Project Report

Human Language Technology course

**Student name**: Ehsan Khakifirooz

**Student Number**: 251221

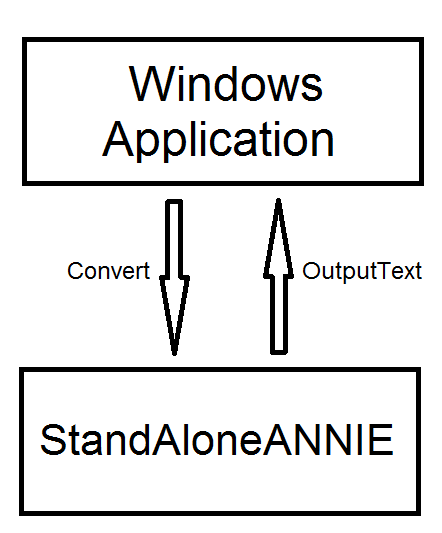
**About the project:**

This project is mainly working on changing the Adjectives in whole text to one of the three available types:

* Love
* Friendly
* Formal

Moreover, it provides some formatting for the text according to above type of letter.

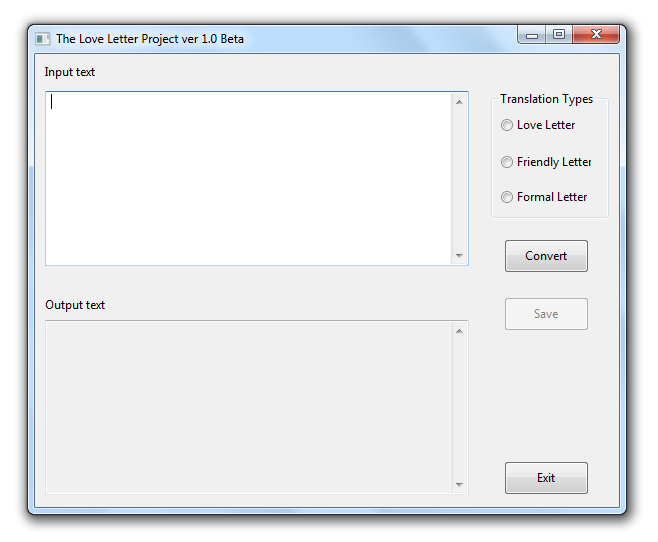
**Main Architecture of the Software:**



It contains two class files which are communicating to each other for doing whole process. The steps are:

1. Windows application is receiving the input text
2. Windows application writes the text into a text file and calls the StanAloneANNIE static Main function.
3. StandAloneANNIE connects program to the Gate ANNIE pipeline and does annotation for **Person** and **Adjective** tokens
4. Then, it formats the adjectives and whole text according to the requested style
5. Windows application shows the converted text (OutputText)

**Windows Application:**



The steps for using this interface are:

1. Putting the considered text for conversion in the input textbox
2. Choosing the type of translation
3. Clicking on the Convert button
4. Waiting for about one minute
5. The result appears in the Output textbox and it can also be saved by pushing the save button.

The used GUI framework is **SWT**.

**Appendix:**

**i-Window application:**

**package** MainCode;

**import** gate.util.GateException;

**public** **class** MainWindow {

**protected** Shell shell;

**private** Text txtInput;

**private** Text txtOutput;

/\*\*

\* Launch the application.

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

**try** {

MainWindow window = **new** MainWindow();

window.open();

} **catch** (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Open the window.

\*/

**public** **void** open() {

Display display = Display.*getDefault*();

createContents();

shell.open();

shell.layout();

**while** (!shell.isDisposed()) {

**if** (!display.readAndDispatch()) {

display.sleep();

}

}

}

/\*\*

\* Create contents of the window.

\*/

**protected** **void** createContents() {

shell = **new** Shell();

shell.setSize(600, 491);

shell.setText("The Love Letter Project ver 1.0 Beta");

Label label = **new** Label(shell, SWT.*NONE*);

label.setText("Input text");

label.setBounds(10, 10, 55, 15);

txtInput = **new** Text(shell, SWT.*BORDER* | SWT.*WRAP* | SWT.*H\_SCROLL* | SWT.*V\_SCROLL* | SWT.*CANCEL*);

txtInput.setBounds(10, 37, 424, 175);

Label label\_1 = **new** Label(shell, SWT.*NONE*);

label\_1.setText("Output text");

label\_1.setBounds(10, 243, 73, 15);

txtOutput = **new** Text(shell, SWT.*BORDER* | SWT.*WRAP* | SWT.*V\_SCROLL*);

txtOutput.setEditable(**false**);

txtOutput.setBounds(10, 266, 424, 175);

Group group = **new** Group(shell, SWT.*NONE*);

group.setText("Translation Types");

group.setBounds(456, 37, 118, 128);

Button btnLoveLetter = **new** Button(group, SWT.*RADIO*);

btnLoveLetter.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

StandAloneAnnie.*emotionState*="Love";

}

});

btnLoveLetter.setText("Love Letter");

btnLoveLetter.setBounds(10, 26, 90, 16);

Button btnFriendlyLetter = **new** Button(group, SWT.*RADIO*);

btnFriendlyLetter.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

StandAloneAnnie.*emotionState*="Friendly";

}

});

//button\_1.setEnabled(false);

btnFriendlyLetter.setText("Friendly Letter");

btnFriendlyLetter.setBounds(10, 62, 98, 16);

Button btnFormalLetter = **new** Button(group, SWT.*RADIO*);

btnFormalLetter.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

StandAloneAnnie.*emotionState*="Formal";

}

});

btnFormalLetter.setText("Formal Letter");

btnFormalLetter.setBounds(10, 98, 90, 16);

**final** Button btnSave = **new** Button(shell, SWT.*NONE*);

btnSave.setEnabled(**false**);

btnSave.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

FileDialog dialog=**new** FileDialog(shell,SWT.*SAVE*);

dialog.setFilterNames(**new** String[] { "Text Files (\*.txt)" });

dialog.setFilterExtensions(**new** String[] { "\*.txt" });

String filterPath = "/";

String platform = SWT.*getPlatform*();

**if** (platform.equals("win32") || platform.equals("wpf")) {

filterPath = "c:\\";

}

String fn = dialog.open();

**if** (fn != **null**) {

File file = **new** File(fn);

FileWriter fileWriter;

**try** {

fileWriter = **new** FileWriter(file);

fileWriter.write(txtOutput.getText());

fileWriter.close();

} **catch** (IOException e1) {

e1.printStackTrace();

}

}

}

});

btnSave.setBounds(469, 243, 85, 34);

btnSave.setText("Save");

Button button\_3 = **new** Button(shell, SWT.*NONE*);

button\_3.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

**if**((txtInput.getText().trim().length()>0)&&(StandAloneAnnie.*emotionState*.trim().length()>0))

{

**try**{

FileWriter writer2 = **new** FileWriter("textInput.txt");

writer2.write(txtInput.getText());

writer2.close();

String inputAr[]={"file:textInput.txt"};

**try**{

StandAloneAnnie.*main*(inputAr);

txtOutput.setText(StandAloneAnnie.*outputText*);

btnSave.setEnabled(**true**);

}

**catch**(GateException ex)

{

}

}

**catch** (IOException ioe)

{

ioe.printStackTrace();

}

}

}

});

button\_3.setText("Convert");

button\_3.setBounds(469, 185, 85, 34);

Button btnExit = **new** Button(shell, SWT.*NONE*);

btnExit.addSelectionListener(**new** SelectionAdapter() {

@Override

**public** **void** widgetSelected(SelectionEvent e) {

shell.dispose();

System.*exit*(0);

}

});

btnExit.setBounds(469, 407, 85, 34);

btnExit.setText("Exit");

}

}

**ii-Adjectives Jape file:**

Rule: adj1

Priority: 2

(

{Token.category=="JJ"}

)

:adj\_1

-->

{

gate.AnnotationSet org = (gate.AnnotationSet) bindings.get("adj\_1");

gate.FeatureMap features = Factory.newFeatureMap();

features.put("Rule ", "adj1");

outputAS.add(org.firstNode(), org.lastNode(), "Adjective",features);

}

**iii-StandAloneANNIE:**

/\*

\* StandAloneAnnie.java

\*

\*

\* Copyright (c) 2000-2001, The University of Sheffield.

\*

\* This file is part of GATE (see http://gate.ac.uk/), and is free

\* software, licenced under the GNU Library General Public License,

\* Version 2, June1991.

\*

\* A copy of this licence is included in the distribution in the file

\* licence.html, and is also available at http://gate.ac.uk/gate/licence.html.

\*

\* hamish, 29/1/2002

\*

\* $Id: StandAloneAnnie.java,v 1.6 2006/01/09 16:43:22 ian Exp $

\*/

package MainCode;

import java.util.\*;

import java.io.\*;

import java.net.\*;

import gate.\*;

import gate.creole.\*;

import gate.util.\*;

import gate.corpora.RepositioningInfo;

/\*\*

\* This class illustrates how to use ANNIE as a sausage machine

\* in another application - put ingredients in one end (URLs pointing

\* to documents) and get sausages (e.g. Named Entities) out the

\* other end.

\* <P><B>NOTE:</B><BR>

\* For simplicity's sake, we don't do any exception handling.

\*/

public class StandAloneAnnie {

public static String inputText="";

public static String outputText="";

public static String personText="";

public static String emotionState="";

public static int positions[][];

/\* The Corpus Pipeline application to contain ANNIE \*/

private SerialAnalyserController annieController;

private static final String[] loveVocab={"Affectionate","Adoring","Caring","Considerate","Devoted","Loving","Beautiful","Attractive","Fair","Lovely","Stunning","Divine","Godly","Heavenly","Glorious","Wonderful","Wondrous","Endles","Continual","Eternal","Perpetual","Unceasing","Unending","Timeless","Faithful","Constant","Steady","Steadfast","True","Fond","Affectionate","Dear","Devoted","Loving","Happy","Cheerful","Glad","Joyful","Joyous","Merry","Pleased","Hopeful","Hoping","Optimistic","Promising","Undying","Deathless","Immortal","Eternal","Everlasting"};

private static final String[] formalVocab={"Academic","Accurate","Active","Adaptable","Adventurous","Affectionate","Aggressive","Alert","Ambitious","Analytical","Artistic","Assertive","Attractive","Bold","Broad-minded","Businesslike","Calm","Capable","Careful","Cautious","Charming","Cheerful","Clear-thinking","Clever","Competent","Competitive","Confident","Conscientious","Conservative","Considerate","Consistent","Cool","Cooperative","Courageous","Curious","Daring","Deliberate","Determined","Dignified","Discreet","Dominant","Eager","Easygoing","Efficient","Emotional","Energetic","Fair-minded","Farsighted","Firm","Flexible","Forceful","Forgiving","Formal","Frank","Friendly","Generous","Gentle","Good-natured","Healthy","Helpful","Honest","Humorous","Imaginative","Independent","Individualistic","Industrious","Informal","Intellectual","Intelligent","Introspective","Inventive","Kind","Liberal","Lighthearted","Likable","Logical, Mature","Methodical","Meticulous","Mild","Moderate","Modest","Motivated","Natural","Obliging","Open-minded","Opportunistic","Optimistic","Organized","Original","Outgoing","Painstaking","Patient","Persevering","Pleasant","Poised","Polite","Practical","Precise","Progressive","Proud","Prudent","Purposeful","Quick","Quiet","Rational","Realistic","Reflective","Relaxed","Reliable","Reserved","Resourceful","Responsible","Robust","Self-confident","Sensible","Sensitive","Serious","Sharp-witted","Sincere","Sociable","Spontaneous","Spunky","Stable","Steady","Strong","Strong-minded","Supportive","Tactful","Teachable","Tenacious","Thorough","Thoughtful","Tolerant","Tough","Trusting","Trustworthy","Unaffected","Unassuming","Understanding","Unexcitable","Uninhibited","Verbal","Versatile Warm","Wholesome","Wise","Witty","Zany"};

private static final String[] friendlyVocab={"Cheerful","Easygoing","Efficient","Energetic","Friendly","Intelligent","Kind","Likable","Modest","Motivated","Outgoing","Persevering","Pleasant","Proud","Quiet","Sensible","Sensitive","Serious","Sincere","Strong","Tactful","Trusting","Trustworthy","Wise"};

/\*

\* Initialise the ANNIE system. This creates a "corpus pipeline"

\* application that can be used to run sets of documents through

\* the extraction system.

\*/

public void initAnnie() throws GateException {

Out.prln("Initialising ANNIE...");

System.out.print("ANNIE\_" + Gate.genSym());

// create a serial analyser controller to run ANNIE with

annieController =

(SerialAnalyserController) Factory.createResource(

"gate.creole.SerialAnalyserController", Factory.newFeatureMap(),

Factory.newFeatureMap(), "ANNIE\_" + Gate.genSym()

);

// load each PR as defined in ANNIEConstants

for(int i = 0; i < ANNIEConstants.PR\_NAMES.length; i++) {

FeatureMap params = Factory.newFeatureMap(); // use default parameters

ProcessingResource pr = (ProcessingResource) Factory.createResource(ANNIEConstants.PR\_NAMES[i], params);

// add the PR to the pipeline controller

annieController.add(pr);

} // for each ANNIE PR

Out.prln("...ANNIE loaded");

} // initAnnie()

/\* Tell ANNIE's controller about the corpus you want to run on \*/

public void setCorpus(Corpus corpus) {

annieController.setCorpus(corpus);

} // setCorpus

/\* Run ANNIE \*/

public void execute() throws GateException {

Out.prln("Running ANNIE...");

annieController.execute();

Out.prln("...ANNIE complete");

} // execute()

/\*

\* Run from the command-line, with a list of URLs as argument.

\* <P><B>NOTE:</B><BR>

\* This code will run with all the documents in memory - if you

\* want to unload each from memory after use, add code to store

\* the corpus in a DataStore.

\*/

public static void main(String args[])

throws GateException, IOException {

// initialise the GATE library

Out.prln("Initialising GATE...");

Gate.init();

// Load ANNIE plugin

File gateHome = Gate.getGateHome();

System.out.print(gateHome.toPath().toString());

File pluginsHome = new File(gateHome, "plugins");

Gate.getCreoleRegister().registerDirectories(new File(pluginsHome, "ANNIE").toURL());

Out.prln("...GATE initialised");

// initialise ANNIE (this may take several minutes)

StandAloneAnnie annie = new StandAloneAnnie();

annie.initAnnie();

// create a GATE corpus and add a document for each command-line

// argument

Corpus corpus = (Corpus) Factory.createResource("gate.corpora.CorpusImpl");

/\* for(int i = 0; i < args.length; i++) {

URL u = new URL(args[i]);\*/

URL u = new URL(args[0]);

FeatureMap params1 = Factory.newFeatureMap();

params1.put("sourceUrl", u);

params1.put("preserveOriginalContent", new Boolean(true));

params1.put("collectRepositioningInfo", new Boolean(true));

Out.prln("Creating doc for " + u);

Document doc1 = (Document)

Factory.createResource("gate.corpora.DocumentImpl", params1);

corpus.add(doc1);

// } // for each of args

// tell the pipeline about the corpus and run it

annie.setCorpus(corpus);

annie.execute();

// for each document, get an XML document with the

// person and location names added

Iterator iter = corpus.iterator();

int count = 0;

String startTagPart\_1 = "<span GateID=\"";

String startTagPart\_2 = "\" title=\"";

String startTagPart\_3 = "\" style=\"background:Red;\">";

String endTag = "</span>";

while(iter.hasNext()) {

Document doc = (Document) iter.next();

AnnotationSet defaultAnnotSet = doc.getAnnotations();

Set annotTypesRequired = new HashSet();

annotTypesRequired.add("Adjective");

annotTypesRequired.add("Person");

Set<Annotation> peopleAndPlaces =

new HashSet<Annotation>(defaultAnnotSet.get(annotTypesRequired));

//FeatureMap features = doc.getFeatures();

String originalContent = (String)

doc.getFeatures().get(GateConstants.ORIGINAL\_DOCUMENT\_CONTENT\_FEATURE\_NAME);

RepositioningInfo info = (RepositioningInfo)

doc.getFeatures().get(GateConstants.DOCUMENT\_REPOSITIONING\_INFO\_FEATURE\_NAME);

System.out.println((String)

doc.getFeatures().get(GateConstants.ORIGINAL\_DOCUMENT\_CONTENT\_FEATURE\_NAME));

// System.out.println((String)features.get(GateConstants.DOCUMENT\_REPOSITIONING\_INFO\_FEATURE\_NAME));

++count;

File file = new File("StANNIE\_" + count + ".HTML");

Out.prln("File name: '"+file.getAbsolutePath()+"'");

if(originalContent != null && info != null) {

Out.prln("OrigContent and reposInfo existing. Generate file...");

Iterator it = peopleAndPlaces.iterator();

Annotation currAnnot;

SortedAnnotationList sortedAnnotations = new SortedAnnotationList();

while(it.hasNext()) {

currAnnot = (Annotation) it.next();

sortedAnnotations.addSortedExclusive(currAnnot);

} // while

StringBuffer editableContent = new StringBuffer(originalContent);

long insertPositionEnd;

long insertPositionStart;

// insert anotation tags backward

Out.prln("Unsorted annotations count: "+peopleAndPlaces.size());

Out.prln("Sorted annotations count: "+sortedAnnotations.size());

for(int i=sortedAnnotations.size()-1; i>=0; --i) {

currAnnot = (Annotation) sortedAnnotations.get(i);

insertPositionStart =

currAnnot.getStartNode().getOffset().longValue();

insertPositionStart = info.getOriginalPos(insertPositionStart);

insertPositionEnd = currAnnot.getEndNode().getOffset().longValue();

insertPositionEnd = info.getOriginalPos(insertPositionEnd, true);

if(insertPositionEnd != -1 && insertPositionStart != -1) {

editableContent.insert((int)insertPositionEnd, endTag);

editableContent.insert((int)insertPositionStart, startTagPart\_3);

editableContent.insert((int)insertPositionStart,

currAnnot.getType());

editableContent.insert((int)insertPositionStart, startTagPart\_2);

editableContent.insert((int)insertPositionStart,

currAnnot.getId().toString());

editableContent.insert((int)insertPositionStart, startTagPart\_1);

} // if

} // for

FileWriter writer = new FileWriter(file);

writer.write(editableContent.toString());

writer.close();

} // if - should generate

else if (originalContent != null) {

Out.prln("OrigContent existing. Generate file...");

Iterator it = peopleAndPlaces.iterator();

Annotation currAnnot;

SortedAnnotationList sortedAnnotations = new SortedAnnotationList();

while(it.hasNext()) {

currAnnot = (Annotation) it.next();

sortedAnnotations.addSortedExclusive(currAnnot);

} // while

StringBuffer editableContent = new StringBuffer(originalContent);

long insertPositionEnd;

long insertPositionStart;

// insert anotation tags backward

Out.prln("Unsorted annotations count: "+peopleAndPlaces.size());

Out.prln("Sorted annotations count: "+sortedAnnotations.size());

for(int i=sortedAnnotations.size()-1; i>=0; --i) {

currAnnot = (Annotation) sortedAnnotations.get(i);

insertPositionStart =

currAnnot.getStartNode().getOffset().longValue();

insertPositionEnd = currAnnot.getEndNode().getOffset().longValue();

if(insertPositionEnd != -1 && insertPositionStart != -1) {

editableContent.insert((int)insertPositionEnd, endTag);

editableContent.insert((int)insertPositionStart, startTagPart\_3);

editableContent.insert((int)insertPositionStart,

currAnnot.getType());

editableContent.insert((int)insertPositionStart, startTagPart\_2);

editableContent.insert((int)insertPositionStart,

currAnnot.getId().toString());

editableContent.insert((int)insertPositionStart, startTagPart\_1);

} // if

} // for

FileWriter writer = new FileWriter(file);

writer.write(editableContent.toString());

writer.close();

}

else {

Out.prln("Content : "+originalContent);

Out.prln("Repositioning: "+info);

}

String xmlDocument = doc.toXml(peopleAndPlaces, false);

String fileName = new String("StANNIE\_toXML\_" + count + ".HTML");

FileWriter writer = new FileWriter(fileName);

writer.write(xmlDocument);

writer.close();

try

{

File file2 = new File("StANNIE\_toXML\_" + count + ".HTML");

BufferedReader reader = new BufferedReader(new FileReader(file2));

String line = "", oldtext = "";

while((line = reader.readLine()) != null)

{

oldtext += line + "\r\n";

}

reader.close();

int positionBegin=0;

int positionEnd=0;

String oldtext2=oldtext;

String keywordBegin="<Adjective>";

String keywordEnd="</Adjective>";

oldtext2=oldtext2.replaceAll("<paragraph>", "");

oldtext2=oldtext2.replaceAll("</paragraph>", "");

if(emotionState=="Love")

{

while((positionBegin=oldtext2.indexOf(keywordBegin,positionBegin))>-1)

{

positionEnd=oldtext2.indexOf(keywordEnd,positionBegin);

String str4=oldtext2.substring(positionBegin+keywordBegin.length(),positionEnd);

oldtext2= oldtext2.replaceFirst(keywordBegin+str4, loveVocab[(new Random()).nextInt(loveVocab.length)]);

oldtext2= oldtext2.replaceFirst(keywordEnd, "");

}

}else if(emotionState=="Formal")

{

while((positionBegin=oldtext2.indexOf(keywordBegin,positionBegin))>-1)

{

positionEnd=oldtext2.indexOf(keywordEnd,positionBegin);

String str4=oldtext2.substring(positionBegin+keywordBegin.length(),positionEnd);

oldtext2= oldtext2.replaceFirst(keywordBegin+str4, formalVocab[(new Random()).nextInt(formalVocab.length)]);

oldtext2= oldtext2.replaceFirst(keywordEnd, "");

}

}else if(emotionState=="Friendly")

{

while((positionBegin=oldtext2.indexOf(keywordBegin,positionBegin))>-1)

{

positionEnd=oldtext2.indexOf(keywordEnd,positionBegin);

String str4=oldtext2.substring(positionBegin+keywordBegin.length(),positionEnd);

oldtext2= oldtext2.replaceFirst(keywordBegin+str4, friendlyVocab[(new Random()).nextInt(friendlyVocab.length)]);

oldtext2= oldtext2.replaceFirst(keywordEnd, "");

}

}

keywordBegin="<Person>";

keywordEnd="</Person>";

Boolean personSelected=false;

String personString="";

positionBegin=0;

while((positionBegin=oldtext2.indexOf(keywordBegin,positionBegin))>-1)

{

positionEnd=oldtext2.indexOf(keywordEnd,positionBegin);

if(!personSelected){

personString=oldtext2.substring(positionBegin+keywordBegin.length(),positionEnd);

personSelected=true;

break;

}

//oldtext2= oldtext2.replaceFirst(keywordBegin+str4, loveVocab[(new Random()).nextInt(loveVocab.length)]);

//

}

oldtext2 = oldtext2.replaceAll(keywordBegin, "");

oldtext2 = oldtext2.replaceAll(keywordEnd, "");

// FileWriter writer2 = new FileWriter("StANNIE\_toXML\_" + count + "\_3.HTML");

// writer2.write(oldtext2);writer2.close();

outputText=oldtext2;

personText=personString;

if(emotionState=="Love")

{

//code for the state that there is no name!

if(personString.length()>0){

outputText="Darling "+personString+",\n\n"+outputText+"\nI wish our love stays forever";

}

else{

outputText="My darling,\n\n"+outputText+"\nI wish our love stays forever";

}

}

else if (emotionState=="Formal")

{

if(personString.length()>0){

outputText="Dear "+personString+",\n\n"+outputText+"\nI will look forward to your reply";

}

else{

outputText="Dear Madam/sir,\n\n"+outputText+"\nI will look forward to your reply";

}

}

else if (emotionState=="Friendly")

{

if(personString.length()>0){

outputText="Hey "+personString+",\n\n"+outputText+"\nHope to see you so soon";

}

else{

outputText="Hey,\n\n"+outputText+"\nHope to see you so soon";

}

}

// String newtext = oldtext.replaceAll("<Adjective>", "");

// String newtext1 = newtext.replaceAll("</Adjective>", "");

// FileWriter writer1 = new FileWriter("StANNIE\_toXML\_" + count + "\_2.HTML");

// writer1.write(newtext1);writer1.close();

}

catch (IOException ioe)

{

ioe.printStackTrace();

}

// do something usefull with the XML here!

// Out.prln("'"+xmlDocument+"'");

} // for each doc

} // main

/\*

\*

\*/

public static class SortedAnnotationList extends Vector {

public SortedAnnotationList() {

super();

} // SortedAnnotationList

public boolean addSortedExclusive(Annotation annot) {

Annotation currAnot = null;

// overlapping check

for (int i=0; i<size(); ++i) {

currAnot = (Annotation) get(i);

if(annot.overlaps(currAnot)) {

return false;

} // if

} // for

long annotStart = annot.getStartNode().getOffset().longValue();

long currStart;

// insert

for (int i=0; i < size(); ++i) {

currAnot = (Annotation) get(i);

currStart = currAnot.getStartNode().getOffset().longValue();

if(annotStart < currStart) {

insertElementAt(annot, i);

/\*

Out.prln("Insert start: "+annotStart+" at position: "+i+" size="+size());

Out.prln("Current start: "+currStart);

\*/

return true;

} // if

} // for

int size = size();

insertElementAt(annot, size);

//Out.prln("Insert start: "+annotStart+" at size position: "+size);

return true;

} // addSorted

} // SortedAnnotationList

} // class StandAloneAnnie