DEVELOPMENT OF FILTERS FOR TRANSCRIPTION BETWEEN ENGLISH AND BENGALI DOCUMENTS

A Project Report
Submitted in Partial Fulfillment of the Requirements
For the Award of Degree
of
BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND TECHNOLOGY

By

SUSHMITA SEN (Exam Roll-111105027)
&
SURAJEET BHARATI (Exam Roll-111105044)

Under the Guidance of

Prof. MANAS HIRA

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY
INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR
HOWRAH - 711103.

MAY 2015

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

Indian Institute of Engineering Science & Technology, Shibpur

Certificate

This is to certify that the project entitled, **“Development of Filters for Transcription Between English and Bengali Documents”** submitted by **Sushmita Sen** (Exam Roll-111105027) and **Surajeet Bharati** (Exam Roll-111105044) in fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science & Technology at the Indian Institute of Engineering Science and Technology, Shibpur, is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University/Institute for the award of any Degree or Diploma.

(Manas Hira)
Project Guide

Acknowledgments

After finishing writing the Progress report, it is time now to thank all the people who have helped us through out the process, which makes this thesis possible for us.

First of all, we wish to express our sincere gratitude to our guide, Mr. Manas Hira, Professor of the Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, for providing us such a great opportunity to work with him. We couldn't thank him enough for his knowledgeable guidance, valuable suggestions and continuous encouragement. We are deeply grateful to the efforts, patience and enthusiasm that he shared with us along the entire process.

Many thanks will be paid to all teaching and non-teaching staffs of the Indian Institute of Engineering Science and Technology, Shibpur.

We owe to the books, websites and other references wherefrom we have borrowed many of our ideas during project work.

Last but not the least, our sincere gratefulness goes to our dearest friends, who were so helpful in every single aspect of our life at Indian Institute of Engineering Science and Technology, Shibpur.

Sushmita Sen
Date :

Surajeet Bharati
Date :

Contents

List of Figures

Chapter 1

Introduction

In the present scenario of globalization, ethnic populations are no more localized to particular geographic area in the world. There are a lot of people who are not able to stick to their homeland or to some particular place all over his life. Consequently, it is becoming increasingly difficult for such people to remain close to the original culture of the ethnic group to which they belong. For example, it is almost impossible for a Bengali person staying outside the province of West Bengal or the country of Bangladesh to go through a formal schooling for reading or writing Bengali scripts. He or she, therefore, may never have the opportunity to get a direct exposure to Bengali literature. He or she however likely to speak in Bengali when back to his or her own people at his or her residence.If he/she has the Bengali literary works transcribed into some other script or if he/she can transcribe his/her own writing to Bengali script, he or she could have had a glimpses over the Bengali literature.

Besides, when most of the concerned people in the present world have at least one smart-phone, tablet or laptop in his hand, they are preferring soft copies more than hard copies for reading purposes. Using hard copies for reading is becoming obsolete day by day. In such cases, they might need some tool to transcribe document into his preferable language script. There comes the importance of having a transcriptor application.

1.1 Describing the work

In the work being proposed, filters are to be designed and implemented for transcription of typed Bengali documents (available in doc, pdf, or HTML formats) into English script and vice-versa. The produced document should be viewable in any web browser and be able to exported to doc or pdf. The transcriptor will be a web application and user will be given the opportunity to transcribe documents using any type of device that has a browser for accessing internet.

1.2 Sectionwise reporting

This section gives a comprehensive preview of the subsequent chapters of this report.

chapter 2 describes the functional requirements of the transcriptor that is which services will be provided to the user by the online transcriptor.

chapter 3 gives details about the platform that are used to develop and execute the transcriptor.

chapter 4 is concentrated on the details of designing the transcriptor. Section 1 of the chapter gives the details about the the data-set i.e. data-table used for the purpose of transcription, section 2 gives an insight on the data-structure alternatives that can be used to store our already designed data-set, section 3 will tell you why does a lexical analyzer suite more than some traditional data-structure with respect to our requirements, where section 4 will tell you about different lexical analyzers that can be used.

chapter 5 will give details of the prerequisites needed for starting the implementation. That includes a section about what web server we have chosen among different alternatives and why we have chosen that and other section tell the same about choosing appropriate development tools to reduce the effort.

chapter 6 covers the whole implementation process. First section gives broad views about the procedure of transcription between different file formats of different script languages. Second Section includes all the UML design diagrams related to our implementation. In the Third section, you can see web-page interfaces that a user will face while using the transcriptor online and other discussions related to implementation of the front-end.

In the last chapter there are three sections which tells about how much of the total requirements we have fulfilled, what we couldn't succeed to implement and what is the future scope of the project one by one.

Chapter 2

Functional Requirement

In this section, we are going to discuss the functional requirements of the online transcriptor that we have developed. A user can get the following services from our developed online tool.

i) As soon as user access the web-page using specified URL, he/she will face a neat and clean interface. In the interface, user will be able to upload a file with extension of HTML, PDF or DOC. He will also be allowed to select the desired output format from a drop down menu which contains three options - HTML,PDF and DOC. User have to tell the system that font of which language he had used in the input file. For this, there will be another drop-down menu containing two options - English and Bengali. Then, clicking on Transcript button will upload the file into server which will be followed by transcription of the input file.

ii) If user selects HTML as output format, then transcribed file will be loaded into user's browser as a normal web-page within a few moments. User can then save the output using *Ctrl+S* or some other key combination depending on the particular browser. For other two output format options, user will get "Download File" prompt in his browser.

iii) Each word written in Bengali fonts will be converted to equivalent English font while keeping the pronunciation unchanged and vice-versa. The word may be within cell of a table,or it may be a heading, customized text segment or normal body content sentences - each of these have to be transcribed.

iv) The Transcriptor tool will not transcribe words that are in a image format though, as it works only on objects where *unicode* encoding is there. For working on such images, further image processing algorithms needed and that isn't in our scope.

v) One of the main requirement of the online transcriptor is that the look and feel of the input file need to be preserved in the transcribed output file. Only words will be transcribed. All the other things e.g. tables, fonts, font colour, font weight, heading, indentation etc. will be same as before.

Chapter 3

Platform

We have developed the Transcriptor using Java. We have worked on Java version 7. In this chapter, we will discuss about the platforms used to develop the tool and platforms that have been used to execute our developed system.

- We have used Java Servlet for executing our Transcription System on server-side. For this purpose, we have chosen Apache Tomcat(v7.0) web-Server that will be active on server-side to service user requests and transcript documents.
- As mentioned above, the online transcription system has been implemented using Java Servlet.For this we have used Eclipse IDE for EE Developers(Eclipse Luna) to implement the Servlet. Reason behind using Eclipse in our case is that the Tomcat Web-server can be integrated with Eclipse EE IDE.
- We have used Tomcat Web-server on Ubuntu 14.04 Operating System.

In each of the above mentioned cases, there were several other alternatives that could have been used to get the target achieved. Comparison among them and reason behind choosing a particular one will be discussed on chapter 5.

Chapter 4

Designing the whole thing

First of all, we have to design the whole application on the basis of which we can implement the transcriptor. For this case, we have to first design the database.

4.1 Designing the Dataset

For transcription we can use a list containing all the possible words and their transcribed forms. Then we can search for match in the input file and replace it with its transcribed form. This is the approach that all the existing dictionary software use. But this approach is highly inefficient. Storing all such words will take huge memory space and searching word from such a huge list will consume resources and it will be also slow.

For speedy transcription of typed Bengali documents to English scripts or vice-versa, we have designed data sets of our own. You can check them in the Figure 4.1 for Bengali Vowels and digits, Figure 4.2 for bengali consonants, Figure 4.3, Figure 4.4, Figure 4.5 and Figure 4.6 for Benglai Consonant clusters (Juktakkhor).

Using these dataset, we can easily transcript bengali script to english script as follows.

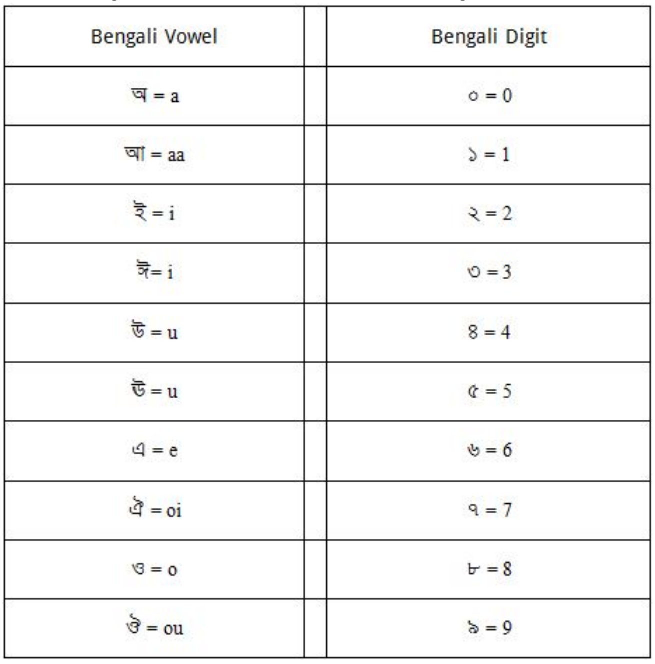
বালক = baalak (ব = b, া = aa, ল = la, ক = k)

অক্ষর = akkhar (অ = a, ক্ষ = kkha, র = r)

লক্ষ্মী = lakkhmi (ল = la, ক্ষ্ম = kkhm, ী = i)

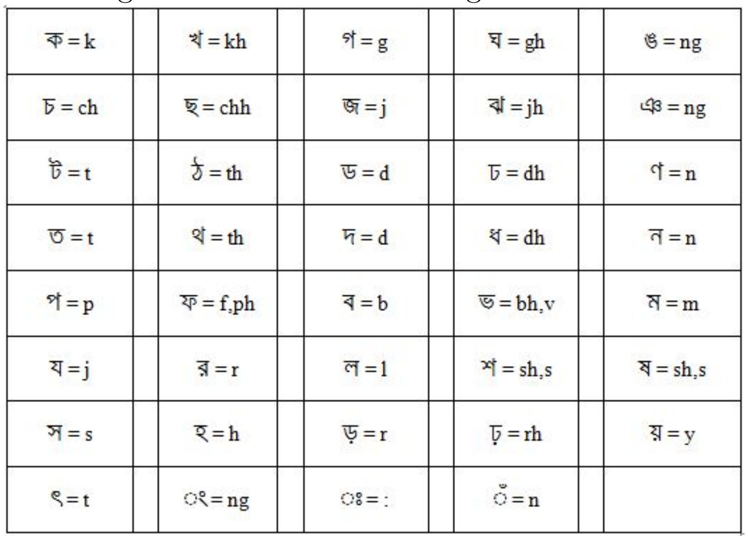
That idea will definitely make transcription easier as we have to go through a limited data set during transcription. For transcription mechanism, we need to scan the input file (available in doc, pdf or HTML format)first. During

Figure 4.1: Dataset for Bengali vowel



Bengali Vowel		Bengali Digit
অ = a		০ = 0
আ = aa		১ = 1
ই = i		২ = 2
ঈ= i		৩ = 3
উ = u		৪ = 4
ঊ = u		৫ = 5
এ = e		৬ = 6
ঐ = oi		৭ = 7
ও = o		৮ = 8
ঔ = ou		৯ = 9

Figure 4.2: Dataset for Bengali Consonant



ক = k		খ = kh		গ = g		ঘ = gh		ঙ = ng
চ = ch		ছ = chh		জ = j		ঝ = jh		ঞ = ng
ট = t		ঠ = th		ড = d		ঢ = dh		ণ = n
ত = t		থ = th		দ = d		ধ = dh		ন = n
প = p		ফ = f,ph		ব = b		ভ = bh,v		ম = m
য = j		র = r		ল = l		শ = sh,s		ষ = sh,s
স = s		হ = h		ড় = r		ঢ় = rh		য় = y
ৎ = t		ং = ng		ঃ = :		ঁ = n		

scanning of each character in a word, we will try to get the longest match from the data set for these characters. Then after getting the match according to the data set, we will write the alternate script into the output file.

In some cases, you can see there are two or more English transcribed form of a single Bengali string in the dataset. Different person can use different English word to write a Bengali word. Such different forms of a Bengali word are listed there. For such English to Bengali transcription, whenever any of these two or more English string will be matched, that will be replaced with the Bengali string. For Bengali to English transcription, whenever the Bengali string is matched, first one among these two or more English strings will be used to replace with.

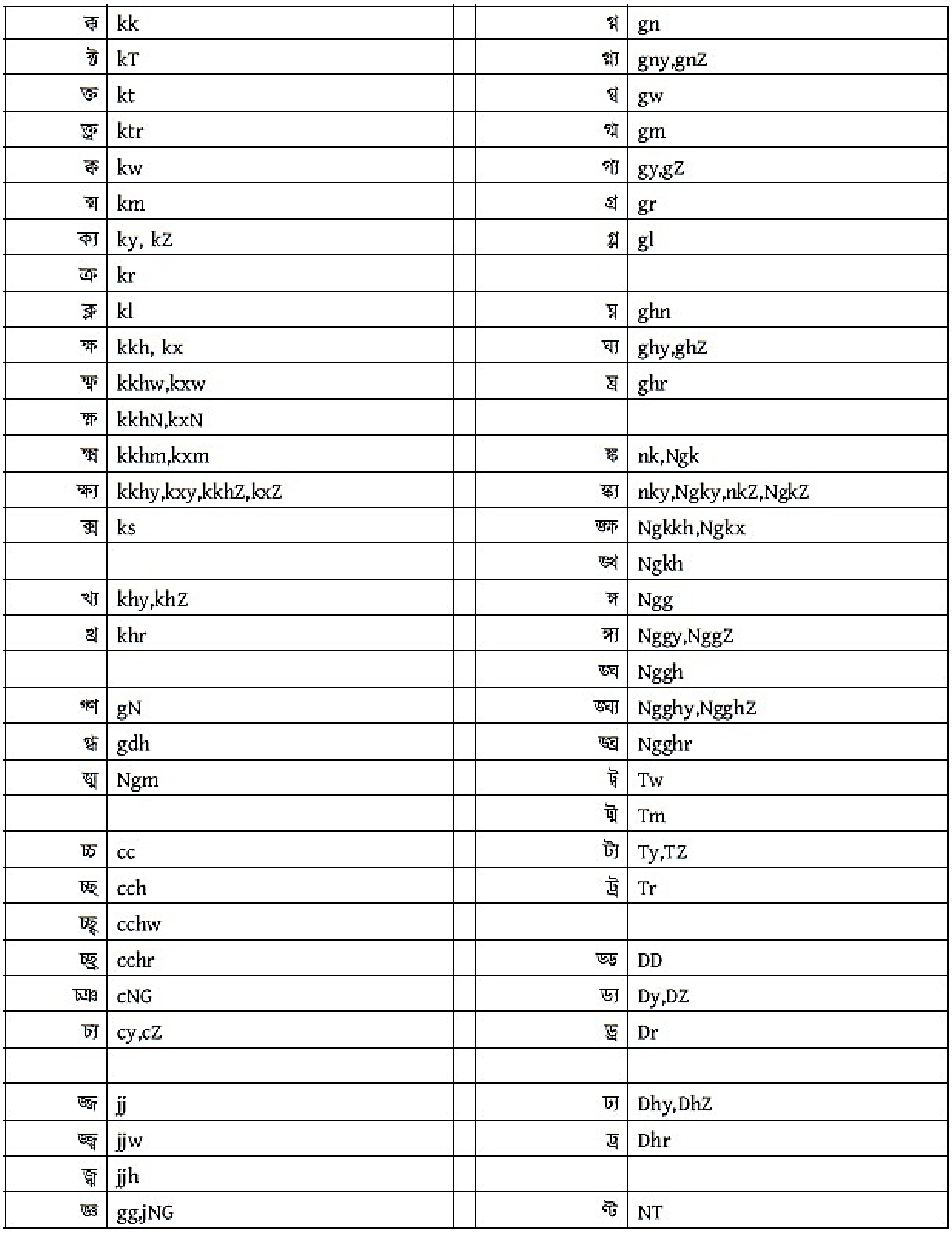
Now a question can come to your mind that why specifically this data set? Actually the already existing data sets(different standards used by different software) have much more conflicts. So after removing all those conflicts we have come to this one and in future we are going to use this data set as standard. Though it doesn't guarantee that it is completely conflict-free but still we have tried to make it conflict-free as much as possible.

4.2 Datastructure Alternatives

For storing the data set (as mentioned in the earlier section), we need a proper data structure. Alternatives for this case are

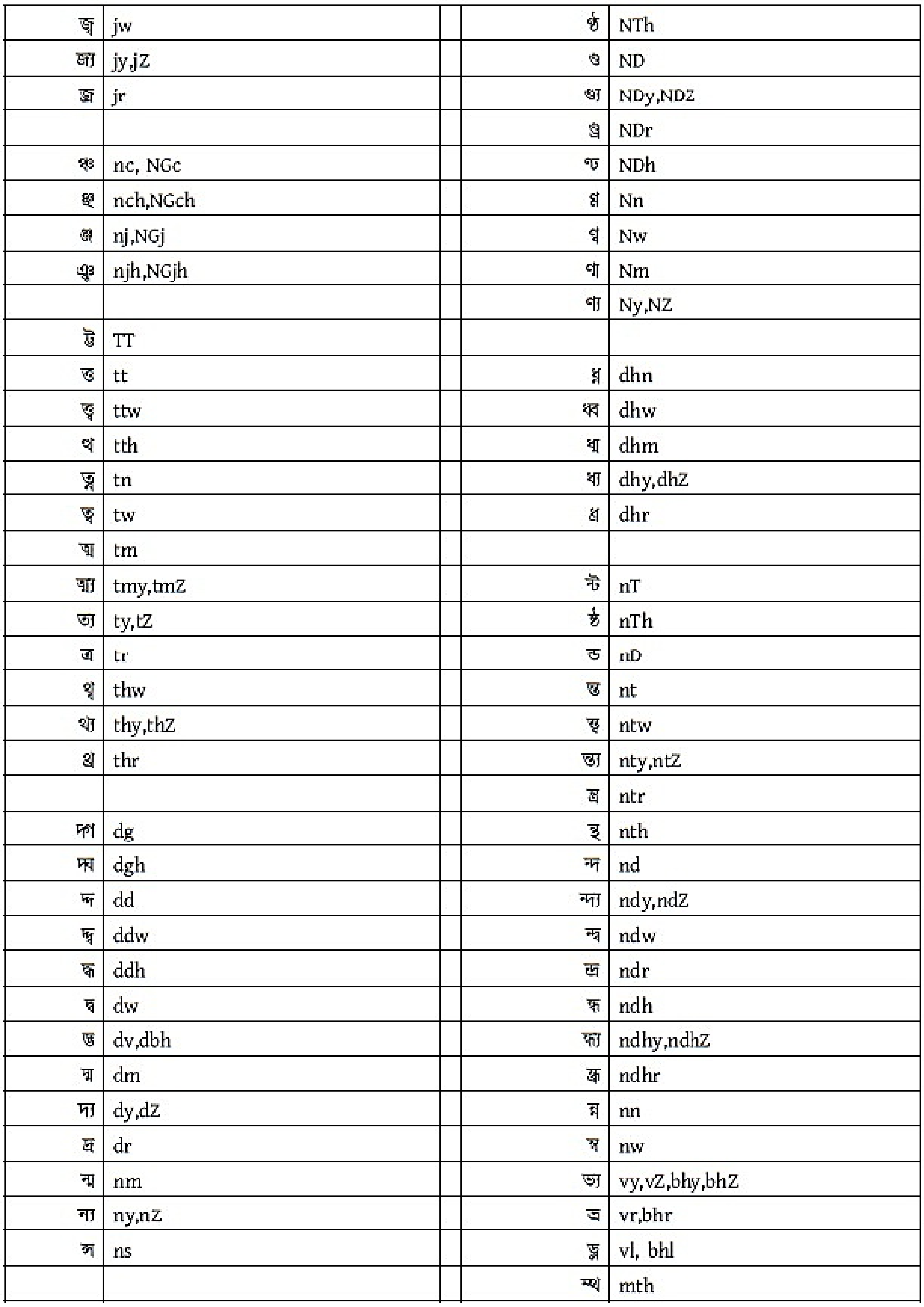
- Dictionary is marked as an obsolete class. So, it should not be used.
- Trie is implemented using tree data structure. It is inefficient in space and time. It should be used when there is an issue of storing a huge dataset with a lot of elements with common prefixes. Our dataset does not follow this characteristic.
- DAWG (Directed Acyclic Word Graph) is used when size of dataset is fixed. Our dataset may be extended dynamically in need.
- HashTable is better when there is a need of multithreading. Possibly, we require single thread in our implementation. So, using hashtable will be just extra overhead. Besides, in hashtable, the hash function has to be implemented by us. So, if we don't implement a good hash function, there will be chance of collision.

Figure 4.3: Juktakkhor DataSet Table No.1



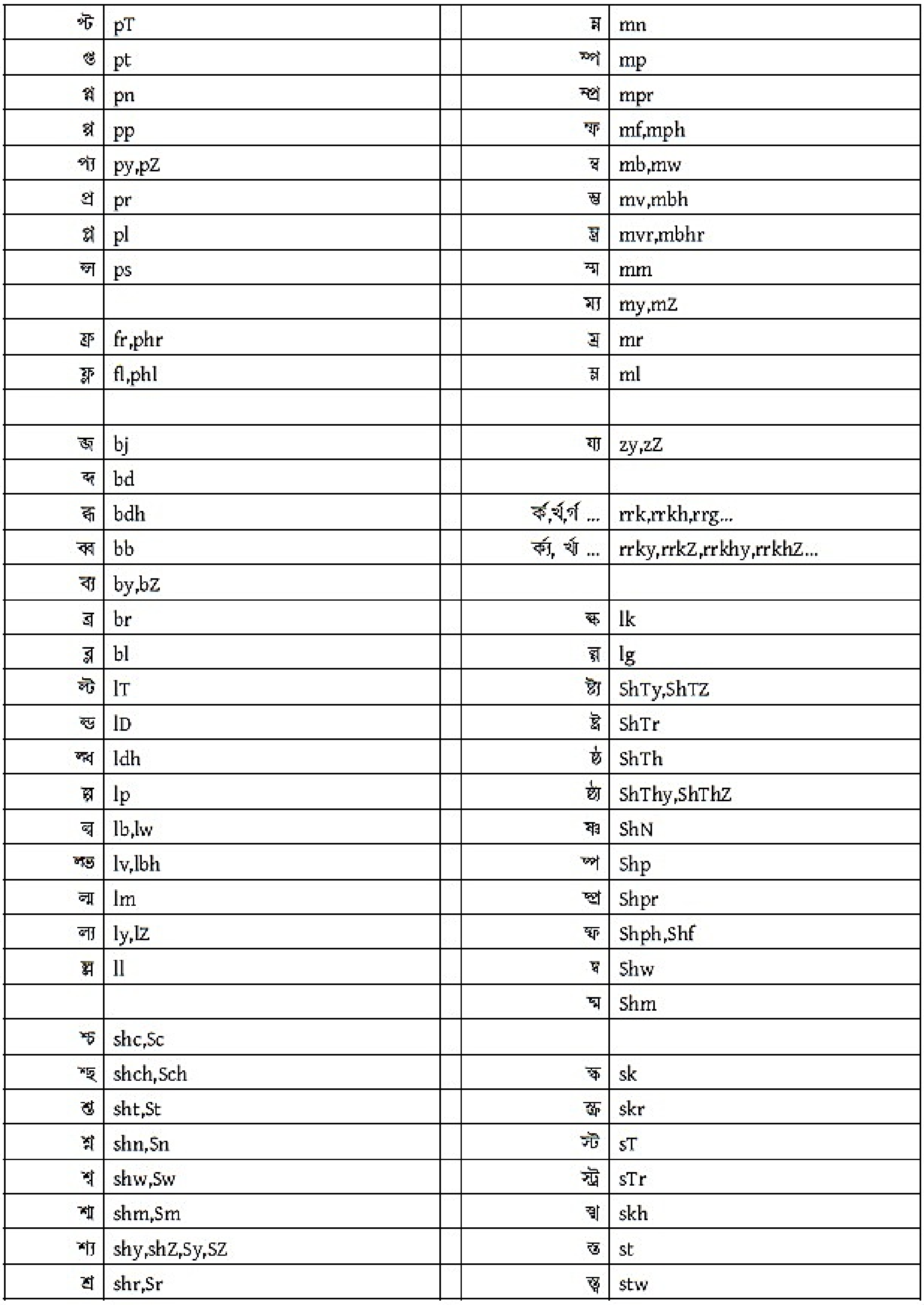
ক্ক	kk		গ্ন	gn
ক্ট	kT		গ্ন্য	gny,gnZ
ক্ত	kt		গ্ব	gw
ক্ত্র	ktr		গ্ম	gm
ক্ব	kw		গ্য	gy,gZ
ক্ম	km		গ্র	gr
ক্য	ky, kZ		গ্ল	gl
ক্র	kr			
ক্ল	kl		ঘ্ন	ghn
ক্ষ	kkh, kx		ঘ্য	ghy,ghZ
ক্ষ্ব	kkhw,kxw		ঘ্র	ghr
ক্ষ্ণ	kkhN,kxN			
ক্ষ্ম	kkhm,kxm		ঙ্ক	nk,Ngk
ক্ষ্য	kkhy,kxy,kkhZ,kxZ		ঙ্ক্য	nky,Ngky,nkZ,NgkZ
ক্স	ks		ঙ্ক্ষ	Ngkkh,Ngkx
			ঙ্খ	Ngkh
খ্য	khy,khZ		ঙ্গ	Ngg
খ্র	khr		ঙ্গ্য	Nggy,NggZ
			ঙ্ঘ	Nggh
গ্ণ	gN		ঙ্ঘ্য	Ngghy,NgghZ
গ্ধ	gdh		ঙ্ঘ্র	Ngghr
ঙ্ম	Ngm		ট্ব	Tw
			ট্ম	Tm
চ্চ	cc		ট্য	Ty,TZ
চ্ছ	cch		ট্র	Tr
চ্ছ্ব	cchw			
চ্ছ্র	cchr		ড্ড	DD
চ্ঞ	cNG		ড্য	Dy,DZ
চ্য	cy,cZ		ড্র	Dr
জ্জ	jj		ঢ্য	Dhy,DhZ
জ্জ্ব	jjw		ঢ্র	Dhr
জ্ঝ	jjh			
জ্ঞ	gg,jNG		ণ্ট	NT

Figure 4.4: Juktakkhor DataSet Table No.2



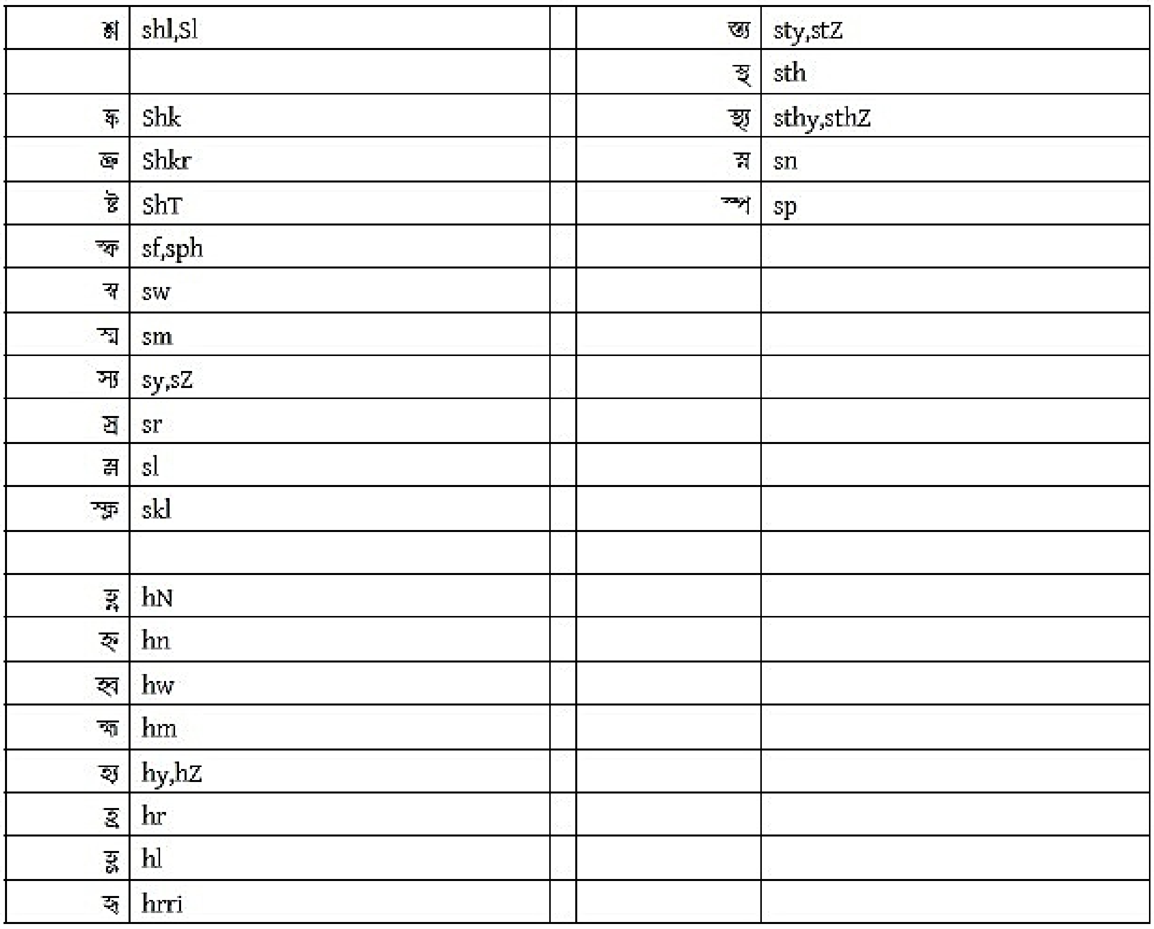
জ্ব	jw		ণ্ঠ	NTh
জ্য	jy,jZ		ণ্ড	ND
জ্র	jr		ণ্ড্য	NDy,NDZ
			ণ্ড্র	NDr
ঞ্চ	nc, NGc		ণ্ঢ	NDh
ঞ্ছ	nch,NGch		ণ্ণ	Nn
ঞ্জ	nj,NGj		ণ্ব	Nw
ঞ্ঝ	njh,NGjh		ণ্ম	Nm
			ণ্য	Ny,NZ
ট্ট	TT			
ত্ত	tt		ধ্ন	dhn
ত্ত্ব	ttw		ধ্ব	dhw
ত্থ	tth		ধ্ম	dhm
ত্ন	tn		ধ্য	dhy,dhZ
ত্ব	tw		ধ্র	dhr
ত্ম	tm			
ত্ম্য	tmy,tmZ		ন্ট	nT
ত্য	ty,tZ		ন্ঠ	nTh
ত্র	tr		ন্ড	nD
থ্ব	thw		ন্ত	nt
থ্য	thy,thZ		ন্ত্ব	ntw
থ্র	thr		ন্ত্য	nty,ntZ
			ন্ত্র	ntr
দ্গ	dg		ন্থ	nth
দ্ঘ	dgh		ন্দ	nd
দ্দ	dd		ন্দ্য	ndy,ndZ
দ্দ্ব	ddw		ন্দ্ব	ndw
দ্ধ	ddh		ন্দ্র	ndr
দ্ব	dw		ন্ধ	ndh
দ্ভ	dv,dbh		ন্ধ্য	ndhy,ndhZ
দ্ম	dm		ন্ধ্র	ndhr
দ্য	dy,dZ		ন্ন	nn
দ্র	dr		ন্ব	nw
ন্ম	nm		ভ্য	vy,vZ,bhy,bhZ
ন্য	ny,nZ		ভ্র	vr,bhr
ন্স	ns		ভ্ল	vl, bhl
			ম্থ	mth

Figure 4.5: Juktakkhor DataSet Table No.3



প্ট	pT		ম্ন	mn
প্ত	pt		ম্প	mp
প্ন	pn		ম্প্র	mpr
প্প	pp		ম্ফ	mf,mph
প্য	py,pZ		ম্ব	mb,mw
প্র	pr		ম্ভ	mv,mbh
প্ল	pl		ম্ভ্র	mvr,mbhr
প্স	ps		ম্ম	mm
			ম্য	my,mZ
ফ্র	fr,phr		ম্র	mr
ফ্ল	fl,phl		ম্ল	ml
ব্জ	bj		য্য	zy,zZ
ব্দ	bd			
ব্ধ	bdh		র্ক,র্খ,র্গ ...	rrk,rrkh,rrg...
ব্ব	bb		র্ক্য, র্খ্য ...	rrky,rrkZ,rrkhy,rrkhZ...
ব্য	by,bZ			
ব্র	br		ল্ক	lk
ব্ল	bl		ল্গ	lg
ল্ট	lT		ষ্ট্য	ShTy,ShTZ
ল্ড	lD		ষ্ট্র	ShTr
ল্ধ	ldh		ষ্ঠ	ShTh
ল্প	lp		ষ্ঠ্য	ShThy,ShThZ
ল্ব	lb,lw		ষ্ণ	ShN
ল্ভ	lv,lbh		ষ্প	Shp
ল্ম	lm		ষ্প্র	Shpr
ল্য	ly,lZ		ষ্ফ	Shph,Shf
ল্ল	ll		ষ্ব	Shw
			ষ্ম	Shm
শ্চ	shc,Sc			
শ্ছ	shch,Sch		স্ক	sk
শ্ত	sht,St		স্ক্র	skr
শ্ন	shn,Sn		স্ট	sT
শ্ব	shw,Sw		স্ট্র	sTr
শ্ম	shm,Sm		স্খ	skh
শ্য	shy,shZ,Sy,SZ		স্ত	st
শ্র	shr,Sr		স্ত্ব	stw

Figure 4.6: Juktakkhor DataSet Table No.4



শ্ল	shl,Sl		স্ত্য	sty,stZ
			স্থ	sth
ষ্ক	Shk		স্থ্য	sthy,sthZ
ষ্ক্র	Shkr		স্ন	sn
ষ্ট	ShT		স্প	sp
স্ফ	sf,sph			
স্ব	sw			
স্ম	sm			
স্য	sy,sZ			
স্র	sr			
স্ল	sl			
স্ক্ল	skl			
হ্ণ	hN			
হ্ন	hn			
হ্ব	hw			
হ্ম	hm			
হ্য	hy,hZ			
হ্র	hr			
হ্ল	hl			
হৃ	hrri			

- Hashmap supports single thread and good hash functions are already implemented in this class. It's faster than the other alternatives and suitable for our design goal.

For storing and manipulating groups of objects, hashmap is highly efficient. Hashmap stores key/value pairs in a hash table. When using a Hashmap, we specify an object that is used as a key, and the value that we want is linked to that key. The key is then hashed, and the resulting hash code is used as the index at which the value is stored within the table. Basically depending upon the matching of longest pattern, we will use this pattern as key and the value will be the alternate script. Here during Bengali to English transcription, key will be Bengali vowel, consonant, digit or complex letters and value will be the English script but for English to Bengali transcription, key will be English pattern and value will be Bengali vowel, consonant, digit or complex letters. As in hashmap, key is hashed and accordingly the value is stored in the table, so we need two different tables for two types of transcription. As using two table will result little bit of space-inefficient (not too much though as our data-set is not such huge), we have tried to implement in such a way that only one table will be in RAM at a time for a particular transcription. For example, during Bengali to English transcription, we will keep that table in memory where key is Bengali pattern and another table will be in disk.

4.3 Choosing Lexical Analyzer instead of Datastructure

Though Hashmap is most suitable among all the alternatives of data structures in case of our design, it will be till an unnecessary use of very heavy thing in comparably smaller issue. It will just result to extra overhead. The data set that we will use for transcription is not so large and Hashmap is not needed to handle it. So, it will be better if we use some lexical analyzer for searching longest match from the input file and replacing it by it's transcribed form instead of using Hashmap datastructure. Some lexical analyzer generator has to be used to generate the analyzer.

Using Lexical Analyzer will help us in two significant ways -

1. We don't have to implement any search algorithm or any datastructure for searching purpose. Instead of a lot of lines of Java code, we have to write very easy and small amount lexical analyzer code.

2. It implements the analyzer following some fine automata. So, it will be faster in operation than Hashmap datastructure.

4.4 Lexical Analyzer Alternatives

There are different lexical analyzer generator available that we can use :

- Lex is a lexical analyzer generator for generating analyzer in C language.
- JFLex and JLex are most popular lexical analyzer generator for generating analyzer in Java.

As we need to implement the whole design in Java, we can use JFlex or JLex. Between these two JFlex is latest and contains less bug. So, we have used JFlex to accomplish the search and replace function.

Chapter 5

Preparation for implementation

5.1 Choosing Web Server

The transcription filter that we are going to implement will run in a Java enabled web server. There are different Java enabled web servers with different sets of functionalities available which are developed by different software development companies. So, firstly we have to choose appropriate web server as per our requirements. Most popular among the web server alternatives are:

1. Red Hat JBoss (recently renamed to Wildfly)

2. Oracle Glassfish

3. IBM WebSphere

4. Apache Tomcat

JBoss, Web Sphere, Glassfish are full stack JavaEE application server that means they include all the JSRs (Java Specification Requests) that are part of a java EE release and whose implementation will be provided by all Java EE certified application servers. On the other hand, Tomcat is just a Java servlet web container. The functionalities, included into the first three are super-set of functionalities included in Apache Tomcat. These three also causes extra resource overhead and administration of these is also complex. So, in case of developing simple web server applications that don't require full Java EE stack, Tomcat can be used. Java EE APIs like EJB or Spring APIs can be imported within Tomcat to extend its functionality.

Our transcription filter does not require full Java EE stack, it only need native Servlet handling functionalities. There need not be any Security purposes, high resource usage or database transactions. So, we have chosen using Apache TomCat v7.0 with Servlet v3.0.

5.2 Choosing Developement tools

We can use various Java Development Tools (JDT) to simplify the process of developing Java applications (applicable for both Web Application and Standalone java application). They can automate the development process in various ways and decrease the overall effort for developing the application. There are large number of JDTs, available for almost all stages of software development. Some of the mostly used JDTs are mentioned here.

- Integrated Development Environments (IDEs) like Eclipse, NetBeans, Boreland Together, BlueJ are added with various debugging tools and convenient functions that make writing the program and debugging much easier than using command line. We will use Eclipse 4.4 Luna to implement the filter.

- We can use automation tools like Apache Ant or Apache Maven to automate various processes like building, compiling, restarting server etc.

- Documentation creating tools like javadoc can be used to create documentation of our implementation.

- Using Java Mission Control we can control running Java programs and resources used by them.

- Apache Torque is an object-relational mapper using which database script can be generated and executed automatically to control a database. It provides high level of abstraction and hides all database-specific implementation details.

Initially we will start using Eclipse 4.4. It can detect unused or unreached code segments or variables in source codes, build source codes to some specified location. Apache Tomcat 7.0, which we are going to use as web server, can also be integrated with Eclipse. In future, we can use other development tools if required.

Chapter 6

Implementation

6.1 Transcription between different formats

Till now, the online tool can provide the facility to transcript a Bengali script of HTML or PDF format into a English script of HTML, PDF or DOC format and a English script of HTML or PDF format into a Bengali script of HTML, PDF or DOC format. The procedure of those transcription are discussed below.

6.1.1 HTML to HTML

In case of HTML to HTML transform, when we transcribing from Bengali document to English Document, we only search for Bengali longest match (in unicode encoding) in the data-set and replace them with corresponding English form. So, all the HTML tags or HTML related words will be unaffected. That will result to a output file, look and feel of which will be same as input file where Bengali words will be transcribed to English words.

When we are transcribing a English document to a Bengali document, we will exclude all those words which are between HTML tags i.e. between "<" and ">" and search for longest match in the rest to replace it with its Bengali form.

6.1.2 HTML to PDF

In both cases of Bengali to English transcription and English to Bengali transcription, at first input file will be converted to transcribed HTML file in the procedure that are mentioned in above subsection. Then that output HTML file will be converted to PDF format.

Now for this format conversion, we have to remember one thing : we have to preserve the look and feel of the input file after transcription. But, all best open source Java API e.g. *PDFBox*, *iText* etc failed to fulfill this requirement.

Now, we have solved that issue using *wkhtmltopdf.* It uses Web Kit Renderer, the same rendering engine that is used by Google Chrome or Apple Safari to render a HTML page. But the only problem was that this renderer is implemented using C and they does not provide any Java API. This problem remains same for another renderer namely Gecko (used by Mozilla Firefox). *wkhtmltopdf* provides a command line tool to convert HTML document to a PDF document without changing the look. We installed that tool on our Ubuntu Machine and created new Process of *wkhtmltopdf* from our servlet to convert HTML to PDF. This process is using much low RAM space (<1MB), low CPU for creating a PDF and also terminating within a moment every time. So, we think, it will be acceptable to be run on server. The output PDF has exactly same look as the input HTML document.

Some commercial Java API like *NitroPDF*, *PDFCrowd* might be useful though. In fact converted document examples in PDFCrowd Website looks quite promising. We haven't tested those API as we are only dealing with open source API.

6.1.3 HTML to DOC

In this case, input file will be transcribed into a PDF file following abovesaid procedure. Then the the PDF will be converted to DOC file. For this conversion, we have used *Apache PdfBox.*

6.1.4 PDF to Other Formats

We faced several problems when working on this transcription. Here, the problem is again conserving the look and feel of the input document. We can transcribe the PDF document by using three different procedures as follows,

i) We can extract texts from the PDF file using *PDFBox* Java API and then transcribe those text using parser and then converting those transcribed text to a file of any format that user want. But, the problem is when we are extracting the text from PDF, we are losing everything about the look of the PDF (some part like indentation, margin remains same though). So, one of our objective failed. It is quite impossible to carry information about look of the PDF while extracting the text, as a

PDF file have its own objects, reference table, different level of abstraction that are hard to manage through programming. So, we left this idea and started thinking some other ways to fulfill our objective.

ii) The second method that we thought is parsing the PDF file, extracting each word, and replacing the word with its transcribed form. In that case, we do not have to think about the look and feel of the PDF, as we are just replacing the string in the PDF without effecting all the other information. Real challenge here is replacing a word in a PDF file. Though all the APIs like *PDFBox*, *iText*, *PDFLib* demand that they can do it, but they failed in 95 percent cases. After a round of net surfing, we discovered that replacing a word in PDF is also quite complicated task and any type of implementation will fail in maximum cases. We haven't tried any commercial APIs like *AsposePDF* though. May be, they will work well in the case of replacing a word or converting to other format.

iii) Another thought was converting the PDF to equivalent HTML and then work on the HTML. We got a Java API namely *PDF2DOM* for this purpose. But, it failed to render a PDF file to similar HTML file correctly. In the output HTML file, consecutive strings are overlapping on each other making the whole document illegible. There is a Commercial API namely *JPEDAL*, which we haven't tried yet.

So we ultimately decided to just extract text from the PDF and transcribe those text to get atleast the transcribed text from the input file. In this way, the look and feel of the input PDF document got completely lost.

6.1.5 DOC to Other Formats

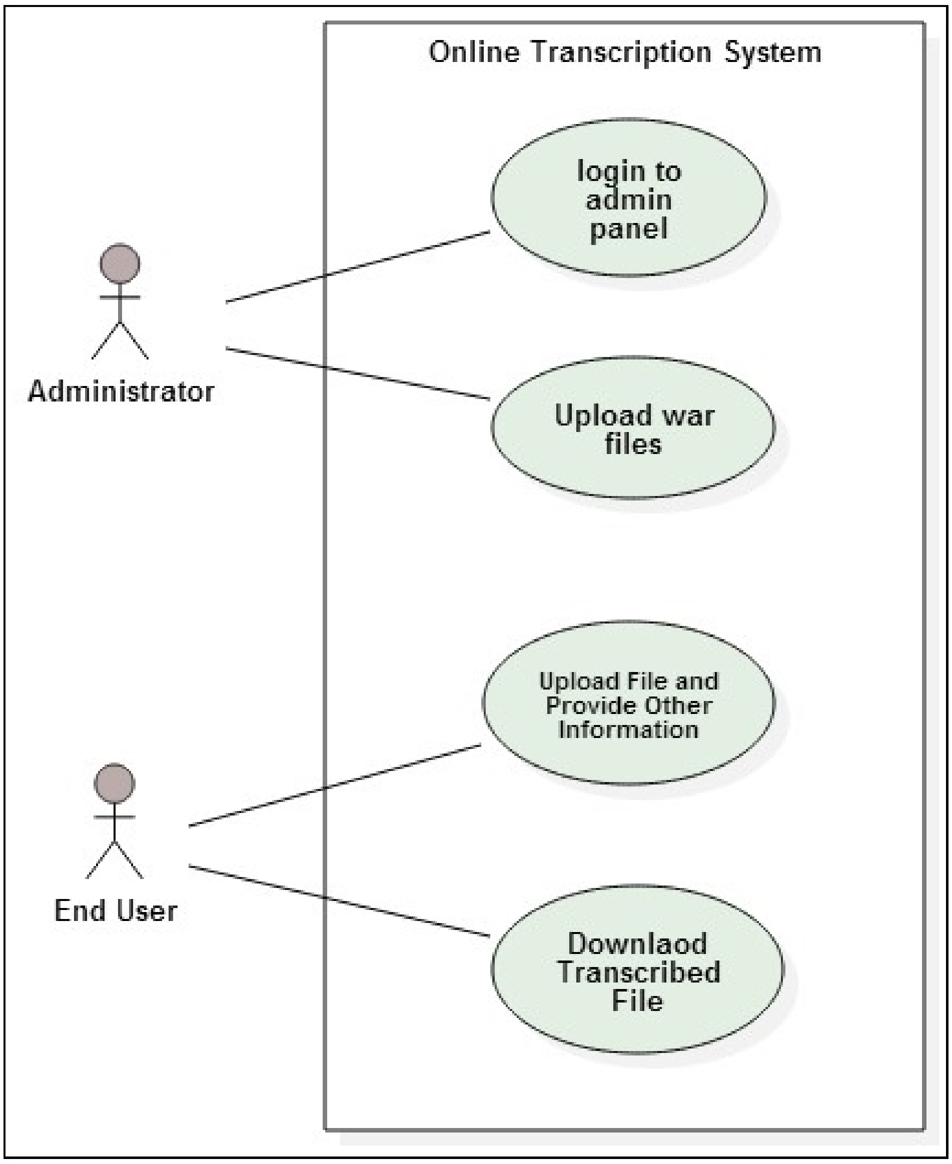
After those work with PDF file, we started working with DOC format, and discovered that internal structure of a DOC file is messier than the PDF file. Sometimes, it differ in its internal structure just because of difference between the Office tools used to write the DOC file. So, even extraction of text from a DOC document fails sometime.

There are some Java API like *Apache POI*, which are used to handle DOC file, replace word in a DOC file, or extract words from a DOC file. But, it is only usable for DOC created by Microsoft Office. We haven't tested this API though.

6.2 UML Diagrams

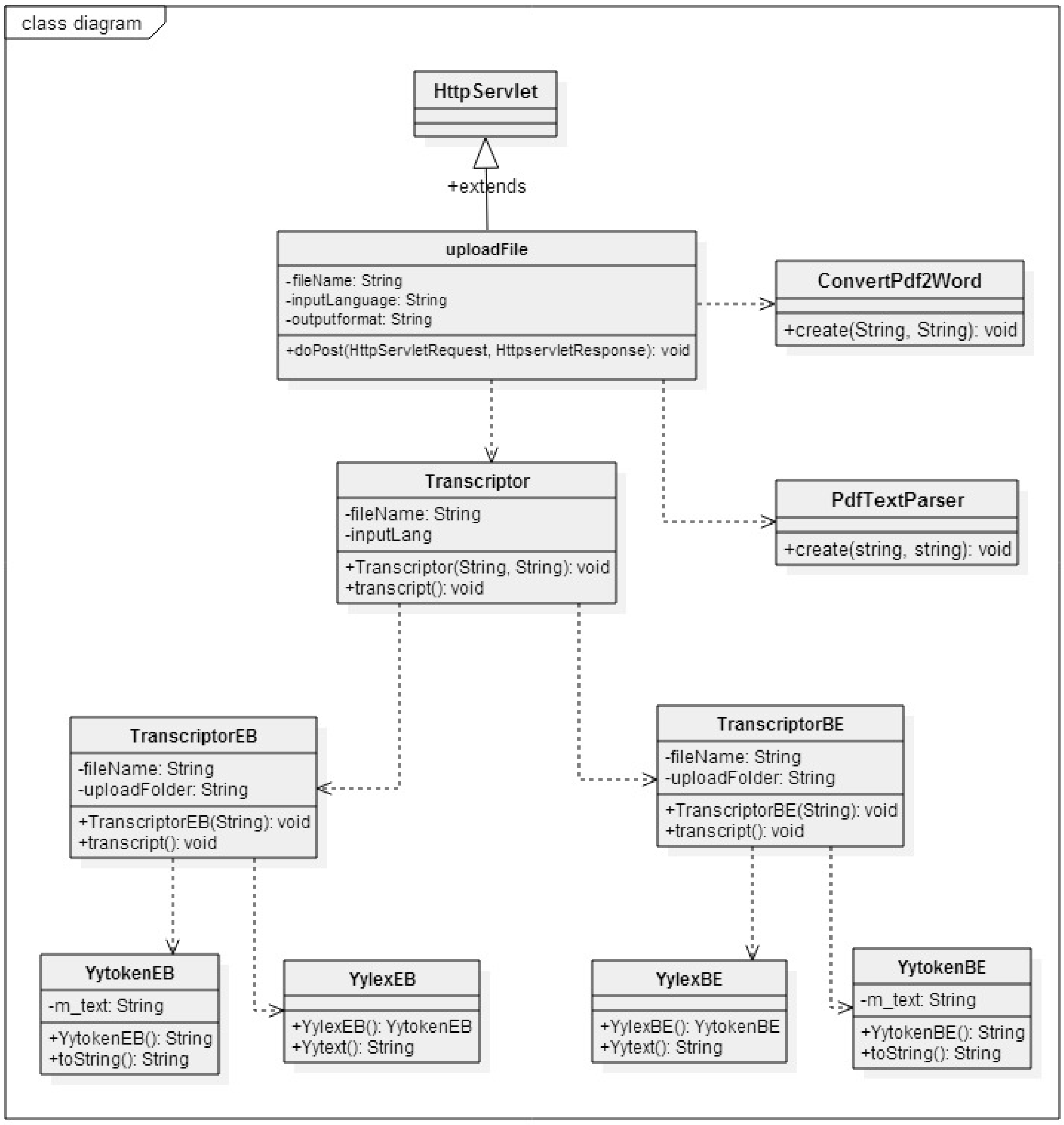
6.2.1 Use Case Diagram

Figure 6.1: Use case Diagram


Online Transcription System
login to admin panel
Administrator
Upload war files
Upload File and Provide Other Information
End User
Downlaod Transcribed File

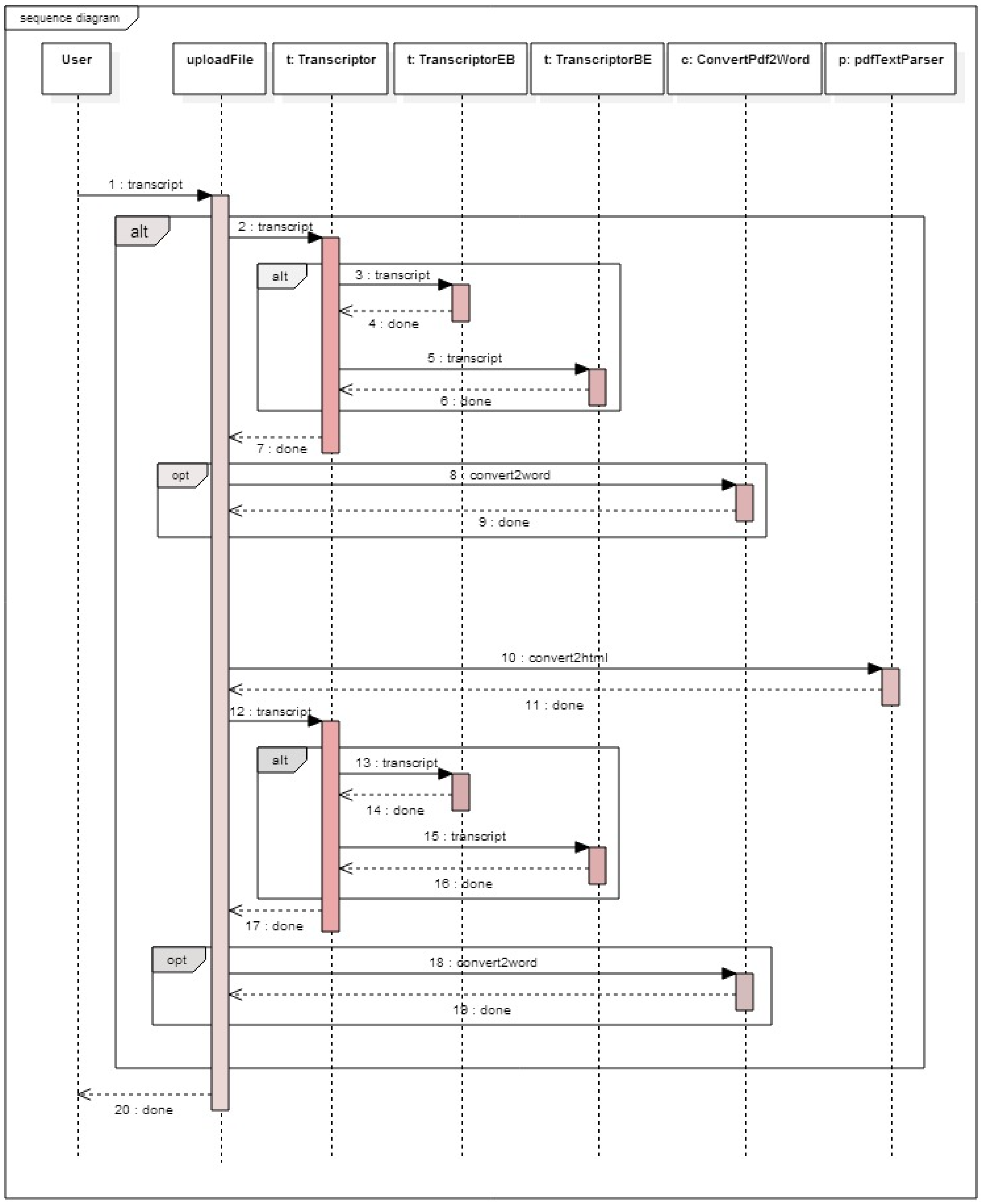
6.2.2 Class Diagram

Figure 6.2: Class Diagram



6.2.3 Sequence Diagram

Figure 6.3: Sequence Diagram



6.3 Designing the GUI

As said before, our transcriptor will be a web application that will run on server-side and user can access it and transcribe his document using a website and corresponding web-pages.

6.3.1 Programming Languages Used

The web-pages should be neat and clean, easily understandable, interactive and easily usable. To design the web-pages we have used the following languages:

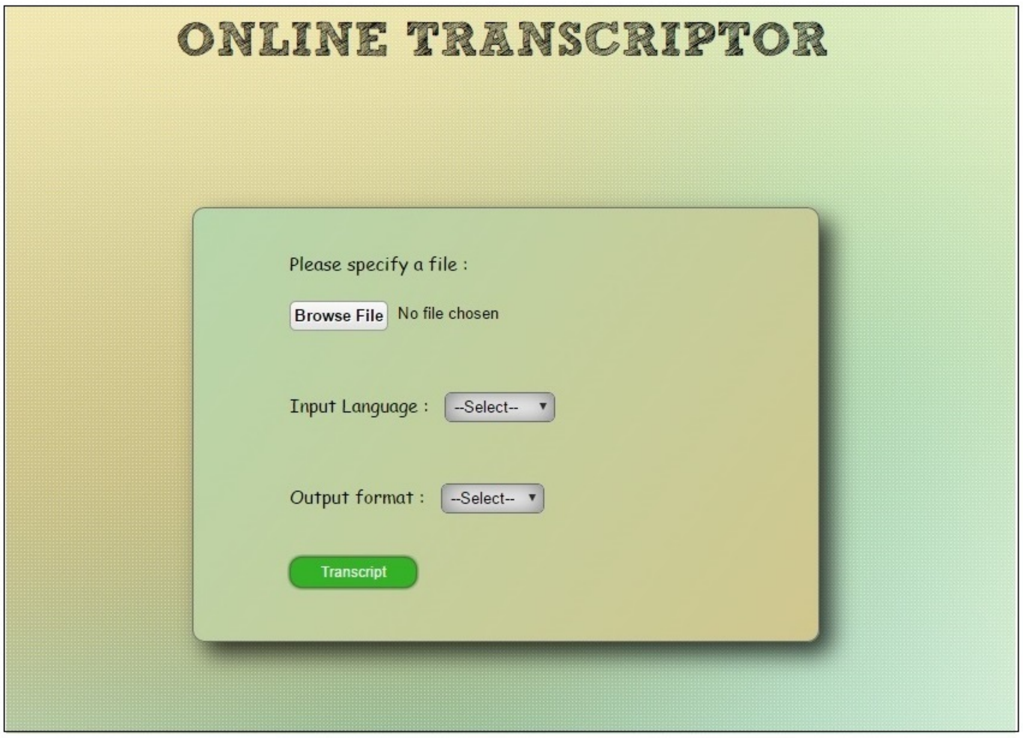
- HTML: We have used HTML5 to develop the skeleton of the web-pages. Using HTML, we have designed file upload button, drop down list, or the submit buttons
- CSS: We have used CSS3 to describe the presentation of Web pages, including colors, layout, and fonts. It has made the web-pages more sophisticated by controlling the look and feel.

6.3.2 Designed Webpages

We have designed only one page as per the requirement. That is the home page that will be faced by the user at the very first. This page can be accessed through *http://purity.becs.ac.in:8080/test-app/*. Take a look on Figure 6.4. It is the home page. In order to transcribe a file, you have to upload the file using "Browse File" button. Then you have to choose your desired output format (HTML, PDF, DOC) from the drop-down menu and have to also tell font of what language used in the input file (English or Bengali). Then you can transcribe the file. If you have chosen the output format as HTML then the output will be shown as an web page that you can save. Otherwise the output will be downloaded.

Here, you can see, you need not to provide the input format and whether it is Bengali script or English script. This information will be detected internally.

Figure 6.4: Home Page Design



Chapter 7

Conclusion and Future Scope

7.1 What we have done

We have completed the whole project and tried to reach all of the requirements as much as possible. We are briefly mentioning what portions of the requirements are completed.

i) We have completed whole of the design phase. We have designed the whole data-set using which transcription have been done. This data-set have been used as standard for the implementation till the end. We have tried to make the data-set free of conflicts as much as possible. Those data-sets have been shown before.

ii) We have chosen appropriate one among various alternative to store this data-set and searching it in a highly efficient way. That have maximized the speed of the process of transcribing a document.

iii) We have chosen java servlet enabled web server and java development tools as per our need so that we can implement the design without any interruption.

iv) We have successfully designed and implemented transcriptor for transcribing typed Bengali documents available in HTML format into English script (exportable to HTML,DOC and PDF format) and vice-versa. But in conversion from PDF to HTML we are losing the look and feel and so in the case of PDF to other formats. In case of DOC format, we are facing the same problem. Ignoring the look and feel,we can successfully transcript the script in any format which is of bit more importance than keeping the look and feel. UNICODE Encoding and JFlex Parser are working well to manage this conversion.

v) We have designed the front page of the website through which user will interact with the online transcriptor. We have tried to make the page easily usable and good to look at.

7.2 Limitation in the design and implementation

We have tried to reach all of the requirements as much as possible. But, still, some part of the requirements seemed impossible or something between extremely hard and impossible. There are some important issues or better to be said as problems that are worth mentionable here as till now we think those problems cannot be solved in our proposed designed approach.

7.2.1 Limitation in data-set

Though we have completed designing the data-set, but it is not completely free of conflicts. We have tried to find and remove those conflicts as much as possible but removing all of those is not possible actually.

Reason of those conflict are some language specific pronunciation anomalies.As a result, some words will be transcribed to some unexpected form. Example of some of such English words are BUS, DO, GATE etc. The transcribed form of such words will loose the original pronunciation. Even if we try to remove this problem as much as possible, it is not possible to remove all such problems with respect to our design approach.

7.2.2 During File Format Conversion

The look and feel of the transcribed document is not same as the original document. For the case of HTML to HTML and PDF conversion, we are successful to make the transcribed form of the document absolutely same as original document but for other cases, it is very hard to make the look same.

From PDF to HTML conversion,we are losing the look and feel of the source or original document. Actually, PDF documents are not supposed to be word-processed. They are designed completely considering the idea of portability so that irrespective of hardware architecture or operating system, a PDF file gives same representation when opened. For this, it internally uses its own data types,data structures,objects,reference table and separate levels of abstraction,design of which also differ from PDF version to version.So, it is very hard to process a PDF file programmatically.

We faced same problem with DOC format, and discovered that internal structure of a doc file is messier than pdf file.Sometimes it differ in its internal structure just because of difference between the office tools used to write doc file, So, even extraction of text from a doc fails sometimes.

7.3 Future Scope

In this section we can discuss how the quality of the transcriptor can be improved by extending the existing design approach that is made by us and already discussed.

The limitation in data-set that is discussed in last section can be resolved by integrating a list of such common words. Input file will be parsed for searching those words and replacing them with their correct transcribed form that are also available in the same list. Other words will be dealt as usual. Though this brute force approach will be highly inefficient, but it will decrease the rate of such errors when as well as decreasing the speed of the process of transcription.

To resolve the issues regarding file format conversion, some commercial Java API can be bought and tried though they may also fail.

Our designed transcriptor can be extended to some more sophisticated form. There are many people who can speak in a language but can't write in that language. Using our design approach some tool can be made for them in future such that they can write words of that language in alphabet of a language that is known to him and then converting that written document to that other language script.

Appendices

Appendix A

Installation and User Manual

In this chapter, we have briefly discussed the installation procedure of the online transcriptor system that we have developed.

Firstly, *tomcat7* server have to be installed and configured in a computer where *ubuntu* OS is used following the steps discussed below.

i) First of all, *tomcat7* server have to be installed. That can be done on a *ubuntu* machine using the following terminal commands.

```
sudo apt−get update
sudo apt−get install tomcat7
```

This command will automatically install *openjdk7*, if it is not already installed as *tomcat7* need *Java* to operate.

ii) *tomcat7-admin* have to be installed to be able to upload *war* files remotely into the *tomcat server*. To install *tomcat7-admin*, one can use following command.

```
sudo apt−get install tomcat7−admin
```

iii) To work with *tomcat7-admin*, it have to be configured firstly. A login have to be added to the *tomcat* server. This can be done by editing the *tomcat-users.xml* file as follows.

```
sudo gedit /etc/tomcat7/tomcat−users.xml
```

This file is filled with comments which describe how to configure the file. You may want to delete all the comments between the following two lines, or you may leave them if you want to reference the examples:

```
<tomcat−users>
</tomcat−users>
```

You will want to add a user who can access the *manager-gui* and *admin-gui* (the management interface that we installed in Step Two). You can do so by defining a user similar to the example below. Be sure to change the password and username if you wish:

```
<tomcat-users>
    <user username="admin" password="password"
      roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Then the *tomcat-users.xml* file have to be saved and quit.

iv) Next, we have to start *tomcat7* service to work with it. We can use following command

```
sudo service tomcat7 start
```

We can stop or restart *tomcat7* when needed using following two commands respectively.

```
sudo service tomcat7 stop
sudo service tomcat7 restart
```

v) After starting *tomcat7* service, we have to upload the *war* file that is provided with the source code DVD. For this purpose, we have to open the admin-panel.We can access admin panel using the following url

```
http://your_ip_address:port/manager/html
```

You will be asked to enter *username* and *password.* Once you give it correctly, you will be redirected to admin panel. In the admin panel, you will get button to upload and deploy *war* file.

vi) After deploying the *war* file, your can access the online transcriptor. If the *war* file name is *test-app.war*, then you can access the system using the following url

```
http://your_ip_address:port/test-app
```

vii) You can change source code when needed using some IDE, create new *war* file and deploy the *war* file in the *admin-panel* to get the changed behavior.

Appendix B

References

1. Head First Java by *Bert Bates and Kathy Sierra*
2. Java The Complete Reference - 8th Edition by *Herbert Schildt*
3. Head First Servlets and JSP by *Bert Bates and Kathy Sierra*
4. Java Servlet Programming, 2nd Edition by *Jason Hunter, William Crawford*
5. lex & yacc, 2nd Edition by *Doug Brown, John Levine, Tony Mason*
6. www.coreservlets.com
7. https://tomcat.apache.org/
8. www.stackoverflow.com
9. www.wkhtmltopdf.org
10. www.eclipse.org
11. http://jflex.de/manual.html

Appendix C

Source Code(DVD)