

Discourse Analysis Group

John Carrasquillo, Oliver Nichols, Crystal Ray, Heather Scott

CS 491: Software Engineering II

November 20, 2013

Table of Contents

Project Description	page 3
Setup Documentation	page 4
Current Progress	page 7
Issues Remaining <i><u>IMPORTANT</u></i>	page 8
Code Documentation	page 9
Pending Tasks	page 14
Appendix (screenshots, database schema)	page 15

Project Description

The problem solved by this system is to subject a literary work (in this case, books from the Bible) to basic discourse analysis. Since discourse analysis provides the spine of the system's core structure and logic, a summary description of the concepts involved in discourse analysis will be presented here. In general, discourse analysis is the process by which a literary body or sequence of messages are split into clauses. Clauses are the most basic, atomic unit of meaning in the communication of ideas. Discourse analysis seeks to show the relationship between clauses and the meanings conveyed in them. The words or sequences of words that join the clauses are referred to as conjunctions. Although, in some cases, implied conjunctions may be inserted between clauses where there was originally no words.

The form of discourse analysis used in this system focuses on generating an outline reflecting the contents of the source document. This approach is based on the idea that any meaningful discourse can be synonymously represented in outline form, possibly the outline that the original author used in the writing of the discourse being examined. This system also takes advantage of the notion that any well-formed outline has a one-to-one mapping to a tree diagram that conveys the same information. Using these concepts, we developed a system that satisfies the client's specifications.

This system takes advantage of discourse analysis to parse and split sentences (again, verses from books of the Bible) into clauses and generate a tree diagram of all the contained clauses. This tree diagram may then be utilized by users of the system to summarize and condense several related clauses into a user-derived clause that still conveys most of the meaning of the original constituent clauses. In this manner, entire sections of the source document may be condensed into summary statements that preserve most of the original meaning. The most practical application of this system is as a reference tool, helping a user to organize and better understand meanings conveyed in the Bible.

Purpose:

The purpose of this project was to create a website that allows users to select or upload a “conjunction list” file or and upload a formatted or unformatted text file, parse the text, open it in an applet for editing, and finally save the edited data as an XML file. Our task for this semester was to improve upon last semester’s progress. The following document outlines the changes we made as well as improvements that should be made in the future.

Setup Documentation

The following section contains a setup guide for the discourse analysis project. This guide provides information regarding the following software used to assist in the development of this project: XAMPP,

GitHub (optional), and Netbeans.

XAMPP

a. Description

- XAMPP is a program that allows one's computer to act as a server. This means that you can execute server-side files (such as PHP files) on your own computer for testing.

b. Installation and Setup

- You can download the latest version of XAMPP from (<http://www.apachefriends.org/en/xampp.html>)
- Clone the git repository (See Dr. Plotnick's GitHub Documentation) for the Website folder into the htdocs folder found within the XAMPP directory
- Clone the git repository (See Dr. Plotnick's GitHub Documentation) for the Applet folder into the htdocs folder that is now located within the XAMPP directory

c. How to use

- To start XAMPP, go to your XAMPP directory and execute xampp-control.exe (Windows) or manager-osx (Mac)
- In the XAMPP control, start the Apache server and the MySQL server
 - Note: Skype can interfere with XAMPP because they use the same port. You can change the port settings in XAMPP or avoid using Skype when using XAMPP.
- Open your favorite browser
- In the address bar, type "localhost"
- From this page, you can access many resources, including "phpMyAdmin" for database resources.
- To view php or html files, type "localhost/phpFilePath", for example "localhost/Website/myFiles.php". Note: The file path begins inside the htdocs folder. The php files might throw errors if you haven't followed the next set of instructions of creating the database.

d. Creating the Database

- Open phpMyAdmin located at localhost on the left side or type ("localhost/phpmyadmin"). You should see a sidebar on the left where the current databases are listed.
- To create a new database, click the "Databases" tab in the top left corner of the main contents panel (just to the right of the sidebar).

- Type “DiscourseAnalysis” in the field labeled “Create database” and click create. You should see the new database listed below the create database field as well as in the sidebar.
- Click the DiscourseAnalysis database. From here you can edit the properties and permissions of the table as well as creating and executing queries.
- Click the “Import” tab. This opens a dialog where you can import .sql files to create tables within your database.
- Click “Browse for files” and navigate to where the .sql file, DiscourseAnalysis_Database.sql, provided for you in the Website folder. Select the file and click “Go”. You should see a notification at the top of the page letting you know that the import has been successful.
- Navigate back to the “Structure” tab and verify that your database now contains tables.
- To log into the website, a username and password were included in the import. Use “user” and “pass” for your login credentials.

GitHub

Github is the git interface we chose to use for this project. All of the project files are located in a repository on Github. The following section explains how to setup Github, pull changes from the server, and push local changes to the server.

- **Setting up GitHub**
 - Navigate to github.com
 - If you haven’t already done so, create a github account and download github to your computer
 - Open github and log in with the account you just created. Your instructor will need your account information to share the repository with you. Once this is done, you should see the CS 491 project repositories listed under github when you login.
- **Cloning a repository**
 - Click the project repository listed under github
 - Mouse over each folder and click the clone button. You should clone the repositories as directed in the XAMMP section.
 - Navigate to your XAMMP htdocs folder. You should now see all of the project repository folders in this directory.
- **Pulling from the server**
 - Navigate to the “My Repositories” folder on GitHub. This shows all of the repositories located on your local machine.

- If there are new changes on the server that have not been pulled to your local repository, a bold “New Commits” message will be shown on the repository containing the changes.
 - Click the arrow on this repository. This will show you all of the changes that have not been synced.
 - To pull the changes to your local repository, click the “Sync Branch” button. The changes should now be reflected in your local repository.
- **Pushing to the server**
 - Navigate to the “My Repositories” folder on GitHub.
 - Click the arrow on the repository containing your changes.
 - A list of all of your changes will be displayed.
 - Click the “Select All” checkbox to commit all the changes or selectively check individual changes to be committed.
 - Add a commit summary and description and click commit. The commit will now be shown under “Unsynced Commits”.
 - Click the “sync” button to share these changes or the “edit” button to edit the changes.

NetBeans

Go to <http://netbeans.org/downloads/> and download the latest version of the NetBeans SE IDE for your operating system.

- **Create a Project for the Applet**
 - With NetBeans open, Click File -> New Project -> Next.
 - Give it an appropriate name, and uncheck the Create Main Class checkbox. Click Finish.

Current Progress:

The following section describes the progress that was made on the project. The majority of the semester was spent making major improvements to the existing functionality. This improved not only the way the project functioned but the level of difficulty in transferring the project between semesters.

- Overcame difficulties in retrieving and transferring the project's repository and documentation over from the last semester's group
- Generated documentation for setting up the project for the group that inherits the project next semester
- Debugged and corrected the previous group's PHP code and corrected the bug causing the upload process to throw an error
- Integrated the parser into the Website using PHP, thus essentially making it part of the Website itself
- Integrated the different pieces of the project into a single cohesive project
- Connected the applet from fall 2012 due to issues with the spring 2013 applet
- Implemented functionality to incorporate the client's desire to be able to parse unformatted text
- Implemented functionality to upload files and retrieve files from the database
- Updated website to display files stored in the database
- Updated database schema to follow a consistent naming convention

Issues Remaining - *IMPORTANT*****

The following section describes our recommendation in dealing with the applet part of the project. These recommendations were based on issues we encountered while connecting everything together. These are simply recommendations but we feel this would be the most efficient way to proceed with this part of the project.

Our Recommendation

We highly recommend NOT using the java applet. This creates additional work, but there are issues with using the java applet. A java applet cannot run in Chrome, it is seen as a high security risk in Firefox, and applet use has to be enabled in the browser for the applet to work at all. This limits the number of users that can or will use the software. The applet jar file certificate will also have to be resigned every 6 months to be usable in browsers. This will be cumbersome for maintenance after the project has been implemented and might be a problem for the client.

One recommendation is using JQuery and JQuery UI. Here is a link to JQuery UI where you can view some of its functionality: <http://jqueryui.com/>.

JQuery is JavaScript made easy. Essentially it is set up like an API in that you can reference the JQuery functions by including a reference to Google's hosted JQuery library. An example of this is as follows:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js" ></script>
```

Note that the "1.10.2" might be an outdated library version, which would mean you would have to lookup the current library version and put its version number in place of the "1.10.2" in the example above.

Another option is to download it and host the library locally, but we recommend you use the Google-hosted library, which can be viewed at: <http://code.google.com/apis/ajaxlibs/>

It can easily manipulate HTML. It might be possible to replicate current applet functionality by allowing JQuery to use div, textarea, and similar HTML tags.

Coding Documentation

The following section provides a list of files that were updated as well as a detailed description of changes that were made. This documentation can also be found within the code itself.

- **fileUpload.php**

- This file will accept the file from "upload.php" to upload the database
- Uses the fileModule.php to upload files
- Connection comes from the header.php
- **fileModule.php**
 - This is the database file module for uploading and downloading files
 - FileModule(\$connection)
 - Uses the connection that was given
 - Function upload
 - Allows you to upload a file based on username
 - Function getFilesInfo
 - Allows you to get files' information based on username
 - Returns an array
 - Function getPublicFilesInfo
 - Allows you to get public files' information
 - Function getFileContents
 - Allows you to get file contents based on username and filename
 - Function validUserName
 - Internal function
 - Checks to see if the username is valid
 - Function fileExists
 - Internal Function
 - Checks to see if filename exists under the username
- **Parser.php**
 - pconj's are potential conjunctions
 - Function Parser
 - Uses a default parameter, so one is not required
 - Ex: \$myParser = new Parser (); OR

```
$myParser = new Parser($myArray);
```

■ Function `parseUnformattedText`

- There is no format required for the parsing of unformatted text
- The applet will display the entirety of the inputted text file in a single node, where the user can then choose his or her own breaks, logical or not.

■ Function `parseFormattedText`

- The format for the inputted text file is as follows:

```
<chapter>:<verse> <conjunction>
```

```
<clause>
```

```
<conjunction>
```

```
<clause>
```

```
...
```

```
..
```

```
.
```

```
*****
```

- *An example of the above format being used in a real input file:*

```
1:1 X
```

```
It seemed good to me also
```

```
X
```

```
having had perfect understanding of
```

*all things from the very first to write you
an orderly account, [most] excellent
Theophilus*

!!NOTE!!

*~ That <chapter> and <verse> are OPTIONAL;
if omitted, then blanks: " "
will be inserted in their place and the
colon will be left out, too*

*~ If the <chapter> and <verse> are stated
once, the remaining conjunction lines will
hold the same <chapter> and <verse> until
the <chapter> and <verse> are specified.*

Ex.

1:1 X

It seemed good to me also

X

*having had perfect understanding of all
things*

1:2 X

from the very first to write

--Is the same as--

1:1 X

It seemed good to me also

--> 1:1 X

*having had perfect understanding of all
things*

1:2 X

from the very first to write

*~ If two clauses are next to each other on
separate lines, the conjunction X will be
placed between the two clauses.*

Ex.

It seemed good to me also

*having had perfect understanding of all
things*

--Is the same as--

It seemed good to me also

X

*having had perfect understanding of all
things*

~ The conjunction X signifies that the conjunction is logically implied, thus suggesting a logical break, unless it is the X at the very beginning of the text, following the first <chapter> and <verse>. This X signifies the beginning of the text itself

Pending Tasks

The following section provides a detailed list of tasks that were not completed due to time constraints.

Applet

Tasks we noticed that are needed in the Java Applet from Spring 2013

1. Fix grouping functionality
2. Optimize merge functionality (smaller verse number is displayed, larger verse displayed in text)
3. Applet Split (conjunction highlighting)
4. The applet should not ask for a local file. It should use the file that it was given.

Website

1. fileUpload.php does not give feedback on upload errors to the user. It might be useful to return the user to upload.php, if the upload is unsuccessful and tell them the issue. The name field is required, but the user can upload a blank filename in some browsers. This bug should be fixed.
2. The website needs to be further connected to the applet (if it is continued). The applet should be passed the contents of the chosen file (using fileModule.php to get contents of file).
3. The website should display public files in addition to user files when the user views files. (There is a function in fileModule.php that can get all the public file information, it just needs to be placed into the website).
4. The registration input fields need to be required so that the user cannot leave them blank when submitting the form.
5. Query error in AdminModule.php from Spring 2013 that we found. (around line 305 and on)
6. Admin Options in the website need to be fully and correctly implemented.
7. (Suggestion) - If the user selects multiple files to view or edit, maybe open each file in a new tab.

Parser

1. If there are any issues with parsing formatted text, first check that the text file was formatted correctly and then analyze the parser. The parser is in parser.php. When we tested the parser with correct formatting, the parser performed correctly.


Documentation

Be sure to document and add comments to new code so that future semesters can easily pick up where you left off.

Appendix

The following section provides screenshots of the progress made on the website and the update database schema.

Home



[Home](#) [Upload to Workspace](#) [My Files](#) [Administrative Options](#) [Login / Register](#)

JSU CS491 Discourse Analysis


Spring 2013

The problem solved by this system is to subject a literary work (in this case, books from the Bible) to basic discourse analysis. Since discourse analysis provides the spine of the system's core structure and logic, a summary description of the concepts involved in discourse analysis will be presented here. In general, discourse analysis is the process by which a literary body or sequence of messages are split into clauses. Clauses are the most basic, atomic unit of meaning in the communication of ideas. Discourse analysis seeks to show the relationship between clauses and the meanings conveyed in them. The words or sequences of words that join the clauses are referred to as conjunctions. Although, in some cases, implied conjunctions may be inserted between clauses where there was originally no words.

The form of discourse analysis used in this system focuses on generating an outline reflecting the contents of the source document. This approach is based on the idea that any meaningful discourse can be synonymously represented in outline form, possibly the outline that the original author used in the writing of the discourse being examined. This system also takes advantage of the notion that any well-formed outline has a one-to-one mapping to a tree diagram that conveys the same information. Using these concepts, we developed a system that satisfies the client's specifications.

This system takes advantage of discourse analysis to parse and split sentences (again, verses from books of the Bible) into clauses and generate a tree diagram of all the contained clauses. This tree diagram may then be utilized by users of the system to summarize and condense several related clauses into a user-derived clause that still conveys most of the meaning of the original constituent clauses. In this manner, entire sections of the source document may be condensed into summary statements that preserve most of the original meaning. The most practical application of this system is

Registration



[Home](#)
[Upload to Workspace](#)
[My Files](#)
[Administrative Options](#)
[Login / Register](#)

Welcome, user ! ([logout](#))

Register Here !

Username


Password

Password Again

Email Address


First Name

Last Name

 [↔](#)

Enter Code:

Upload to Workspace



[Home](#)
[Upload to Workspace](#)
[My Files](#)
[Administrative Options](#)
[Login / Register](#)

Welcome, user ! ([logout](#))

Please choose your file to upload

File Name:


File Location: No file chosen

Make this file public? ☐

Use default conjunction list? ☒

Is your text file formatted? ☐

My Files



Home Upload to Workspace My Files Administrative Options Login / Register

Welcome, user ! ([logout](#))

	File Name	Public	Last Update
<input type="checkbox"/>	another	Yes	2013-11-08 20:52:40
<input type="checkbox"/>	fname	No	2013-11-13 11:37:46
<input type="checkbox"/>	sample	No	0000-00-00 00:00:00
<input type="checkbox"/>	some file	No	2013-11-06 12:20:17
<input type="checkbox"/>	test	Yes	2013-10-29 20:52:36

Edit in workspace View in workspace Delete File

Database Schema

files

This table contains information about files uploaded to the database. The owner of the file, the filename, the location of the file, and the date of the file's last update.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> 1	owner	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
<input type="checkbox"/> 2	fileName	varchar(35)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
<input type="checkbox"/> 3	fileLocation	varchar(150)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
<input type="checkbox"/> 4	lastUpdate	datetime			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

permissions

This table contains information about the users' permissions. This table keeps up with whether or not the user is an admin, their username, the file they have uploaded, and whether they have permissions to edit, read, delete, or add.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> 1	admin	tinyint(1)			No	0		Change Drop Primary Unique Index More
<input type="checkbox"/> 2	userName	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 3	fileName	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 4	edit	tinyint(1)			No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 5	read	tinyint(1)			No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 6	delete	tinyint(1)			No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 7	add	tinyint(1)			No	None		Change Drop Primary Unique Index More

sessions

This table contains information about the users' session. The table keeps track of the username, start and end time, and a session ID.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> 1	userName	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 2	startTime	datetime			No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 3	endTime	datetime			No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 4	sessionID	int(15)			No	None		Change Drop Primary Unique Index More

tempusersinfo

This table keep track of temporary user information while the user is registering. The table contains the users confirmation code, username, password, email, name and admin status. Once the user has confirmed their registration, their information is moved to the usersinfo table.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> 1	confirmCode	varchar(65)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 2	userName	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 3	password	varchar(60)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 4	email	varchar(75)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 5	name	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 6	admin	tinyint(1)			No	0		Change Drop Primary Unique Index More

usersinfo

This table contains information for users that are registered with the system. This table contains the users username, password, email, name and admin status.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> 1	<u>userName</u>	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 2	password	varchar(60)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 3	email	varchar(75)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 4	name	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index More
<input type="checkbox"/> 5	admin	tinyint(1)			No	0		Change Drop Primary Unique Index More