Natural Language Processing for Urdu TTS System

Hasan Kabir, Syed Raza Shahid, Abdul Mannan Saleem and Sarmad Hussain
Centre for Research in Urdu Language Processing
National University of Computer and Emerging Sciences, Lahore,
Emails: hasankabir@hotmail.com, theraza777@yahoo.com, amannansaleem@yahoo.com,
sarmad.hussain@nu.edu.pk

Abstract: *Natural Language Processing (NLP) provides the foundation for every text-to-speech system. NLP caters all the issues related to the language. NLP for Urdu Text-to-Speech System is a part of research project being done at CRULP, NUCES, Lahore. Its objective is to generate phonetic stream from plain Urdu text for the production of natural and intelligible speech. To carry out this transformation from text to speech, the NLP performs a series of steps. These steps are tokenization of input text, generation of phonemic stream, syllabification of words present in text, primary stress assignment on words, application of phonological rules and resyllabification of words. This paper focuses on all these processes from the perspective of Urdu.*

Keywords: *Natural Language Processing, NLP, Stress, Syllabification, TTS, Urdu Phonology.*

1 INTRODUCTION

A Text-to-Speech (TTS) synthesis system transforms plain text into intelligible speech. This transformation process can be divided into two major sub-processes namely High-level synthesis and Low-level synthesis. High-Level synthesis deals with the text analysis that changes the text into an intermediary narrow phonetic representation, whereas the Low-Level synthesis converts it into the acoustic signal for the purpose of speech production, as shown in Figure 1.

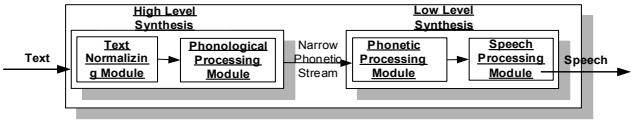


Figure 1: Text-to-Speech System

Low-level speech synthesis process can be carried out through three different techniques namely Articulatory, Concatenative and Formant based synthesis. Articulatory speech synthesis attempts to model the human speech production system directly [4]. Concatenative synthesis smoothly joins the small units of speech together to generate intelligible and natural sound [2]. Formant synthesis is based on the source-filter theory of speech production. Formant based synthesis technique undertakes the assumption that the formants transfer function can simulate the formant frequencies and formant amplitude. As mentioned above that the High-Level synthesis process transforms natural language text into an intermediary phonetic representation. This transformation is carried out by NLP component. The NLP component is capable of producing a normalized orthographic transcription of the natural language text, together with the syntactic and morphological information [7]. No matter what Low-Level synthesis processes are followed, the NLP component remains common. Therefore it is necessary for all kind of synthesis techniques. This paper focuses on NLP (i.e. High level synthesis) and discusses all of its modules in detail.

2 NLP ARCHITECTURE

The NLP component is divided into two major modules shown in Figure 2; Pre-processor and Phonological Processor.

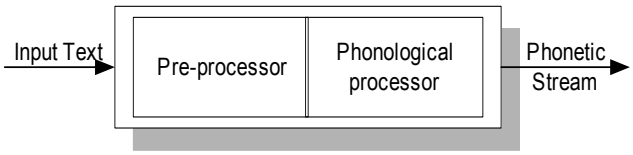


Figure 2: High Level NLP Architecture

Pre-processor organizes the input sentences into manageable lists of words. It identifies numerals, dates, times, abbreviations, acronyms and special symbols and converts them into phonemic stream. Pre-processor constitutes the following modules.

1. Number-to-Text Converter
2. Date-to-Text Converter
3. Time-to-Text Converter
4. Special Symbol to Text Converter
5. Grapheme to Phoneme Converter

Phonological Processor takes phonemic stream as input and marks syllable boundaries, stress, and rhythm and ultimately generates phonetic stream after applying phonological rules. Phonological Processor constitutes the following modules.

1. Syllable Marker
2. Stress Marker
3. Intonation Marker
4. Phonological Rule Processor

The example provided in the Table 1 depicts the functionality of the NLP. Input column shows the input stream and the output column is the output of the corresponding component. Urdu text is provided as input which contains the date which gets transformed into plain Urdu text by Date-to-Text converter. It is then passed to Grapheme to Phoneme converter which generates the stream of phonemes. Syllable Marker then marks syllable boundaries and passes the resultant stream to the Stress Marker which assigns primary stress. Finally, phonological rules convert this phonemic stream into phonetic stream. This process is schematized in Figure 3.

Table 1: Example describing the NLP Functionality

Input	Output	NLP Functionality
عثمان کی تاریخ پیدائیش ۱۰-۱۱-۱۹۸۰ -□□	عثمان کی تاریخ پیدائیش دس نومبر انیس سو اسی □□-	Date to Text Converter
عثمان کی تاریخ پیدائیش دس نومبر انیس سو اسی □□-	ʔʊsmɑn ki ʈarix pæɖaʔɪʃ ɖəs nəvəmbər ʔʊnnis sɔ ʔəssi hæ	Grapheme to Phoneme Converter
ʔʊsmɑn ki ʈarix pæɖaʔɪʃ ɖəs nəvəmbər ʔʊnnis sɔ ʔəssi hæ	ʔʊs.mɑn ki ʈa.rix pæ.ɖa.ʔɪʃ ɖəs nə.vəm.bər ʔʊn.nis sɔ ʔəs.si hæ	Syllable Marker
ʔʊs.mɑn ki ʈa.rix pæ.ɖa.ʔɪʃ ɖəs nə.vəm.bər ʔʊn.nis sɔ ʔəs.si hæ	ʔʊs.’mɑn ‘ki ʈa.’rix pæ.’ɖa.ʔɪʃ ‘ɖəs nə.’vəm.bər ʔʊn.’nis ‘sɔ ‘ʔəs.si ‘hæ	Stress Marker
ʔʊs.’mɑn ‘ki ʈa.’rix pæ.’ɖa.ʔɪʃ ‘ɖəs nə.’vəm.bər ʔʊn.’nis ‘sɔ ‘ʔəs.si ‘hæ	ʊs.’mɑn ‘ki ʈa.’rix pæ.’ɖa.ɪʃ ‘ɖəs nə.’vəm.bər ʊn.’nis ‘sɔ ‘əs.si ‘hæ	Phonological Rule Processor

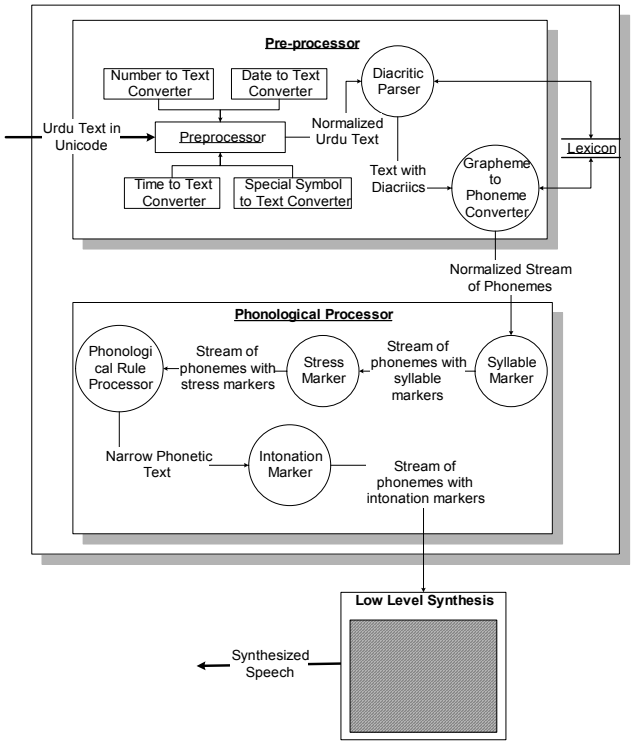


Figure 3: NLP Architecture

3 PRE-PROCESSING MODULE

Preprocessing Module contains several modules which are language dependent. Text Normalization is one parts of the Pre-processing module, which comprises of various phases that include Text Segmentation and abbreviations, date/time, symbols and numerals handling.

Digits and numerals must be expanded into full words. For example in Urdu, numeral 1990 would be expanded as *aik hazar no saw naway* and year 1990 would be expanded as *unis saw naway*. Fractions and dates are also problematic. 16/5 can be expanded as *solah-paanch* or *solah may* (if date); [4].

Special characters and symbols (i.e. ‘%’, ‘&’, ‘/’, ‘-’, ‘+’, etc) are also converted to text. For example, *5.50 rupay* must be expanded as *paanch rupay pachaas paisay*. The expression *1-2* may be expanded as *aik nafi do* or *aik say do*.

To cater all these problems, we have defined different algorithms for the numerals, date, time, currency and are using direct mapping for the special symbols stated above. The details of these algorithms are given in the following sections.

3.1 Number to Text Converter

The Number to Text Converter is devised to convert the real numbers into their equivalent literal text string in Urdu. This component uses the rules and conventions present in Urdu that govern the conversion process of a number into its equivalent string literal form, e.g. if the input number in the number form is ١٢٣٤٥٦٧٫٤٥٦ then the output number in the string literal form will be □□ لاکھ□ بار□ چونتیس□ ہزار پانچ سو سرس□□ اعشاری□ چار پانچ چ□. This component is further divided into two sub modules that are actually responsible to carry out this conversion process.

3.1.1 Number-to-Lexeme Converter

This sub-module takes a number as an input and converts it into stream of lexemes. This stream of lexemes is generated according to the rules and conventions used in Urdu to determine what number coefficient should be attached with each unit (i.e. سو، ہزار، لاکھ□ etc), e.g. if the input number is ١٢٣٤٥٦٧٨٩٫٥٦٧٨٤ then this will be converted into a stream like [١٢, ٣٤, ٥٦, ٧, ٨٩, ٫, ٥, ٦, ٧, ٨, ٤].

3.1.2 Lexemes-to-Literal String Converter

This sub-module takes a stream of lexemes, generated in the previous step, where each lexeme corresponds to the number coefficient, and converts that stream into its equivalent standard literal form of number in Urdu, e.g. if the input stream of lexeme is [١٢, ٣٤, ٥٦, ٧, ٨٩, ٫, ٥, ٦, ٧] then the literal string output will be بار□ کرو□ چونتیس لاکھ□ چ□پن□ ہزار سات سو نواسی اعشاری□ پانچ چ□ سات. This conversion is based on the Push-Down Automaton given in Figure 4. Before starting automaton a variable isFraction is set to true if the input number has a fractional part and set it to false otherwise. Secondly, the automaton starts by taking input from right side of the number e.g. if the number is

١٢٣٤٥٦ then input of automaton starts from ٦ and move towards left.

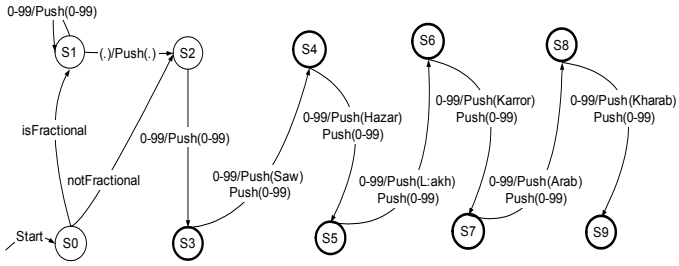


Figure 4: Automaton for Number to Literal String Converter

An example of this conversion process is given in Table 2.

Table 2: Examples of Number to Text Conversion

Input Number →	Lexemes →	Output String
١٢٣٤٥٦ء١٢٣	١, ٢٣, ٤, ٥٦, ء, ١, ٢, ٣	ایک لاکھ تئیس ہزار چار سو چھپن اعشاریہ ایک دو تین

3.2 Date to Text Converter

This component is used by the Pre-processor to convert the date in the number format into its equivalent literal string in Urdu. This component uses the rules and conventions present in Urdu that govern the conversion process, e.g. if the input date is ١٥-١١-١٩٨٠ or ١٥/١١/١٩٨٠ then this will be converted into پندرہ نومبر انیس سو اسی and if the input date is ٢٠-١٢-٢٠٠٢ or ٢٠/١٢/٢٠٠٢ then this will be converted into بیس دسمبردو ہزار دو. This component is also divided into two sub-modules that are actually responsible to carry out this conversion process.

3.2.1 Date-to-Lexeme Converter

This module takes the date, in numeral format, as an input, and converts it into stream of lexemes. This stream of lexemes contains the day, month and year in number format and also contains the extra characters used in the written numeral form of date, e.g. if the input date is ٥-١١-١٩٨٠ or ٥/١١/١٩٨٠ then the output stream of lexemes will be [٥, -, ١١, -, ١٩, ٨٠] or [٥, /, ١١, /, ١٩, ٨٠] respectively and if the input date is ١٥-١-٢٠٠١ or ١٥/١/٢٠٠١ then the output stream of lexemes will be [١٥, -, ١, -, ٢, ٠, ٠١] or [١٥, /, ١, /, ٢, ٠, ٠١] respectively.

3.2.2 Lexemes-to-Literal String Converter

This sub-module of *Date to Text Converter* takes a stream of lexemes generated in the previous step and converts that stream into a standard literal form of Date in Urdu, e.g. if the input stream of lexeme is [٥, -, ١١, -, ١٩, ٨٠] or [٥, /, ١١, /, ١٩, ٨٠] then the literal string output will be پانچ نومبر انیس سو اسی. This conversion is based on the Push-Down Automaton given in Figure 5. In the automaton 'L' represents the null transition i.e., there is no output in this transition.

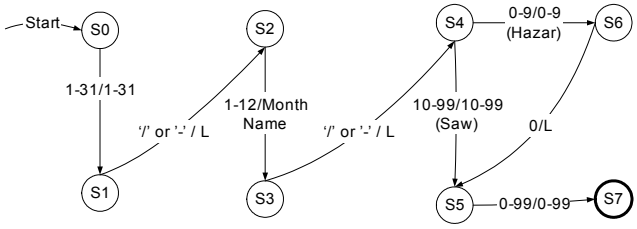


Figure 5: Automaton for Date to Literal String Converter

A complete example of this conversion process is show in Table 3.

Table 3: Examples of Date to Text Conversion

Input Date→	Lexemes →	Output String
٩-٦-١٩٨٠	٩, -,٦, -, ١٩, ٨٠	نو جون انیس سو اسی
٩-٦-٨٠	٩, -, ٦, -, ٨٠	
٩/٦/١٩٨٠	٩, /, ٦, /, ١٩, ٨٠	
٩/٦/٨٠	٩, /, ٦, /, ٨٠	

3.3 Time to Text Converter

This component is used by the Pre-processor to convert the time in the number format into its equivalent literal string in Urdu. This component uses the rules and conventions present in Urdu that govern the conversion process of the time in number format into its equivalent string literal form, e.g. if the input time is١١:٠٥ then it will be converted into گیارہ بج کر پانچ منٹ. This component is also divided into two sub-modules that are actually responsible to carry out this conversion.

3.3.1 Time-to-Lexeme Converter

This sub-module takes the time, in numeral format, as an input and converts it into a stream of lexemes. This stream of lexemes contains the hours and minutes in number format and also contains the extra character used in the written numeral form of time, e.g. if the input time is ١:٥٥ then the output stream of lexemes will be [١, :, ٥٥].

3.3.2 Lexeme-to-Literal String Converter

This sub-module of *Time to Text Converter* takes a stream of lexemes, which contains the hours, minutes and extra character, and converts that stream into a standard literal form of time in Urdu, e.g. if the input stream of lexeme is [١, :, ٥٥] then the literal string output will be ایک بج کر پچپن منٹ. This conversion is based on the Push-Down Automaton given in Figure 6.

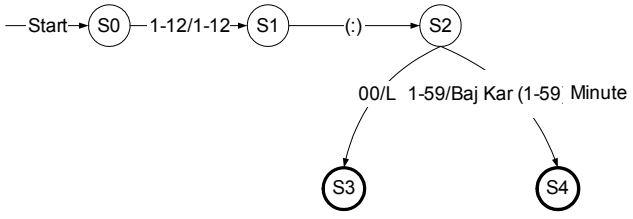


Figure 6: Automaton for Time to Literal String Conversion

3.4 Special Symbol Handler

Special symbols (-, +, ء, /, *, %, #, >, <, =, @,) are handled through direct mappings. For example (- is mapped to نفی, ء is mapped to اعشاریہ, > is mapped to اسبق, and is mapped to اللہ etc.). Other Urdu Zabta Takhti (UZT) characters are not available in Unicode 3.0. Few are present in Unicode 3.2 but Office XP does not support it at the moment.

4 GRAPHEME TO PHONEME CONVERTER

The procedure for converting Urdu text into the stream of Phonemes is presented in the form of an automaton in Figure 7. The presented automaton was tested successfully on many words of Urdu language. The tested words were supposed to have all the diacritics correctly marked on them. The Phonemic conversion process is extremely useful for building speech synthesizer for Urdu language.

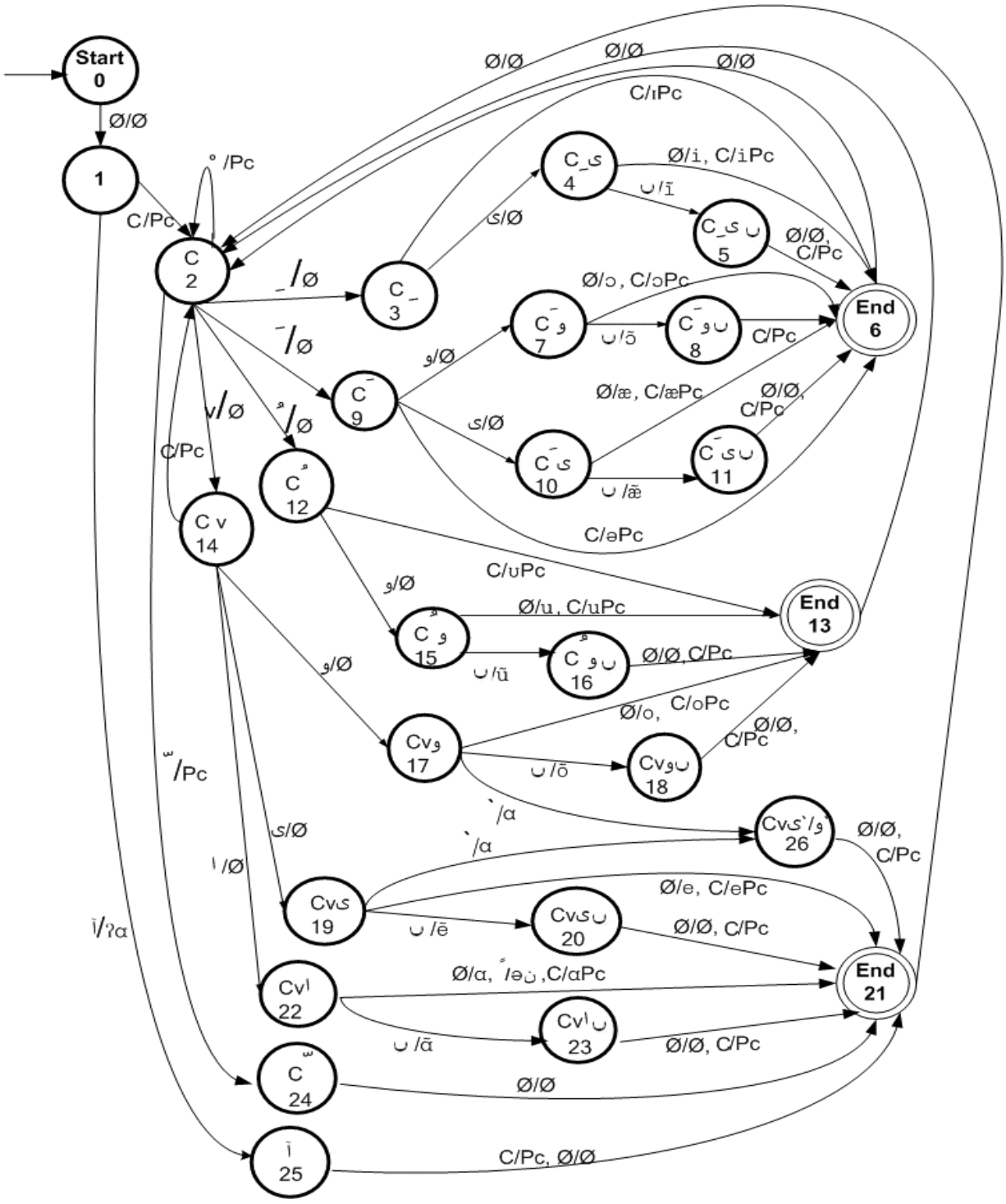


Figure 7: Grapheme to Phoneme Automaton

Since the automaton is represented in the form of a Mealey machine, so there is a corresponding output associated with each input in transition from one state to another. Circles in the Figure 7 represent states. Each intermediate state remembers all the previous inputs. Table 4 shows the list of diacritics used in the automaton. English letters are used to denote the diacritics; Urdu examples with their IPA transcription are also mentioned. The letters in the bold format under IPA column have the corresponding diacritic mentioned in Diacritics column.

Table 4: List of Urdu diacritics with example

Convention Used	Urdu Diacritics	Name	Examples	IPA
z	◌َ	Zaber	رَنگ◌َ	**rəng**
r	◌ِ	Zer	بچ◌ِنا	**bɪʧʰnɑ**
p	◌ُ	Paish	سکون	**sʊkun**
j	◌ْ	Jezem / Sukun	والدین	**vɑldæn**
m	◌ٓ	Alifmeda	آم◌ٓ	**ʔɑm**
s	◌ّ	Shed	بتّی	**beṯṯi**
d	◌ً	Do-Zaber	تقریباً◌ً	**ṯəkribən**
k	◌ٕ	Kari-Zaber	زکوٰت◌ٕ	**zəkɑṯ**
n		Null for vowels	کو	**ko**
v		Void for consonants	◌ہ◌وں	**hõ**
Pc		Phonemic Conversion	س	**s**

The automaton is built for only one syllable. It executes for each syllable from the start label and concatenates the output of each syllable thus produces the phonemic conversion of the whole word.

There are three characters in Urdu language that are responsible for representing the vowels in the word. These are 'ا', 'و', and 'ی' and can act as supporting characters for vowel insertion process or can act as consonants. If any one of the above mentioned characters has a null 'n' label, then it acts as a character for vowel. And they came as consonants and they have sounds like /ʔ, v, j/ respectively. There exist one special character 'آ' in Urdu language, which is a consonant /ʔ/ plus a vowel /ɑ/.

The consonants in Urdu language must have a diacritic, except one case. For that particular case, label void 'v' is used, which means that no diacritic is assigned to the consonant. In Urdu script, writers usually do not write words with diacritics. For example, 'بیل' (ox) and 'بیل' (ivy) have same characters, but they can be distinguished from each other by using different diacritics. The 'بیل' (ox) has "znj" and 'بیل' (ivy) has "vnj".

Grapheme to phoneme converter also processes other diacritics of Urdu script. The diacritic do-zaber '◌ً' comes at the word final position above the character 'ا', and produces the phonemes /◌n/. The diacritic khari-zaber '◌ٖ' appears with the consonants and produces the sound /ɑ/. The diacritic shed '◌ّ' appears with the consonant, and geminates it.

The nasalized vowels exist at the word final position indicated by character '◌ں' which always appears at the word final position.

Table 5 shows the method to read the transitions from the automaton in Figure 7 by considering the word "بچ◌ِنا", which has diacritics as "rjvn".

Table 5: Executing Automaton on Urdu word

State no.	Read Urdu Character	Read Diacritic	Output
0	-	-	-
1	ب	-	*b*
2	ب	*r*	*b*
3	بچ◌ِ	*r*	*bɪʧʰ*
6	بچ◌ِ	*r*	*bɪʧʰ*
2	بچ◌ِن	*rj*	*bɪʧʰn*
2	بچ◌ِن	*rjv*	*bɪʧʰn*
14	بچ◌ِنا	*rjv*	*bɪʧʰn*
22	*Ø*	*rjvn*	*bɪʧʰnɑ*
21	*End State*	*rjvn*	*bɪʧʰnɑ*

First character 'ب' is read during the transition from state 1 to state 2 and change into the phoneme /b/. Till now the only phoneme /b/ is present in the output string. Diacritic 'r' is read during the transition from state 2 to state 3, but no change appears in the output string till next character is read. Next character 'چ◌ِ' is read during the transition from state 3 to state 6 and concatenates /ɪʧʰ/ into the output string /bɪʧʰ/. Diacritic 'j' is read during the transition from state 2 to state 2, which do not play a role to insert any vowel into the output string but reads next character 'ن', which adds 'n' into the output string /bɪʧʰn/. Diacritic 'v' is read during the transition from state 2 to state 14, but no change appears into the output string /bɪʧʰn/. Finally Next character 'ا' is read during the transition from state 14 to state 22 without changing the output string /bɪʧʰn/. Phoneme /ɑ/ is finally added into the output string /bɪʧʰnɑ/, and the execution of automaton comes to an end at state 21.

5 SYLLABLE MARKER

Urdu follows a very simple rule for syllabification. Syllable templates at phonemic level include CV, CVC, CVCC, CVV, CVVC and CVVCC[5]. Firstly, it is clear from these templates that Urdu does not allow vowels at syllable initial position. Secondly, Urdu also does not allow two consonants in onset. The process of putting syllable boundaries on a word includes first finding the vowel and then marking the syllable boundary right before the first consonant in its onset. However, for initial syllable, syllable boundary is not required.

The syllable identification algorithm is given in Exhibit 1 and its pseudo code in Exhibit 2.

Step 1: *Start from right side of the phonemic string* **Step 2:** *Find vowel in the phonemic stream* **Step 3:** *Mark syllable boundary right before the first consonant in the onset.* **Step 4:** *Absorb remaining consonants in the coda of relevant syllable.* **Step 5:** *Syllable boundary is not required for the first syllable in the word as its beginning itself indicates the syllable start.*
Exhibit 1: Syllabification Algorithm

6 STRESS MARKER

According to [5] which uses moraic theory[1], Urdu language only divides syllables into two groups for stress assignment purposes, mono-moraic and multi-moraic syllables. In addition, the final mora in the final syllable is considered extrametrical i.e. its weight is not counted in the syllable weight. Thus, bi-moraic (heavy) syllable becomes mono-moraic and tri-moraic (super-heavy) syllable becomes bi-moraic in word-final position.

Algorithm in Exhibit 2 given by [5] is used for stress assignment.

Step 1: *Start from right side of the syllabified phonemic string* **Step 2:** *Assign single mora to each consonant in Coda* **Step 3:** *Assign single mora to Short Vowels* **Step 4:** *Assign two moras to Long Vowels & Diphthongs* **Step 5:** *Truncate the extrametrical mora from the word's final syllable* **Step 6:** *Count moras in syllables* **Step 7:** *Assign primary stress to first syllable found having moras greater than equal to 2 from right side.*
Exhibit 2: Stress Assignment Algorithm [5]

7 PHONOLOGICAL RULE PROCESSOR

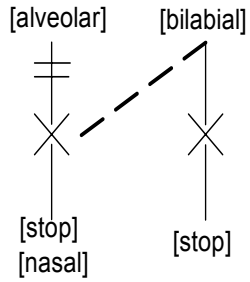
Various phonological rules have been devised by analyzing Urdu text which is listed below. These rules are applied to convert phonemes to phones. An important point to be noted here is that the following rules are devised by analyzing the data collected from Urdu dialect of people in Punjab.

1. Glottal Stop does not exist in Urdu at phonetic level. This rule states that glottal stop gets deleted whenever it is encountered. On existence of glottal stop, this rule is fired.

For example,

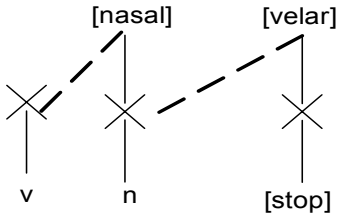
Phonemic Stream	Phonetic Stream
ʔɑʤ	ɑʤ

2. Whenever, */n/* comes before a bilabial stop, it is converted into bilabial nasal */m/*.



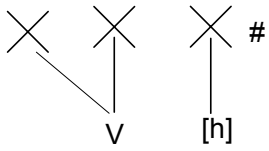
Phonemic Stream	Phonetic Stream
□nbɑr	□mbɑr

3. This rule states that velar stop gets deleted whenever it follows a vowel and the vowel becomes nasal and the nasal becomes velar.



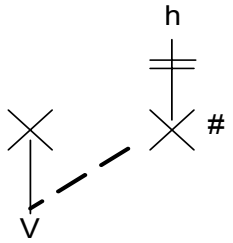
Phonemic Stream	Phonetic Stream
bɑnk	bɑ̃ ŋ

4. */h/* gets deleted whenever it comes at word final position and is preceded by a long vowel.



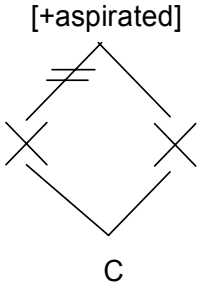
Phonemic Stream	Phonetic Stream
xɑnkɑh	xɑnkɑ

5. h gets deleted and short vowel becomes long vowel whenever h comes at word final position and is followed by a short vowel.



Phonemic Stream	Phonetic Stream
ro.z□h	ro.zɑ

6. Whenever gemination occurs among aspirated consonants, the aspiration of the first consonant is deleted.



Phonemic Stream	Phonetic Stream
□ʧʰ.ʧʰɑ	□ʧ.ʧʰɑ

7. Whenever two aspirated consonants come in start of adjacent syllables and both of them belong to the same place, then the aspiration of the second consonant disappears.

Phonemic Stream	Phonetic Stream
bʰɑ.bʰi	bʰɑ.bi

8 RESYLLABIFICATION

Resyllabification does take place in Urdu after the application of phonological rules. Since, phonological rules change the syllable structure by deleting or modifying phonemes, syllable structure sometimes become illegal. Resyllabification converts syllables into legal syllable structures. Application of phonological rules in different sequence changes a word differently. One important thing to note here is that syllable templates inventory at phonetic level is much richer than syllable templates inventory at phonemic level. Syllable templates inventory at phonetic level include CV, CVC, CVCC, CVV, CVVC, CVVCC, VV, VVC, VC and VCC [3].

Resyllabification process is under investigation these days and no concrete rules have been devised yet. However, to explain Resyllabification an example has been given below.

Phonemic Stream	Phonological Rule Fired	Resyllabification
bʰɑ.ʔi -> bʰɑ.i	ʔ -> Ø	bʰɑi

In the example above, bʰɑ.ʔi becomes bʰɑ.i when the rule ʔ -> Ø was applied. Consequently, CVV.VV transformed into CVV. This rule is currently being investigated in more detail.

9 INTONATION MARKER

Intonation defines the rhythm of a language while pronouncing the sentences of different grammatical categories. Intonation conveys meanings that apply to phrases or utterances as a whole, such as question, surprise etc.

Intonation can be handled and analyzed by controlling the pitch of the speech. The stylization technique is being used for the intonational analysis of Urdu. The analysis of Urdu revealed that a Low-High-Low (LHL) pattern/contour of pitch applies to declarative sentences of Urdu. The intonational analysis of Interrogative and other sentences is currently in progress. This component will take the intonational tones and align them with the phonetic stream. It is still under development.

10 DISCUSSION AND CONCLUSION

This paper has briefly discussed the various steps required to convert input normalized text into a phonetic stream with syllable, stress and intonation marked. This phonetic stream can be converted to speech through a low level synthesis process using articulatory, concatenative or formant synthesis techniques. As have been seen, Urdu has its own specific requirements especially for vowel specification, and all the phonological processes including syllabification, stress assignment, intonation assignment and phonological rules application. The algorithms discussed are still under investigation and still may not cater to all possibilities. For example, the syllable templates for diphthongs are still not determines. As an example it is still not clear whether /bʰɑi/ when reduced to a single syllable (see section 8) has CVVV or CVVVV template. Acoustic analysis is still underway to determine it. Similarly phonological rule inventory given is by no means complete. More rules will eventually be added as they are discovered. The NLP component does not cater to syntactic or morphological analysis. This may be required for the intonation assignment. However the word pronunciation in Urdu is very regular and vowels are predictable if diacritics are known. In addition morphological information (e.g. POS) does not change stress (or pronunciation in general). Therefore the detailed Morpho-syntactic analysis may not be required.

Still much more research needs to be done to thoroughly test and extend the algorithms proposed.

11 REFERENCES

[1] John A. Goldsmith. *Autosegmental and Metrical Phonology*; Basil Blackwell Ltd., UK, page 179, 1990.

[2] J. M. Pickett. *The Acoustics of Speech Communication*, Allyn And Bacon, London, pages 7-10, 1998.

[3] Noman Nazar. "Syllable Templates in Urdu Language". Akbhar-e-Urdu, National Language Authority, Islamabad, page 198, April-May 2002.

[4] Sami Lemmetty. "Review of Speech Synthesis Technology", Helsinki University of Technology, Master Thesis, 1999.

[5] Sarmad Hussain. "Phonetic correlates of lexical stress in Urdu"; Unpublished Ph.D. dissertation, North Western University, IL, USA, 1997.

[6] Thierry Dutoit. *Introduction to Text-To-Speech Synthesis*; Kluwer Academic Publishers, NY. 1997.

[7] Thierry Dutoit. *A Short Introduction to Text-To-Speech Synthesis*, 1996.

[8] G. Fant. *Acoustic Theory of Speech Production*. The Hague: Mounton. 1960.

[9] H. Dennis Klatt. *Software for a cascade/parallel formant synthesizer*. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.