Transliterating Urdu for a Broad-Coverage Urdu/Hindi LFG Grammar

Muhammad Kamran Malik, Tafseer Ahmed, Sebastian Sulger, Tina Bögel, Atif Gulzar, Sarmad Hussain, Miriam Butt

LREC2010, Malta

Contents of the Talk:

Context of Work

- Computational LFG grammar in development in Konstanz

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
    - Collaborative, world-wide research project

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
  - Collaborative, world-wide research project
  - Devoted to developing *parallel* LFG grammars for a variety of languages

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
  - Collaborative, world-wide research project
  - Devoted to developing *parallel* LFG grammars for a variety of languages
  - Features and analyses are kept parallel for easy transfer between languages

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
    - Collaborative, world-wide research project
    - Devoted to developing *parallel* LFG grammars for a variety of languages
    - Features and analyses are kept parallel for easy transfer between languages
    - Languages involved:

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
    - Collaborative, world-wide research project
    - Devoted to developing *parallel* LFG grammars for a variety of languages
    - Features and analyses are kept parallel for easy transfer between languages
    - Languages involved:

  $\longrightarrow$ large-scale: English, German, French, Japanese, Norwegian

Context of Work

- Computational LFG grammar in development in Konstanz
- Aim: large-scale LFG grammar for parsing Urdu/Hindi
- Grammar is part of the ParGram project
  - Collaborative, world-wide research project
  - Devoted to developing *parallel* LFG grammars for a variety of languages
  - Features and analyses are kept parallel for easy transfer between languages
  - Languages involved:

  $\longrightarrow$ large-scale: English, German, French, Japanese, Norwegian

  $\longrightarrow$ smaller-scale (yet...): Welsh, Georgian, Hungarian, Turkish, Chinese, **Urdu** (among many others)

The ‘Parallel’ in ParGram

Analysis for transitive sentence in English ParGram grammar (F-Structure, “Functional Structure”):

The ‘Parallel’ in ParGram

Analysis for transitive sentence in English ParGram grammar (F-Structure, “Functional Structure”):

```
"Nadya saw the book."

   PRED     'see<[1:Nadya] [113:book]>'
               PRED   'Nadya'
               CHECK  [_LEX-SOURCE morphology, _PROPER known-name]
   SUBJ        NTYPE  [NSEM [PROPER [NAME-TYPE first_name, PROPER-TYPE name]]]
                      [NSYN proper                                          ]
             1 CASE nom, GEND-SEM female, HUMAN +, NUM sg, PERS 3
               PRED   'book'
               CHECK  [_LEX-SOURCE countnoun-lex]
               NTYPE  [NSEM [COMMON count]]
   OBJ                [NSYN common        ]
               SPEC   [DET [PRED      'the']]
                      [    [DET-TYPE def  ]]
           113 CASE obl, NUM sg, PERS 3
   CHECK    [_SUBCAT-FRAME V-SUBJ-OBJ]
   TNS-ASP  [MOOD indicative, PERF -_, PROG -_, TENSE past]
57 CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

The ‘Parallel’ in ParGram (cont.)

Analysis for the same transitive sentence in Urdu ParGram grammar (F-Structure, “Functional Structure”):

The ‘Parallel’ in ParGram (cont.)

Analysis for the same transitive sentence in Urdu ParGram grammar (F-Structure, “Functional Structure”):

```
"nAdiyah nE kitAb dEkHI"

   PRED     'dEkH<[1:nAdiyah] [19:kitAb]>'
            PRED      'nAdiyah'
            CHECK     [_NMORPH obl]
   SUBJ     NTYPE     NSEM [PROPER [PROPER-TYPE name]]
                      NSYN proper
            SEM-PROP [SPECIFIC +]
           1CASE erg, GEND fem, NUM sg, PERS 3
            PRED  'kitAb'
   OBJ      NTYPE NSEM [COMMON count]
                  NSYN common
          19CASE nom, GEND fem, NUM sg, PERS 3
   CHECK    _VMORPH [_MTYPE infl]
            _RESTRICTED -, _VFORM perf
   LEX-SEM [AGENTIVE +]
   TNS-ASP [ASPECT perf, MOOD indicative]
 40CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

The 'Parallel' in ParGram (cont.)

Analysis for the same transitive sentence in Urdu ParGram grammar (F-Structure, "Functional Structure"):

```
"nAdiyah nE kitAb dEkHI"

  PRED      'dEkH<[1:nAdiyah] [19:kitAb]>'
             PRED      'nAdiyah'
             CHECK     [_NMORPH obl]
  SUBJ       NTYPE     [NSEM [PROPER [PROPER-TYPE name]]]
                       [NSYN proper                    ]
             SEM-PROP  [SPECIFIC +]
           1 CASE erg, GEND fem, NUM sg, PERS 3
             PRED  'kitAb'
  OBJ        NTYPE [NSEM [COMMON count]]
                   [NSYN common       ]
          19 CASE nom, GEND fem, NUM sg, PERS 3
  CHECK      [_VMORPH [_MTYPE infl]           ]
             [_RESTRICTED -, _VFORM perf      ]
  LEX-SEM    [AGENTIVE +]
  TNS-ASP    [ASPECT perf, MOOD indicative]
40CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

→ **Analyses are kept parallel where possible**

The ‘Parallel’ in ParGram (cont.)

Analysis for the same transitive sentence in Urdu ParGram grammar (F-Structure, “Functional Structure”):

```
"nAdiyah nE kitAb dEkHI"

     [PRED     'dEkH<[1:nAdiyah] [19:kitAb]>'                       ]
     [          [PRED     'nAdiyah'                                ]]
     [          [CHECK    [_NMORPH obl]                            ]]
     [SUBJ      [NTYPE    [NSEM [PROPER [PROPER-TYPE name]]]       ]]
     [          [         [NSYN proper                     ]       ]]
     [          [SEM-PROP [SPECIFIC +]                             ]]
     [        1 [CASE erg, GEND fem, NUM sg, PERS 3                ]]
     [          [PRED  'kitAb'                         ]            ]
     [OBJ       [NTYPE [NSEM [COMMON count]]           ]            ]
     [          [      [NSYN common        ]           ]            ]
     [       19 [CASE nom, GEND fem, NUM sg, PERS 3    ]            ]
     [CHECK     [_VMORPH [_MTYPE infl]            ]                 ]
     [          [_RESTRICTED -, _VFORM perf       ]                 ]
     [LEX-SEM   [AGENTIVE +]                                        ]
     [TNS-ASP   [ASPECT perf, MOOD indicative]                      ]
   40[CLAUSE-TYPE decl, PASSIVE -, VTYPE main                      ]
```

→ **Analyses are kept parallel where possible**

→ **Features are kept parallel where possible**

Urdu

Urdu is

Urdu

Urdu is

- a South Asian language spoken primarily in Pakistan and India

Urdu

Urdu is

- a South Asian language spoken primarily in Pakistan and India
- descended from (a version of) Sanskrit (sister language of Latin)

Urdu

Urdu is

- a South Asian language spoken primarily in Pakistan and India
- descended from (a version of) Sanskrit (sister language of Latin)
- structurally identical to Hindi (spoken mainly in India)

Urdu

Urdu is

- a South Asian language spoken primarily in Pakistan and India
- descended from (a version of) Sanskrit (sister language of Latin)
- structurally identical to Hindi (spoken mainly in India)
- together with Hindi the fourth most spoken language in the world ($\sim$ 250 million native speakers)

Two Scripts, One Language

- While Urdu uses an Arabic-based script, Hindi uses Devanagari

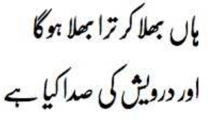
Two Scripts, One Language

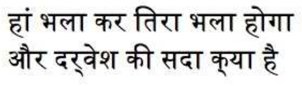
- While Urdu uses an Arabic-based script, Hindi uses Devanagari
- The same couplet by the poet Mirza Ghalib in both of the scripts:

Two Scripts, One Language

- While Urdu uses an Arabic-based script, Hindi uses Devanagari
- The same couplet by the poet Mirza Ghalib in both of the scripts:

Urdu vs. **Hindi**

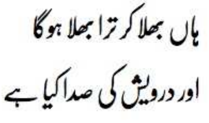

ہاں بھلا کر ترا بھلا ہوگا
اور درویش کی صدا کیا ہے

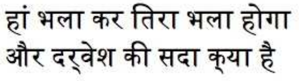

हां भला कर तिरा भला होगा
और दर्‌वेश की सदा क्या है

Two Scripts, One Language

- While Urdu uses an Arabic-based script, Hindi uses Devanagari
- The same couplet by the poet Mirza Ghalib in both of the scripts:

Urdu vs. **Hindi**


ہاں بھلا کر ترا بھلا ہوگا
اور درویش کی صدا کیا ہے


हां भला कर तिरा भला होगा
और दर्वेश की सदा क्या है

- **Common transliteration in Latin alphabet:**

hAN	bHalA	kar	tirA	bHalA	hOgA
yes	good.M.Sg	do	then	good	be.Fut.M.Sg
Or	darvES	kI	sadA	kyA	he
and	dervish	Gen.F.Sg	call.F.Sg	what	be.Pres.3.Sg

‘Yes, do good then good will happen, what else is the call of the dervish.’

Abstracting Away from the Scripts

- Faced with 2 possibilities:

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

(2) Try to abstract away from scripts to *a common transliteration*

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

(2) Try to abstract away from scripts to *a common transliteration*

- Since one grammar and lexicon can deal with both languages, efficiency and size considerations commanded us to explore the common transliteration option...

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

(2) Try to abstract away from scripts to *a common transliteration*

- Since one grammar and lexicon can deal with both languages, efficiency and size considerations commanded us to explore the common transliteration option...

Current approach:

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

(2) Try to abstract away from scripts to *a common transliteration*

- Since one grammar and lexicon can deal with both languages, efficiency and size considerations commanded us to explore the common transliteration option...

Current approach:

- Abstract away from both scripts

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

(2) Try to abstract away from scripts to *a common transliteration*

- Since one grammar and lexicon can deal with both languages, efficiency and size considerations commanded us to explore the common transliteration option...

Current approach:

- Abstract away from both scripts
- Use a common ASCII-based transliteration (A-Z, a-z, 0-9)

Abstracting Away from the Scripts

- Faced with 2 possibilities:

(1) Hard-coding the grammar and lexicon using *both scripts*

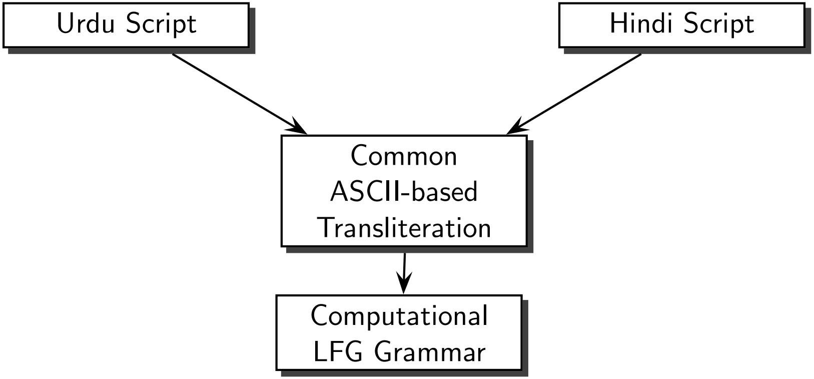
(2) Try to abstract away from scripts to *a common transliteration*

- Since one grammar and lexicon can deal with both languages, efficiency and size considerations commanded us to explore the common transliteration option...

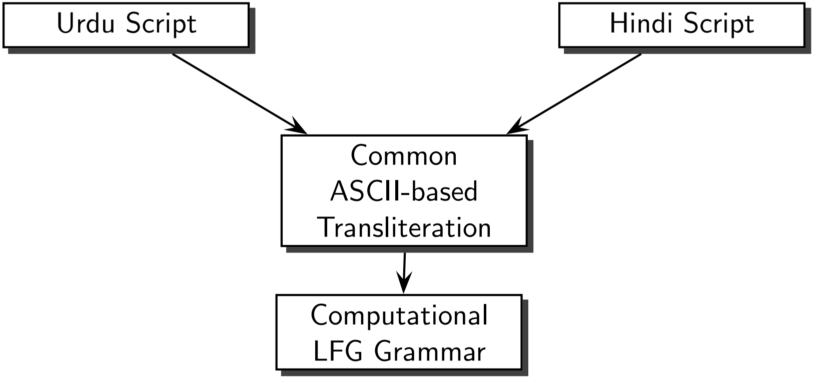
Current approach:

- Abstract away from both scripts
- Use a common ASCII-based transliteration (A-Z, a-z, 0-9)
- Encode a *single grammar and lexicon* in ASCII-based transliteration

Abstracting Away from the Scripts

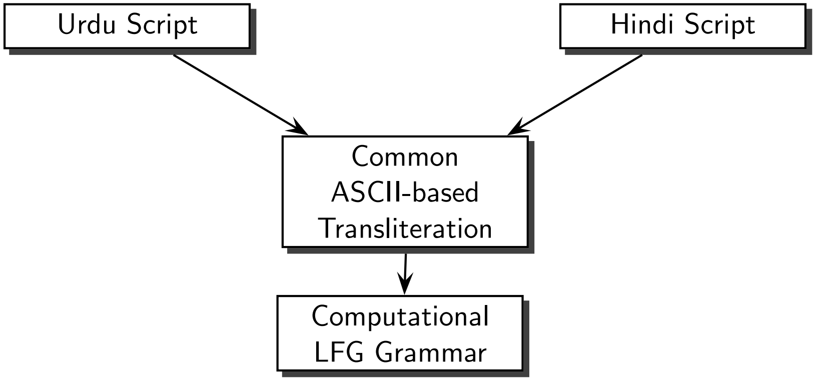


Abstracting Away from the Scripts



→ **Size of the lexicon is kept minimal**

Abstracting Away from the Scripts



→ **Size of the lexicon is kept minimal**

→ **Grammar development effort is kept minimal**

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common
  - Means that short vowels are normally not written in the script

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common
  - Means that short vowels are normally not written in the script
  - Results in some ambiguity – difficult to interpret the string

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common
  - Means that short vowels are normally not written in the script
  - Results in some ambiguity – difficult to interpret the string
- Extensive borrowing from Arabic and Persian

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common
    - Means that short vowels are normally not written in the script
    - Results in some ambiguity – difficult to interpret the string
- Extensive borrowing from Arabic and Persian
    - Foreign spelling retained in written Urdu

The Urdu Script: Some Peculiarities

- Uses extended Arabic character set
- Full letters for consonants/long vowels, *Aerabs* (diacritics) for short vowels
- Written Urdu: *Aerab* diacritics are not common
  - Means that short vowels are normally not written in the script
  - Results in some ambiguity – difficult to interpret the string
- Extensive borrowing from Arabic and Persian
  - Foreign spelling retained in written Urdu
  - Arabic and Persian graphemes map onto a single Urdu phoneme

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

(2) Dual (Consonant and Vocalic) Characters, e.g. ے $\rightarrow$ /j/ or /ae/

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

(2) Dual (Consonant and Vocalic) Characters, e.g. ے $\rightarrow$ /j/ or /ae/

(3) A Vowel Modifier Character: ں $\rightarrow$ /$\sim$/

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

(2) Dual (Consonant and Vocalic) Characters, e.g. ے $\rightarrow$ /j/ or /ae/

(3) A Vowel Modifier Character: ں $\rightarrow$ /∼/

(4) A Consonant Modifier Character: ھ $\rightarrow$ /^{h}/

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

(2) Dual (Consonant and Vocalic) Characters, e.g. ے $\rightarrow$ /j/ or /ae/

(3) A Vowel Modifier Character: ں $\rightarrow$ /$\sim$/

(4) A Consonant Modifier Character: ھ $\rightarrow$ /^{h}/

- For classes (1), (3) and (4), the mapping from graphemes to phonemes is one-to-one: a simple rule-based model can be developed

The Urdu Script: Some Peculiarities

- Urdu has 4 different character classes:

(1) Simple Consonant Characters, e.g. ف $\rightarrow$ /f/

(2) Dual (Consonant and Vocalic) Characters, e.g. ے $\rightarrow$ /j/ or /ae/

(3) A Vowel Modifier Character: ں $\rightarrow$ /∼/

(4) A Consonant Modifier Character: ھ $\rightarrow$ /^{h}/

- For classes (1), (3) and (4), the mapping from graphemes to phonemes is one-to-one: a simple rule-based model can be developed
- For class (2), context-sensitive rules were designed to account for the dual behavior

The Urdu Script: Some Peculiarities

An excerpt from our scheme table:

Unicode Urdu character	Latin letter in transliteration scheme	Phoneme
ب	b	/b/
پ	p	/p/
ت	t	/t/
ٹ	T	/ʈ/
ج	j	/ʤ/
چ	c	/t͡ʃ/

The Transliterator: A Modular Approach

- Transliterator program: component-based approach

The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules

The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules
- Components may be used as standalone applications

The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules
- Components may be used as standalone applications
- Program implemented in C++

The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules
- Components may be used as standalone applications
- Program implemented in C++
    - Program development done at CRULP, Lahore, Pakistan

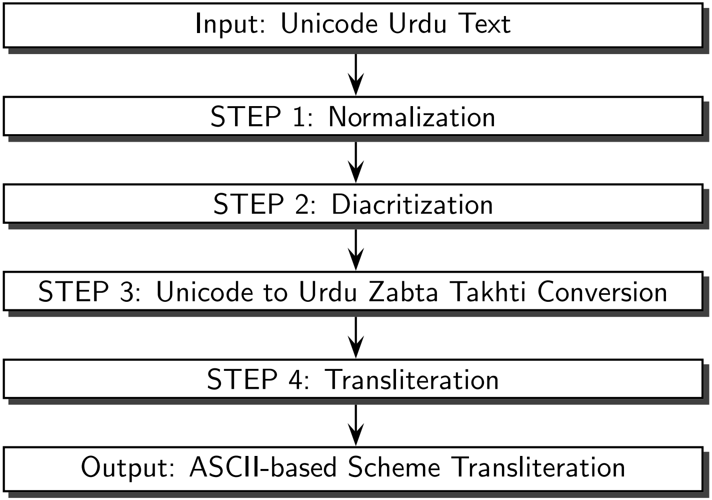
The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules
- Components may be used as standalone applications
- Program implemented in C++
  - Program development done at CRULP, Lahore, Pakistan
  - ASCII-based transliteration scheme devised in Konstanz

The Transliterator: A Modular Approach

- Transliterator program: component-based approach
- Pipeline implemented using 4 different modules
- Components may be used as standalone applications
- Program implemented in C++
  - Program development done at CRULP, Lahore, Pakistan
  - ASCII-based transliteration scheme devised in Konstanz
  - Integration in computational LFG grammar done in Konstanz

The Transliterator Pipeline



STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā
  - *decomposed form*: combined out of 2 or more characters:

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā
  - *decomposed form*: combined out of 2 or more characters:

    Alef: ا a

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā
  - *decomposed form*: combined out of 2 or more characters:

    Alef: ا a

    + lengthening diacritic *madda*: ٓ

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā
  - *decomposed form*: combined out of 2 or more characters:

    Alef: ا a

    + lengthening diacritic *madda*: ◌ٓ
- To avoid a duplication of rules, the input text is normalized to composed character form

STEP 1: Normalization

- Unicode Arabic script: characters can be written in 2 ways
  - *Composed form*: as a single entity in Unicode block:

    Alef madda: آ ā
  - *decomposed form*: combined out of 2 or more characters:

    Alef: ا a

    + lengthening diacritic *madda*: ـٓ
- To avoid a duplication of rules, the input text is normalized to composed character form

→ The system works on composed characters only!

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
  - *Aerabs* combine with simple consonants to indicate short vowels

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
    - *Aerabs* combine with simple consonants to indicate short vowels
    - *Aerabs* combine with dual behavior characters to indicate long vowels

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
  - *Aerabs* combine with simple consonants to indicate short vowels
  - *Aerabs* combine with dual behavior characters to indicate long vowels
- Our solution: lexicon lookup

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
    - *Aerabs* combine with simple consonants to indicate short vowels
    - *Aerabs* combine with dual behavior characters to indicate long vowels
- Our solution: lexicon lookup
    - Urdu lexicon data (80.000 diacritized words - gathered by CRULP in Lahore)

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
    - *Aerabs* combine with simple consonants to indicate short vowels
    - *Aerabs* combine with dual behavior characters to indicate long vowels
- Our solution: lexicon lookup
    - Urdu lexicon data (80.000 diacritized words - gathered by CRULP in Lahore)
    - Lexicon lookup: place diacritics in input words by looking up words in the lexicon

STEP 2: Diacritization

- Problem: short vowel diacritics (*Aerabs*) usually not written in Urdu
  - *Aerabs* combine with simple consonants to indicate short vowels
  - *Aerabs* combine with dual behavior characters to indicate long vowels
- Our solution: lexicon lookup
  - Urdu lexicon data (80.000 diacritized words - gathered by CRULP in Lahore)
  - Lexicon lookup: place diacritics in input words by looking up words in the lexicon

$\longrightarrow$ Ambiguity created by absence of aerab diacritics is resolved

STEP 3: Unicode to Urdu Zabta Takhti Conversion

- Urdu Zabta Takhti (UZT): national standard encoding for Urdu language processing

STEP 3: Unicode to Urdu Zabta Takhti Conversion

- Urdu Zabta Takhti (UZT): national standard encoding for Urdu language processing
  - Maps Unicode Urdu characters onto unique number sequences

STEP 3: Unicode to Urdu Zabta Takhti Conversion

- Urdu Zabta Takhti (UZT): national standard encoding for Urdu language processing
    - Maps Unicode Urdu characters onto unique number sequences
    - Developed as there was no standard industry codepage available

STEP 3: Unicode to Urdu Zabta Takhti Conversion

- Urdu Zabta Takhti (UZT): national standard encoding for Urdu language processing
  - Maps Unicode Urdu characters onto unique number sequences
  - Developed as there was no standard industry codepage available
  - Included in the pipeline for reasons of compatibility

STEP 3: Unicode to Urdu Zabta Takhti Conversion

- Urdu Zabta Takhti (UZT): national standard encoding for Urdu language processing
  - Maps Unicode Urdu characters onto unique number sequences
  - Developed as there was no standard industry codepage available
  - Included in the pipeline for reasons of compatibility

Example:

Urdu Unicode text

چَابِی *čābī* ‘key’

UZT–converted text

898083120 *čābī* ‘key’

STEP 4: Transliteration

- Convert number-based UZT format into our ASCII-based transliteration scheme

STEP 4: Transliteration

- Convert number-based UZT format into our ASCII-based transliteration scheme
- Transliteration rules are compiled into a Finite-State Machine – fast & efficient

STEP 4: Transliteration

- Convert number-based UZT format into our ASCII-based transliteration scheme
- Transliteration rules are compiled into a Finite-State Machine – fast & efficient

Example:

UZT–converted text

`898083120`	*čābī* ‘key’

transliterated Latin letter-based notation

`cAbI`	*čābī* ‘key’

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

$\longrightarrow$ Result: multiple graphemes in Urdu can map to the same phoneme

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

$\longrightarrow$ Result: multiple graphemes in Urdu can map to the same phoneme

- Solution: Map genuine Urdu letter to general letter *s*; map foreign variants to *s2, s3* etc.

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

$\longrightarrow$ Result: multiple graphemes in Urdu can map to the same phoneme

- Solution: Map genuine Urdu letter to general letter *s*; map foreign variants to *s2, s3* etc.

ص , ث , س $\rightarrow$ /s/

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

$\rightarrow$ Result: multiple graphemes in Urdu can map to the same phoneme

- Solution: Map genuine Urdu letter to general letter *s*; map foreign variants to *s2, s3* etc.

  ص , ث , س $\rightarrow$ /s/

  - Most common, genuine Urdu character: س $\rightarrow$ *s*

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

$\longrightarrow$ Result: multiple graphemes in Urdu can map to the same phoneme

- Solution: Map genuine Urdu letter to general letter *s*; map foreign variants to *s2, s3* etc.

  ص , ث , س $\rightarrow$ /s/

  - Most common, genuine Urdu character: س $\rightarrow$ *s*
  - Borrowed characters: ص , ث $\rightarrow$ *s2, s3*

STEP 4 (cont.): Transliteration of Loan Graphemes

- Loan words from Arabic and Persian include graphemes from these languages
- These graphemes occur in loan words in Urdu

→ Result: multiple graphemes in Urdu can map to the same phoneme

- Solution: Map genuine Urdu letter to general letter *s*; map foreign variants to *s2, s3* etc.

  ص , ث , س $\rightarrow$ /s/

  - Most common, genuine Urdu character: س $\rightarrow$ *s*
  - Borrowed characters: ص , ث $\rightarrow$ *s2, s3*

→ Lexicon is kept simple to read in most of the cases

Evaluation of the Transliterator

- 1000 high frequency words collected from 18 million word Urdu corpus

Evaluation of the Transliterator

- 1000 high frequency words collected from 18 million word Urdu corpus
- Accuracy is near flawless if input is diacritized

Evaluation of the Transliterator

- 1000 high frequency words collected from 18 million word Urdu corpus
- Accuracy is near flawless if input is diacritized
- Accuracy is almost as good (0.07 difference) if input contains foreign words and no diacritics

Evaluation of the Transliterator

- 1000 high frequency words collected from 18 million word Urdu corpus
- Accuracy is near flawless if input is diacritized
- Accuracy is almost as good (0.07 difference) if input contains foreign words and no diacritics
- Performance of the transliterator:

Test Corpus Size	$A = C_w / T_w$ (diacritized input)	$A = C_w / T_w$ (input without diacritics, with foreign words)
1000	0.995	0.925

Table: Accuracy Results for Transliterator

The Architecture of the Grammar

The transliterator is integrated into a parsing architecture using a Finite-State Morphological Transducer (FSMT) and the XLE Grammar Development Platform (XLE).

The Architecture of the Grammar

The transliterator is integrated into a parsing architecture using a Finite-State Morphological Transducer (FSMT) and the XLE Grammar Development Platform (XLE).

Transliterator (Urdu & Hindi Unicode to ASCII-Based Transliteration)

↓

Tokenizer

↓

Morphology (FSMT)

↓

Syntax (C- and F-Structure) (XLE)

Integrating the Transliterator

$\longrightarrow$ **Transliterator applies first**

Integrating the Transliterator

→ **Transliterator applies first**

Example (*gARI calI* ‘The car worked/started.’)

transliterator input:

گاَڑِي چَلِي

gāṛī čālī

transliterator output:

`gARI calI` *gāṛī čalī*

Integrating the Transliterator (cont.)

→ **Transliterator output feeds in XLE tokenizer**

Integrating the Transliterator (cont.)

→ **Transliterator output feeds in XLE tokenizer**

Example (*gARI calI* 'The car worked/started.')

tokenizer input:

`gARI calI` *gāṛī čalī*

tokenizer output:

`gARI TB calI TB` *gāṛī čalī*

Integrating the Transliterator (cont.)

→ **Transliterator output feeds in XLE tokenizer**

Example (*gARI calI* 'The car worked/started.')

tokenizer input:

`gARI calI` *gāṛī čalī*

tokenizer output:

`gARI TB calI TB` *gāṛī čalī*

→ **Tokenizer output feeds in FST morphological transducer**

Integrating the Transliterator (cont.)

→ **Transliterator output feeds in XLE tokenizer**

Example (*gARI calI* ‘The car worked/started.’)

tokenizer input:

`gARI calI`	*gāṛī čalī*

tokenizer output:

`gARI TB calI TB`	*gāṛī čalī*

→ **Tokenizer output feeds in FST morphological transducer**

Example (*gARI calI* ‘The car worked/started.’)

morphology output:

`gARI+Noun+Fem+Sg`	*gāṛī*
`calI+Verb+Perf+Fem+Sg`	*čalī*

Integrating the Transliterator (cont.)

$\longrightarrow$ **Morphology output feeds in XLE syntactic rules**

Integrating the Transliterator (cont.)

→ **Morphology output feeds in XLE syntactic rules**

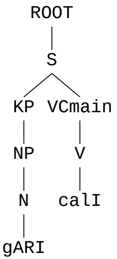
Example (*gARI calI* 'The car worked/started.')

Morphology Output/Syntax input:

gARI+Noun+Fem+Sg	*gāṛī*
calI+Verb+Perf+Fem+Sg	*čalī*

Syntax output (C-Structure and F-Structure):

CS 1:



```
"gARI calI"

   PRED     'cal<[1:gAR]>'
            PRED   'gAR'
   SUBJ     NTYPE  NSEM [COMMON count]
                   NSYN common
          1 CASE nom, GEND fem, NUM sg, PERS 3
   CHECK    _VMORPH [_MTYPE infl]
            _RESTRICTED -, _VFORM perf
   LEX-SEM  [AGENTIVE -]
   TNS-ASP  [ASPECT perf, MOOD indicative]
17 CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

References

Kenneth Beesley and Lauri Karttunen. 2003. *Finite State Morphology.* CSLI Publications, Stanford, CA.

Tina Bögel, Miriam Butt, Annette Hautli, and Sebastian Sulger. 2007. Developing a Finite-State Morphological Analyzer for Urdu and Hindi. In *Proceedings of the Sixth International Workshop on Finite-State Methods and Natural Language Processing.* Potsdam.

Miriam Butt, Tracy H. King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook.* CSLI Publications.

Miriam Butt, Helge Dyvik, Tracy H. King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar project. In *Proceedings of COLING-2002, Workshop on Grammar Engineering and Evaluation,* pages 1–7. Taipei.

Dick Crouch, Mary Dalrymple, Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell III, and Paula Newman. 2008. *XLE Documentation.* Palo Alto Research Center.

Mary Dalrymple. 2001. *Lexical Functional Grammar.* Academic Press.

Sarmad Hussain and Muhammad Afzal. 2001. Urdu Computing Standards: Urdu Zabta Takhti (UZT) 1.01. In *Proceedings of the 2001 IEEE International Multi-Topic Conference,* pages 223–228.

Sarmad Hussain. 2004. Letter-to-Sound Conversion for Urdu Text-to-Speech System. In *Proceedings of COLING-2004, Workshop on Arabic Script Based Languages.* Geneva, Switzerland.

Sarmad Hussain. 2008. Resources for Urdu Language Processing. In *Proceedings of the 6th Workshop on Asian Language Resources.* IIIT Hyderabad.

Madiha Ijaz and Sarmad Hussain. 2007. Corpus Based Urdu Lexicon Development. In *Proceedings of the Conference on Language and Technology 2007 (CLT07).* University of Peshawar, Pakistan.

Ronald M. Kaplan, John T. Maxwell III, Tracy H. King, and Richard Crouch. 2004. Integrating Finite-State Technology with Deep LFG Grammars. In *Proceedings of ESSLLI, Workshop on Combining Shallow and Deep Processing for NLP.*

Abbas Malik. 2006. *Hindi Urdu Machine Transliteration System.* MSc Thesis. University of Paris 7.

Thank you!

Are there questions?