Integrating Bangla Computing Support in OpenOffice.org

Asif Iqbal Sarkar, Dewan Shahriar Hossain Pavel and Mumit Khan
BRAC University, Dhaka, Bangladesh.
asif@bracuniversity.net, pavel@bracuniversity.net, mumit@bracuniversity.net

Abstract

This paper addresses the issues of Integrating Bangla Computing support for OpenOffice.org office suite and in the process, Identifies and describes the different problems associated with OpenOffice.org and that should be solved in order to get optimum performance in Bangla Script rendering and computing. The paper also discusses the integration and methodology of a number of small applications that have been developed to work with OpenOffice.org and the problems related to this issue.

1. Introduction

The history of Bangla computing is not very rich. There has not yet been any significant development other than some sporadic effort directed towards Bangla keyboards and some word processors. Our work is to accomplish the goal of developing several small applications for Bangla word processing and OCR and hence integrate them in a complete office suit as separate components, so that anybody who uses the office program can be benefited in his/her Bangla computing activities. Initially, an open source office suit, namely OpenOffice.org [1], has been selected for integration of components like Bangla Spellchecker, Search & replace utility and a simple OCR engine. These small applications are part of almost all popular word processors available at present, but none of them support Bangla scripts and hence cannot perform on Bangla documents. Our applications will help Bangla users to perform spell-checking, searching and replacing and simple OCR operations on Bangla documents using the same Office program. Since these programs are integrated as components into the office program the users only need to have the office program installed in their systems rather than individually in-stalling the components. While working on the applications we have come across the fact that the biggest problem for Bangla computing is co-existence of multiple encoding for Bangla letters and so far no progress towards standardization of character encoding has been made. So, it creates compliance problem among different Bangla software, which basically discourages developers to develop Bangla software. But with the advent of Unicode for Bangla script, this encoding problem can be solved successfully and for this very reason our applications are developed to work with Unicode.

2. Reasons for choosing OpenOffice.org

At the beginning of the project we had to decide which Office suit we would work with. There were several issues we had to keep in mind before making this choice. For the convenience of our project we had to go for an open source office package as oppose to any proprietary Office suit. We had several options like, ABI word of GNOME, K-word of KDE and OpenOffice.org. ABI doesn't provide support for Bangla script. K-word uses QT rendering engine, which is under research. OpenOffice.org is a huge project conducted by Sun Micro systems and it comes with Redhat Linux as a free Office program. There is a project called Localization of OpenOffice.org, which is conducted by several voluntary teams working all over the world. So it was easy for us to get hold of a lot of information regarding integration of components into OpenOffice.org before we could start the project. In this paper we have not discussed about localizing OpenOffice.org rather Bangla-computing integration into OpenOffice.org. OpenOffice.org provides flexibility for which anyone can add components to it through component based API called UNO (Universal Network Object).

3. Connection of client program to OpenOffice.org through UNO

Universal Network Objects or UNO is the base component technology for OpenOffice that is designed to write client components that interact with OpenOffice.org across languages, platforms and networks. [2] UNO is currently supported on Linux, Solaris, and Windows for the Java and C++ programming languages. It is quite reasonable to write components using UNO that extend the functionality

of OpenOffice.org without changing the base system at all. In fact, one can write complete applications that use OpenOffice.org as the backend application, or use OpenOffice.org services and relay that to the user.

4. Remote connectivity to OpenOffice.org

Since UNO allows clients to remotely interact with OpenOffice.org, this section looks at how the remote connectivity is established. Since remote clients communicate over a TCP/IP socket, OpenOffice.org must be told to listen for incoming TCP/IP connections on a given port. It can be done on each invocation of OpenOffice.org, or made effective for all future invocations. The process for each is shown below; see [1] and [2] for details on the configuration changes required. To make OpenOffice.org listen for network connections on all future invocations, first open the file <OfficePath>/share/registry/data/org/OpenOffice.org/Setup.scu and look for the element *Office*. Add the appropriate host and port parameters to its *ooSetupConnectionURL* property. An alternative is to add "*accept=socket,port=8100;urp"* to the command line when invoking OpenOffice.org, telling it to listen on the given port for incoming connections. This is of course only effective for the current session, and does not affect other running instances, if any.

5. Client program for connectivity

The following code, taken verbatim from [3], shows how to write a simple client that connects to OpenOffice.org remotely..

```
import com.sun.star.bridge.XUnoUrlResolver;
improt com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XComponentContext;
import
com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.beans.XPropertySet;

public class FirstConnection extends
java.lang.Object{
private XcomponentContext xRemoteContext =
null;
private XmultiComponentFactory
xRemoteServiceManager = null;

public static void main (String[]args) {
FirstConnection firstConnection1 = new
FirstConnection();
try{
   firstConnection1.useConnection();
}
catch (java.lang.Exception e){
e.printStackTrace();
}
finally {
System.exit(0);
}
}

protected void useConnection() throws
java.lang.Exception {

try{
 xRemoteServiceManager =
this.getRemoteServiceManager(
     "uno:socket,
host=localhost,port=8100;urp;
StarOffice.ServiceManager");
String avaialable =(null
!=xRemoteSserviceManager ?"available" : "not
available");
System.out.println("remote ServiceManager is
"+ available);
//
/// do something with the service manager
....
///
}
catch
(com.sun.star.connection.NoConnectException
e) {
System.err.println("No process listening on
the resource");
e.printStackTrace();
throw e;
}
catch (com.sun.star.lang.DisposedException e)
{
xRemoteContext = null;
throw e;
}
}

protected XmultiComponentFactory
getRemoteServiceManager (String unoUrl)
throws java.lang.Exception {

if (xRemoteContext = null) {
//  First setp: create local component
context, get local servicemanager and
// ask it to create a UnoUrlResolver object
with an XunoUrlResolver interface
XcomponentContext
xLocalContext=com.sun.star.com.helper.Bootstr
ap.createInitialComponentContext(null);

XmultiComponentFactory xLocalServiceManager =
xLocalContext.getServiceManager();
Object urlResolver =
xLocalServiceManager.createInstanceWithContex
t("com.sun.star.bridge.UnoUrlResolver",
xLocalContext);

XUnoUrlResolver xUnoUrlResolver =
(XUnoUrlResolver)
UnoRuntime.queryInterface(XUnoUrlResolver.cla
ss, urlResolver);

Object initialObject =
xUnoUrlResolver.resolve(unoUrl);
XPropertySet xPropertySet = (XPropertySet)
UnoRuntime.queryInterfac( XpropertySet.class,
initialObject);
Object context =
xPropertySet.getPropertyValue("DefaultContext
");
xRemoteContext = (XComponentContext)
UnoRuntime.queryInterfac(XComponentContext.cl
ass, context);
```

```
}
return xRemoteContext.getServiceManager();
}
}
```

6. Integration of search & replace client program

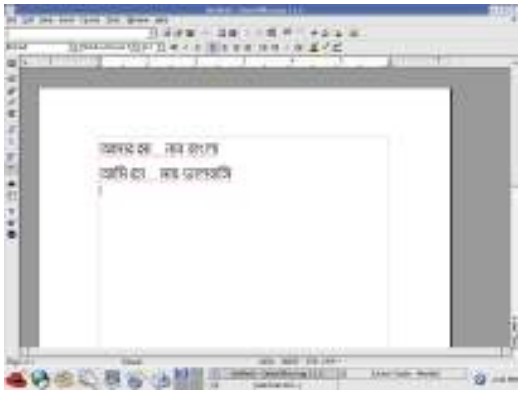

Figure 1: Bangla document before replacement

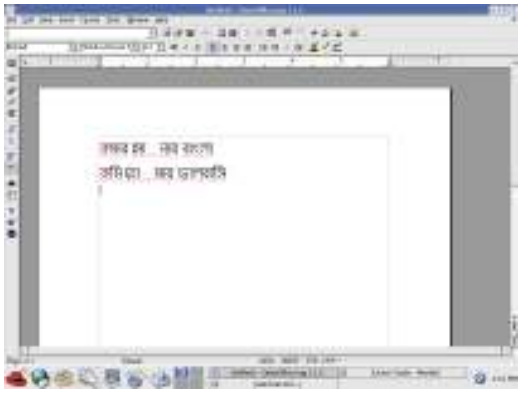

Figure 2: Bangla document after replacement

OpenOffice has its own search and replace utility but it has some problems especially regarding handling of Bangla scripts. Its search and replace dialog box can not display Bangla letters even though it can accept Unicode code points of Bangla letters and do rest of the task accordingly. But from the user's perspective it is a major concern to let the user view what he/she types in. So after considering this issue a component has been developed and can be added to OpenOffice.org through remote connection to the OpenOffice.org for search and replace for Bangla.

7. Integration of Bangla Spellchecker program

Figure 3: Bangla Spellchecker in editor

Figure 4: Checking and replacement of misspelled word

OpenOffice doesn't provide dictionary for Bangla. And it has the similar problem as we discussed before for search and replace dialog box. So it has been felt to develop a spell checker for Bangla. And like the previous one it is also a client program written in java and can be connected to OpenOffice.org remotely and can check spelling of any Bangla word document loaded in OpenOffice.org. The spell checker maintains a Bangla lexicon of about more than 50,000 words. It searches the lexicon for match of each word in the document for spellchecking decisions. If wrong word is typed the speller identifies it highlights the wrong word. The user has the option to replace it with the right word or ignore it when the spellchecker program is invoked.

8. Integration of Bangla OCR client program

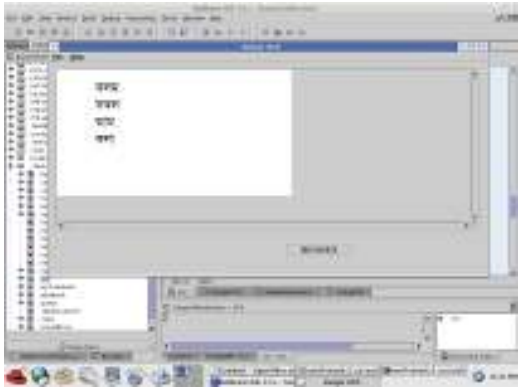

Figure 5: Bangla OCR interface – loading image containing Bangla words

Figure 6: Bangla OCR – Producing the words in an editor

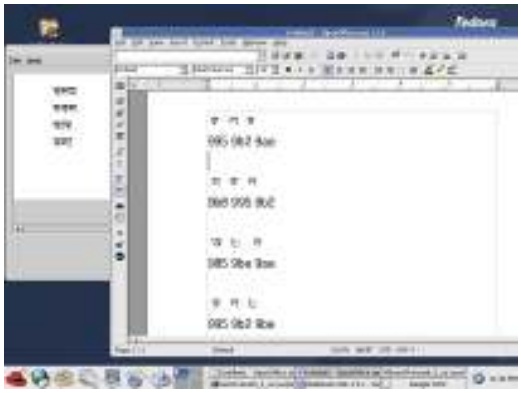

Figure 7: Comparison of the image and document after OCR program function

It is a very good demonstration for convincing ourselves about how useful it is to develop a component using UNO. Bangla OCR driver has been developed in our team and with the help of that driver a scanned image of Bangla document has been read and produced as text in a Bangla document which can be displayed and edited in OpenOffice.org through the component developed using UNO in Java. The computer that is plugged to scanner doesn't need have OpenOffice.org installed but still capable of displaying the document by remotely connecting to OpenOffice.org resided in another computer.

9. Problems associated with OpenOffice.org

The main problem associated with OpenOffice.org is its rendering engine's capability of displaying Bangla letters. There is another problem that limits the Bangla text processing in OpenOffice.org, is its unsynchronized forward and backward movement of cursor. For this problem, the replacement, deletion operations are hampered.

10. Acknowledgement

This work has been partially supported through PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada, administered through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

11. Conclusion

Our aim is to work further for integrating several other small applications and tools and to eradicate the persisting problems in Openoffice.org. We hope our work will lead the way in localization efforts of Office programs to work for Bangla.

12. References

[1] OpenOffice, available online at www.openoffice.org

[2] OpenOffice Java UNO Reference, available online at http://api.openoffice.org/docs/java/ref/overview-summary.html