Text Processing for Urdu TTS System

Rida Hijab Basit
Center for Language Engineering,
KICS-UET Lahore
rida.hijab@kics.edu.pk

Sarmad Hussain
Center for Language Engineering,
KICS-UET Lahore
sarmad.hussain@kics.edu.pk

Abstract

Natural Language Processing plays an important role in any Text to Speech (TTS) system. The raw text given as input to TTS may consist of numbers, dates, time, acronyms or symbols. NLP processes the raw text and converts it in the form that can be used by TTS to generate its corresponding speech. NLP consists of three parts; "Text Processing", "Text Annotation" and "Phonological Annotation". This paper enhances earlier work and details the text processing in NLP from the perspective of Urdu and also reports the results given by NLP.

1. Introduction

Text to Speech (TTS) system for any language takes a sequence of words as input and converts it into speech. The accuracy of TTS system lies in the intelligibility and naturalness of the speech it produces. Text to Speech system can be divided into three parts: Natural Language Processing, Text Parameterization and Speech Generation [1].

Natural Language Processing normalizes the raw input text and converts it to its corresponding phonetic transcription. Text Parameterization then uses this phonetic transcription to generate certain numeric parameters. Based on these parameters, Speech Generation module synthesizes corresponding speech.

The raw text given as input to TTS system can be of any form. It may consist of numbers, time, dates, symbols and any miscellaneous characters. Therefore, before converting it into speech it must be converted to some form that can be spoken by the TTS system. For this purpose, raw text first undergoes the process of Natural Language Processing.

In the past, many researchers have proposed different NLP frameworks for several languages. NLP in some Romanian TTS has been divided into three parts: text pre-processing, text normalization and phonological processing [2]. It is capable of extracting sentences, paragraphs, abbreviations, numerals, phone numbers, time and punctuations. It then syllabifies and marks stress positions on the words. NLP for many English TTS also apply Part Of Speech (POS) tags to the tokens [3, 4].

NLP for some Tamil TTS systems only handles abbreviations, acronyms and numbers [5]. Numbers include ordinary numbers, phone numbers, dates, time and currency figures. It also handles some foreign language words that the input text may encounter.

Text to Speech systems for Urdu also use NLP before generating the digital speech signal. NLP can be divided into three parts: Text Processing, Text Annotation and Phonological Annotation. This paper focuses on Text Processing and is an extension of work done in [6].

The rest of the paper is structured as follows: Section 2 describes the NLP architecture, Section 3 details the Text processing in NLP from the Urdu language perspective, Section 4 reports the results and Section 5 discusses the results whereas Section 6 concludes the paper.

2. NLP Architecture

Natural Language Processing can be divided into three categories - Text Processing, Text Annotation and Phonological Annotation, as mentioned in the previous section. Text Processing converts the raw text that may consist of numbers, dates, time or symbols into a simple text string [1]. Text annotation adds morphological, syntactic and semantic information to the input text, for example, assigns a grammatical tag to each word in a text string representing its Part of Speech (POS).

Finally, this annotated string undergoes phonological processing, which annotates further information including phonetic transcription (either through letter to sound rules [7] or looking-up pronunciation lexicon), syllable, stress and intonation. The high level architecture diagram for NLP is shown in Figure 1.

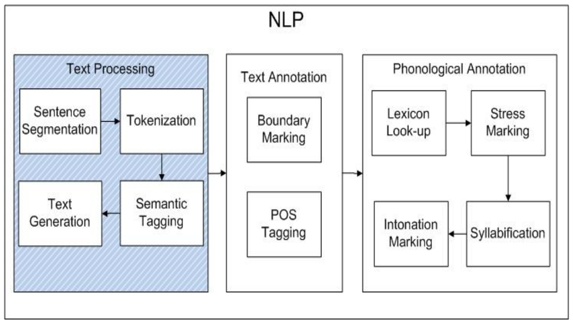

Figure 1: High level architecture diagram

The shaded portion in Figure 1 has been discussed in detail in this paper.

3. Text Processing Module

Text Processing module takes raw input from the user and converts it into normalized text. Raw input may contain any form of the text. The input text undergoes sentence segmentation as shown in Figure 1. These segmented sentences are then converted to 8-bit Urdu Zabta Takhti (UZT) [8] format for internal processing.

Text processing mainly consists of tokenization, semantic tagging and text generation. Tokenization module tokenizes the incoming sentences into words or equivalent units and sends them to the semantic tagger. Semantic tagger analyzes multiple tokens and where necessary labels them as number, time, date, text, or some other category. Each tagged token is then passed to text generation module which converts the token into its corresponding text string based on its label. The flow diagram for Text Processor has been shown in Figure 2.

Text Processor module in NLP must be able to convert numbers, symbols, time, date and miscellaneous strings to text, which can then be converted to speech by any TTS. This complexity of Text Processor module has been handled and is discussed in detail in the following sections.

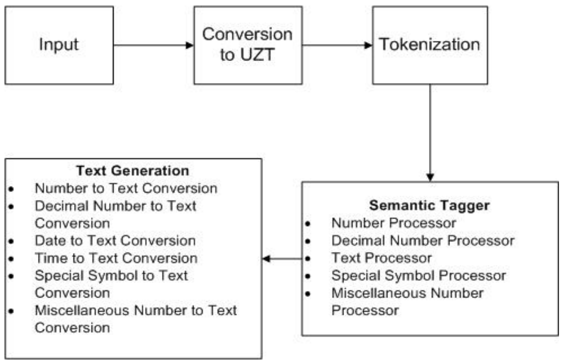

Figure 2: Text processor module

3.1. Sentence Segmentation

The input string undergoes the process of sentence segmentation. This module breaks the input string into sentences - if it encounters a full stop, question mark, line feed or carriage return character. In addition, if a sentence is very long, it is segmented at a hard limit of 400 characters (slightly shorter strings are truncated to adjust word boundaries). The whole input text is first broken into sentences and then it is sent to other modules for further processing.

3.2. Conversion to Urdu Zabta Takhti (UZT)

The segmented sentence, initially in Unicode format, is then converted to UZT format. A conversion map containing, for each Unicode symbol, a corresponding UZT symbol is then used to convert this sentence into the desired format.

This conversion is needed as UZT takes less number of bytes as compared to Unicode characters. This makes processing faster and easier. Conversion is limited to the characters which are available in UZT. Other characters are ignored at this time, except ASCII digits, which are mapped onto Urdu digits. This converted sentence is then passed on to the tokenization module.

3.3. Tokenization

The tokenization module separates words in the input string according to space and the punctuation, including ( ) ' " ! : / ، ؛ - etc. It also contains a few rules for specific cases, for example, to separate a number and text joined together like the string 12بجے into separate tokens; to identify a decimal number between digits as separately from an end of sentence marker.

However, space is not a reliable cue for tokenization. According to Naseem et al., 75% of the errors in Urdu corpus are due to spaces [9]. Therefore, Akram et al. [10] proposed a statistical technique to address this issue, based on Urdu ligature and word n-grams. The current work will be extended to include this algorithm in the future.

3.4. Semantic Tagger

It is necessary to determine how the tokens are to be converted to text and read out. For example, 22/10/2010 should be converted to " بائیس اکتوبر سن دو ہزار دس" (bɑ:i:s əkt̪u:bər sən d̪o: həzɑ:r d̪əs, "Twenty second october year two thousand ten") and not as "بائیس سلیش دس سلیش دو ہزار اور دس" (bɑ:i:s səle:ʃ d̪əs səle:ʃ d̪o: həzɑ:r ɔ:r d̪əs, "Twenty two slash ten slash two thousand and ten"). This is done by first marking "22/10/2010" strings as a single semantic unit and tagging it as a date, but "22/10" will be tagged as fractional number. Semantic tagger analyzes the tokens - tagging them as date, time, numbers (whole numbers, fractional numbers and decimal numbers), text, special symbols and miscellaneous strings [6] as shown in Figure 2. This is done by considering each token (separated in the tokenization phase) in its context to see if they can be grouped into larger semantic forms.

Semantic tagger module, initially designed in [6], has been extended to meet additional requirements for Urdu TTS. The modules of the semantic tagger are discussed in more detail below.

3.4.1. Text Processor. Text processor checks the incoming token and its next tokens (context) to determine if it's a date or text and then tags it accordingly. Some of the date formats covered by this module are جون ٢٠٠١ (June 2001) or جون ؁٢٠٠١ (June year 2001) or جون ٢٠٠١ء (June 2001 A.D.) or جون ؁٢٠٠١ء (June year 2001 A.D.) or جون ٢٣، ٢٠٠١ (June 23, 2001) or جون ٢٣، ؁٢٠٠١ (June 23, year 2001). It covers both Urdu and English digits and the months of Islamic and Gregorian calendar. It also recognizes the Arabic sign Sanah (؁) before the year and Hijri (ھ) or Eeswin (ء) symbols after the year in a date.

The text which belongs to categories other than date and time is considered miscellaneous by this text processor module.

3.4.2. Number Processor. Number processor handles both English and Urdu digits in the input string. It checks the next tokens of the numbers and based on this, tags them as whole numbers, fractional numbers, time, date or miscellaneous strings.

Whole numbers are individual digits that appear in the input string and do not have any context (relevant next tokens), for example, 12, ۵ etc،

Fractional numbers contain a '/', for example, 12/13, ١٢/١٣. Such sequences are tagged as fractional numbers. Similarly, sequences with additional constraints of having a ':' instead of a '/' and digits within hour and minute ranges, for example, 12:03, ١٢:٠٣ are tagged as time.

For dates, it checks the context and obtains a single semantic unit. Some of the examples include: ١٢-١٢-٢٠٠١ (12-12-2001) or ١٢/١٢/٢٠٠١ (12/12/2001) or ١٢ جون ٢٠٠١ (12 June 2001).

The numbers which do not lie in any of the above mentioned categories are tagged as miscellaneous strings.

3.4.3. Decimal Number Processor. As mentioned previously, tokenization module creates one token for the decimal number in the input string. Decimal number processor takes that decimal number as input and tags it. Examples include ١٢.٣ or ١٢,٣ etc،

3.4.4. Special Symbol Processor. It handles symbols and a date format that starts with a symbol. Symbols include @, #, $, %, ؓ, ؐ, ؑ, ؒ, ؁, ﷻ, ﷺ,ﷲ along-with the Urdu characters like .cte ا ب پ

Date format tagged by this processor starts with an Arabic sign Sanah (؁). Examples include: ؁٢٠٠١ or ؁2001 etc.

3.4.5. Miscellaneous String Processor. All the tokens that contain colon (:), slash (/) or dash (-), in some combination with numbers or texts, are considered miscellaneous. These strings do not lie in any of the above mentioned categories but can appear in the input text. This processor tags such strings.

3.4.6. Punctuations. Semantic tagger also tags punctuations in the input string. The punctuations that are displayed in the output string include ! ؛ ' ؟ ، -. This is because these punctuations affect the speech and must be there to ensure accurate speech synthesis.

3.5. Text Generation

Tagged tokens are then passed to text generation module which converts them into their corresponding UZT text equivalents [6]. String generation has been divided into several different sub-modules depending upon the categories as shown in Figure 2. These are

- Number to Text Converter
- Date to Text Converter
- Time to Text Converter
- Special symbol to Text Converter
- Miscellaneous to Text Converter

These modules have been discussed in detail in the following subsections.

3.5.1. Number to Text Converter. It deals with whole numbers, decimal numbers and fractional numbers. The numbers 0 to 99 have their UZT text equivalents stored, which are referred during conversion. The basic pronunciations - سو (sɔ, "hundred"), ہزار (həzɑ:r, "thousand"), لاکھ (lɑ:k^h, "Lac"), ارب (ərəb, "million"), کھرب (k^{h}ərəb, "billion"), بٹا (bət̪ɑ:, "by - used in fractional numbers"), اعشاریہ (i:ʃɑ:rɪjɑ:, "decimal point"), نصف (nɪsf, "half"), تہائی (t̪ɪhɑ:i:, "one third"), چوتھائی(ʧot̪^{h}ɑ:i:, "one fourth") etc. are also stored for future reference.

It reads whole numbers from right to left and pushes the text equivalent of each number into the stack. Counter checks the addition of سو (sɔ, "hundred"), ہزار (həzɑ:r, "thousand"), لاکھ (lɑ:k^h, "Lac"), ارب (ərəb, "million"), کھرب (k^{h}ərəb, "billion") and pushes the required pronunciation into the stack as well. At the end, the stack will have the equivalent UZT text for the whole number. The example for whole numbers is shown in Table 1.

Table 1: Whole numbers in NLP

Input	Output	IPA Transcription of Output	English Translation
100000	ایک لاکھ	e:k lɑ:k^h	One lac
١٠٠,٠٠٠	ایک لاکھ	e:k lɑ:k^h	One lac
12324	بارہ ہزار تین سو چوبیس	bɑ:rɑ: həzɑ:r t̪i:n sɔ ʧɔ:bi:s	Twelve thousand three hundred twenty four
٨٩٣٣	آٹھ ہزار نو سو تینتیس	ɑ:t^h həzɑ:r nɔ sɔ t̪e:nt̪i:s	Eight thousand nine hundred thirty three
213901	دو لاکھ تیرہ ہزار نو سو ایک	d̪o: lɑ:k^h t̪e:rɑ: həzɑ:r nɔ sɔ e:k	Two lac thirteen thousand nine hundred one
21345320	دو کروڑ تیرہ لاکھ پینتالیس ہزار تین سو بیس	d̪o: kəro:ɽ t̪e:rɑ: lɑ:k^h pæ:nt̪ɑ:li:s həzɑ:r t̪i:n sɔ bi:s	Two crore thirteen lac forty five thousand three hundred twenty

Decimal numbers are also read from right to left until a decimal point is encountered. The numbers after the decimal point are considered unique, for example, 123 in 12.123 will be converted to ایک دو تین (e:k d̪o t̪i:n, "one two three") instead of ایک سو تئیس (e:k sɔ t̪e:i:s, "one hundred and twenty three"). After their conversion, decimal point gets its equivalent pronunciation اعشاریہ (i:ʃɑ:rɪjɑ:, "decimal point") whereas rest of the numbers (before the decimal number) are dealt as whole numbers. Table 2 shows the examples for decimal number inputs.

Table 2: Decimal numbers in NLP

Input	Output	IPA Transcription of Output	English Translation
12.02	بارہ اعشاریہ دو	bɑ:rɑ: i:ʃɑ:rɪjɑ: d̪o:	Twelve point two
١٣.١١	تیرہ اعشاریہ ایک ایک	t̪e:rɑ: i:ʃɑ:rɪjɑ: e:k e:k	Thirteen point one one

Fractional numbers are converted in a same way as whole numbers except that they have slash (/) between them which is given its text equivalent بٹا (bət̪ɑ:, "by - used in fractions). This converter also recognizes special fractions with text equivalents different than the normal fractions. Special fractions include 1/2, 1/3, 1/4, 2/3, 2/4 and 3/4. 1/2 is converted to **نصف** (nɪsf, "half") instead of **ایک بٹا دو** (e:k bəʈɑ: d̪o, "one by two") in Urdu. Table 3 shows examples for fractional numbers.

Table 3:Fractions with special pronunciations

Input	Output	IPA Transcription of Output	English Translation
۶/۱۰	**چھ بٹا دس**	ʧʰ bəʈɑ: d̪əs	Six by ten
1/3	**ایک تہائی**	e:k t̪ɪha:i:	One third

The numbers can have both Urdu and English digits. Moreover, whole numbers can be represented in the form "100,000". Due to this diversity, it covers 4 formats for whole numbers, 2 for decimal numbers and 2 for fractional numbers.

3.5.2. Date to Text Converter. Date to Text Converter handles three different types of dates, which are:

- D(D)-M(M)-Y(Y) & D(D)/M(M)/Y(Y)
- D(D) Month-Text Y(Y) & Month-Text D(D), Y(Y)
- YY(YY)

Here, D(D) is the date, M(M) is the month in numbers and Y(Y), a year. Month-Text is the month already in Urdu string.

The D(D) should be in the range 1 to 31 whereas M(M) in the range of 1 to 12. Years before 2000 have different corresponding Urdu string as compared to the years like 2000 or greater than this. For example, 1992 has انیس سو بانوے (ʊni:s sɔ bɑ:nVe:, "nineteen ninety two") as its corresponding text whereas 2002 has دو ہزار دو (d̪o həzɑ:r d̪o, "two thousand and two"). Date to Text converter takes care of these two different formats of years and gives the output accordingly.

Some formats of dates are shown in Table 2.

Table 4: Date to text converter output

Input	Output	IPA Transcription of Output	English Translation
12-2-2000	باره فروری سن دو ہزار	bɑ:rɑ: fərvəri: sən d̪o: həzɑ:r	Twelve February year two thousand
۱۲/۲/۲۰۰۰	باره فروری سن دو ہزار	bɑ:rɑ: fərvəri: sən d̪o: həzɑ:r	Twelve February year two thousand
12 فروری 2000ء	باره فروری سن دو ہزار عیسوی	bɑ:rɑ: fərvəri: sən d̪o: həzɑ:r ʔi:səvi:	Twelve February year two thousand A.D.
۱۲ فروری, ۲۰۰۰	فروری باره سن دو ہزار	fərvəri: bɑ:rɑ: sən d̪o: həzɑ:r	February twelve, year two thousand
ھ1992	سن انیس سو بانوے ہجری	sən ʊni:s sɔ bɑ:nVe: hɪʤri:	Year nineteen ninety two A.H.
۲۰۰۱ء؁	سن دو ہزار ایک عیسوی	sən d̪o: həzɑ:r e:k ʔi:səvi:	Year two thousand one A.D.

During text conversion, it also keeps track of symbols like Arabic Sanah (؁), Hijri (ھ) and Eeswin (ء) which can appear with the year in a date. Type 2 covers both Islamic and Gregorian calendar months. Each type for date can be represented with both Urdu and English digits. They appear with different date symbols constituting different formats for each date type.

Type 1 has a total of 24 formats, Type 2 has 108 whereas type 3 consists of 10 different formats.

3.5.3. Time to Text Converter. It converts hours and minutes in time just as whole numbers whereas ':' in a time is replaced by بج کر (bəʤ kər, "Used to display time in Urdu"). It also checks the minutes in the time; if it shows zero minutes then it only gives the information about the hours. For example, in case of 10:30, it gives دس بج کر تیس منٹ (d̪əs bəʤ kər t̪i:s mɪnʈ, "ten thirty") whereas for 10:00 it just gives دس (d̪əs, "ten"). This module covers 6 different formats for time. Table 5 shows the examples for time.

Table 5: Time examples in NLP

Input	Output	IPA Transcription of Output	English Translation
5:00	**پانچ**	pɑ:nʧ	Five
۴:۴۰	**چار بج کر**	ʧɑ:r bəʤ kər	Four Forty

	چالیس منٹ	ʧɑ:li:s mɪnʈ	

3.5.4. Special Symbol to Text Converter. Special symbol to text converter handles 56 different symbols which are mapped onto its equivalent text by this converter. The symbols covered by this converter have already been discussed in Section 3.4.4. Some examples of special symbols and their conversion are shown in Table 6.

Table 6: Special symbols in NLP

Input	Output	IPA Transcription of Output	English Translation
رسول اکرمؐ	رسول اکرم صلی اللہ علیہ وسلم	rəsu:l əllɑ:h sələllɑ:hʊ ʔəlæ:hVəsələm	Rasool Akram salalahu alehi wasalam
@	ایٹ	æ:ʈ	At (Symbol)
ز	ز	ze:	Urdu character
ؐ	صلی اللہ علیہ وسلم	sələllɑ:hʊ ʔəlæ:hVəsələm	Arabic Symbol
ؓ	رضی اللہ عنہ	rəzi: əllɑ:hʊ ʔənhu:	Arabic Symbol
$	ڈالر	ɖɑ:lər	Dollar
%	فیصد	fi:sə̪d	Percentage
؁	سن	sən	Year
ف	ف	fe:	Urdu character
ع	ع	ʔi:n	Urdu character

3.5.5. Miscellaneous String to Text Converter. (:), (/) and (-) in miscellaneous strings are converted to their equivalent texts - **کولن** (colon), سلیش (slash) and ڈیش (dash) respectively. For any text or number that may be present in the miscellaneous strings undergo simple conversion of text or number as described in previous sections. This module covers more than 142 different formats for miscellaneous strings. Some examples are shown in Table 7.

Table 7: Miscellaneous strings in NLP

Input	Output	IPA Transcription of Output	English Translation
جون:-۲۰۰۱ ۲۰	جون دو ہزار ایک ڈیش بیس	ʤu:n d̪o: həzɑ:r e:k dæ:ʃ bi:s	June two thousand one dash twenty
12-12	بارہ ڈیش بارہ	bɑ:rɑ: dæ:ʃ bɑ:rɑ:	Twelve dash twelve
۱۲-۱۲-۱۲/	بارہ ڈیش بارہ ڈیش بارہ سلیش	bɑ:rɑ: dæ:ʃ bɑ:rɑ: dæ:ʃ bɑ:rɑ: səle:ʃ	Twelve dash twelve dash twelve slash
/جون-جون	سلیش جون ڈیش جون	səle:ʃ ʤu:n dæ:ʃ ʤu:n	Slash june dash june
جون- 2001:20	جون ڈیش دو ہزار ایک بیس	ʤu:n dæ:ʃ həzɑ:r e:k bi:s	June dash two thousand one twenty

4. Results

Text Processor module has been tested with some real data. Data used for the testing purpose is taken from Urdu newspapers, Urdu typed books, Urdu news websites and Urdu digest. Sentences are selected from each of the corpora and given to NLP. Each word is checked if it has given desired output or not, and the results are integrated.

The coverage of formats for dates, time and numbers depends on their frequency in the real data. Time and decimal numbers have fewer occurrences in the given corpora as compared to dates, whole numbers and symbols.

In the testing process, 13297 words are covered, which along-with text strings; contain numbers, symbols, and miscellaneous strings. Decimal numbers and time, found in the testing data have 100% accuracy. Accuracies for whole numbers, dates and miscellaneous strings are 99%, 91% and 93% respectively. Accuracy for symbols is reduced to 50%. This is because normalization of symbols like ؒ ؐ ؓ etc. is not correct and it does not give its full equivalent string, thus reducing the accuracy. Here, accuracy determines the correctness of conversion of input string to its Urdu format. Table 8 tabulates the results.

Table 8: Results

	Total Tokens	Correctly Identified	%age accuracy
Whole Numbers	271	267	99%
Dates	92	84	91%
Miscellaneous Numbers	40	37	93%
Symbols	62	31	50%
Time	5	5	100%
Decimal No.	16	16	100%

Majority modules show accuracy above 90%, as it can be seen in Table 8. Issues encountered during testing phase have been discussed in Section 5.

5. Discussion

The results given in Table 8 show sufficient accuracy of NLP text processing module but there are some issues that need to be resolved.

Space issues in the test data and invalid Unicode code points resulted in inappropriate conversion of whole numbers. For example, لفظ۲ shows no spaces between them, thus resulting in wrong interpretation of the whole number present in the given example. Some of the numbers in the test data contained invalid Unicode code points (not belonging to Urdu) which were not recognized by NLP thereby, giving incorrect results.
Similarly, some dates were not recognized because of space issues. In ۹ اکتوبر ۲۰۰۱ءبس, there is no space between 'ء ' and "بس " due to which it was unable to recognize the proper date format. At another point, the date of format اگست ۲۳، ۲۰۰۱ was not identified because English comma ',' was used instead of Urdu comma '،'. Some of the other dates were used with miscellaneous characters due to which they were tagged as miscellaneous strings. For example, 1999 - 2000 .ء

Symbols showing 50% accuracy are a major concern of NLP. The symbols ؐ, ◌ؑ, ◌ؓ etc. were not recognized properly during the testing phase. NLP could not generate the whole text equivalent for these symbols which reduced the overall accuracy of this module. For example, for ◌ؓ it gave رضی (rəzɪ, "Arabic symbol") only instead of رضی اللہ عنہ (rəzi: əllɑ:hʊ ʔənhu:, “Arabic symbol”).

Moreover, there were some errors related to tokenization and sentence segmentation. Tokenization errors occurred due to typing errors and space issues in the input file. Sentence segmentation errors resulted because some of the sentences consisted of more than 400 characters and as described in section 3.1, NLP segments the sentence if characters in a sentence are more than 400. This resulted in inappropriate sentence segmentation.

6. Conclusion

This paper has discussed various steps in detail, needed for converting raw text string into normalized Urdu string. Raw input may consist of numbers, date, time or symbols that must be normalized using text processing module before sending it to TTS system for speech generation. The overall accuracy for text processing module is 90.5%, which is a quite acceptable number. However, this is a work in progress and some future goals are yet to be achieved. Future goals include refining NLP output and handling more formats for each sub-module depending upon the requirements.

7. Acknowledgement

This work has been conducted through the project, Enabling Information Access for Mobile based Urdu Dialogue Systems and Screen Readers supported through a research grant from ICTRnD Fund, Pakistan.

8. References

[1] Hussain S., "Phonological Processing for Urdu Text to Speech System."*Yadava, Y, Bhattarai, G, Lohani, RR, Prasain, B and Parajuli, K (eds.) Contemporary issues in Nepalese linguistics*, 2005.

[2] Ungurean C., and Burileanu D., "An advanced NLP framework for high-quality Text-to-Speech synthesis." In *Speech Technology and Human-Computer Dialogue (SpeD)*. IEEE, 2011.

[3] Trilla A,. "Natural Language Processing techniques in Text-To-Speech synthesis and Automatic Speech Recognition.", 2009.

[4] Bhatt S., "Natural Language Processing with Text-to-Speech on Android", May 2011.

[5] Ramakrishnan A. G., Kaushil L. N., and Narayana. L., “Natural Language Processing for Tamil TTS.” in Proc. of the *3rd Language and Technology Conference*, Poznan, Poland, 2007.

[6] Hussain S., "Urdu localization project: Lexicon, MT and TTS (ULP)." in Proc. of the *Workshop on Computational Approaches to Arabic Script-based Languages*, Association for Computational Linguistics, 2004.

[7] Hussain S., "Letter-to-sound conversion for Urdu text-to-speech system." in Proc. of the *Workshop on Computational Approaches to Arabic Script-based Languages*, Association for Computational Linguistics, 2004.

[8] Hussain S., and Afzal M., "Urdu computing standards: Urdu zabta takhti (uzt) 1.01." In *Multi Topic Conference, INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, IEEE, 2001.

[9] Naseem, T., and Hussain, S., "Spelling Error Trends in Urdu." in Proc. of *Conference on Language Technology (CLT07)*, University of Peshawar, Pakistan, 2007

[10] Akram M., and Hussain S., "Word segmentation for urdu OCR system." in Proc. of the *8th Workshop on Asian Language Resources*, Beijing, China, 2010.