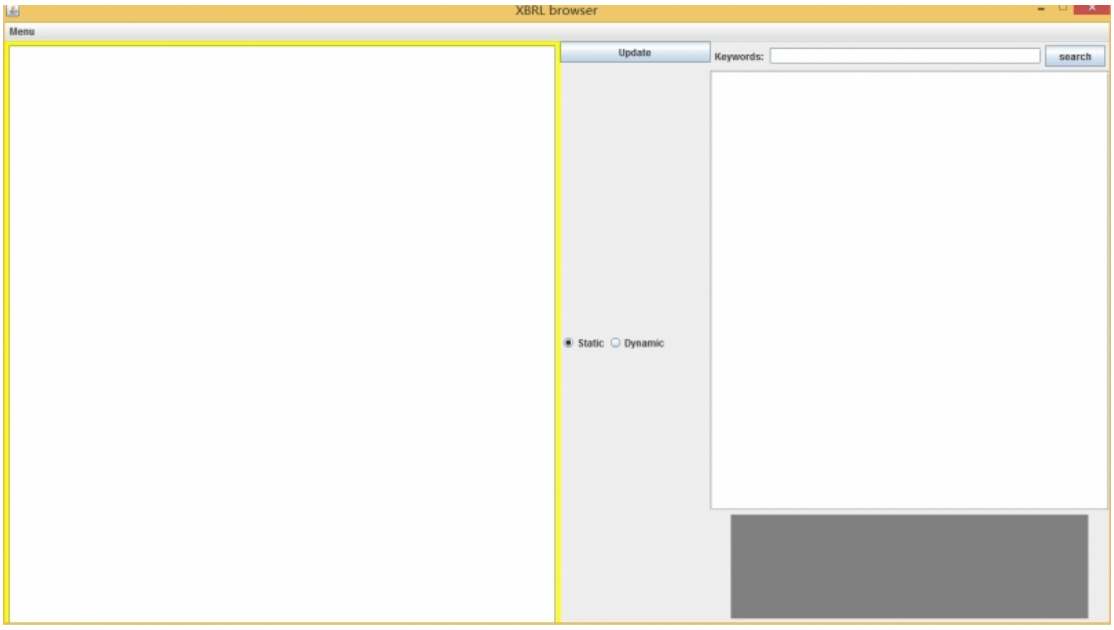

Fig. 2. initial interface

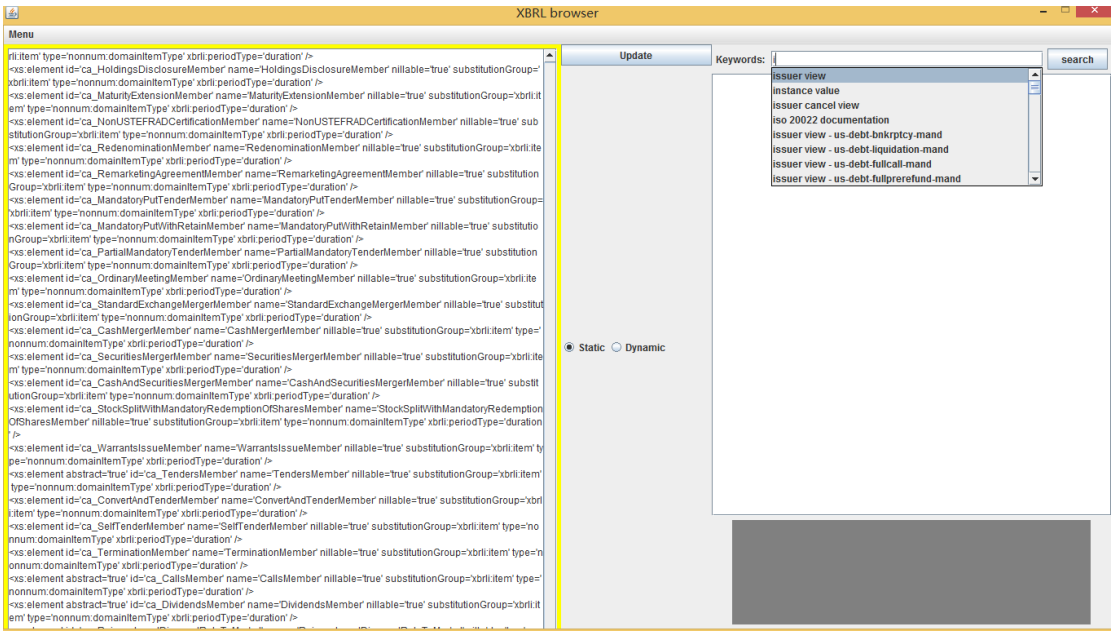

Fig. 3. combobox

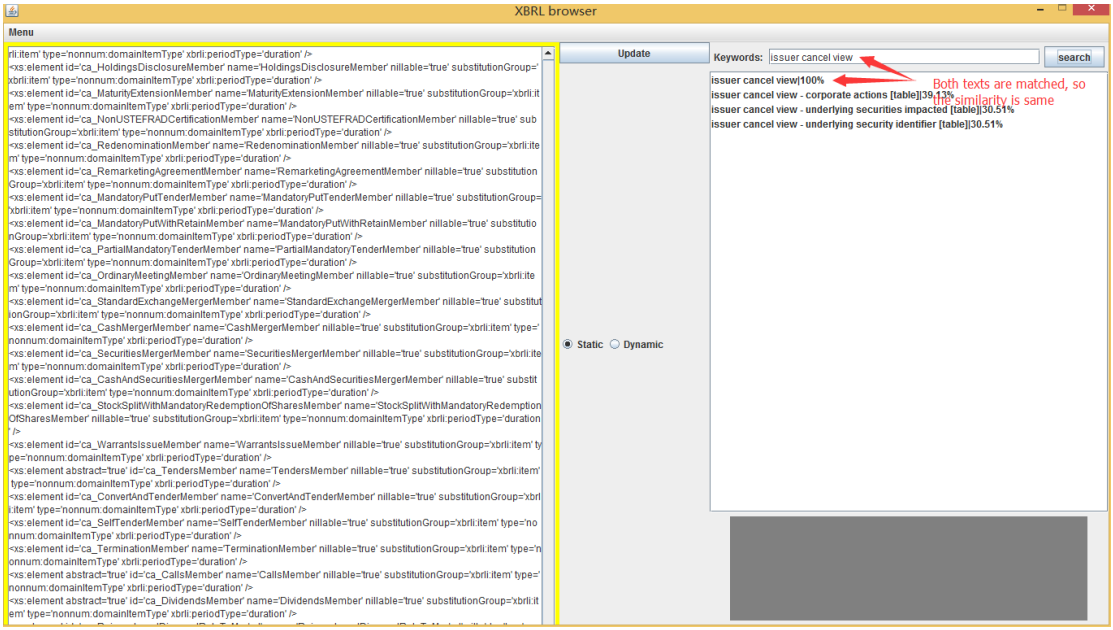

Fig. 4. final result