Urdu Morphology, Orthography and Lexicon Extraction

Presented by:
Muhammad Humayoun
Department of Mathematics
University of Savoie
France
mhuma@univ-savoie.fr

Co-authors:
Harald Hammarström
Aarne Ranta
Department of Computer Science
Chalmers University of Technology
Sweden
{harald2, aarne}@cs.chalmers.se

CAASL-2, Stanford
**Introduction**

- *Indo-European → Indo-Iranian → Indo-Aryan*

- Written from right to left using Perso-Arabic Script.

- Grammar and Vocabulary influenced by Arabic, Persian and the native languages of South Asia.

- Widely Spoken in Pakistan, India and Jammu & Kashmir.

- Also spoken all over the world due to big south Asian Diaspora.

- **Urdu-Hindi**: share grammar, almost all phonology and lot of vocabulary.

- Urdu-Hindi together is the second most widely spoken language (Native + second language).
Contribution

- **Orthography component**: A Unicode Infrastructure to accommodate Perso-Arabic script of Urdu

- **Morphology component**:
  - A *type system* that covers the language abstraction completely
  - An *inflection engine* that covers word-and-paradigm morphological rules for all word classes

- **Lexicon**: Automatically extracted, 4,816 words generating 137,182 word forms.

- **Grammar component**: A small fragment of syntax
Urdu Orthography

- An alphabet of 57 letters and 15 diacritic marks
- The use of diacritic marks: optional
- Morphology and the lexicon saved in ASCII characters
  - Reusability for Hindi in future, by adding a lexicon and the transliteration scheme
  - Easy manipulation on different platforms
- Unicode support provided by a clear, strict and reversible transliteration scheme (Transliterator)
- A GUI application and useful tools
  - (Keyboard input method, Urdu Extractor)
- Implemented in Java by using ICU4J and Swing packages
Urdu Morphology

- Morphology is implemented in **Functional Morphology (FM)**
- An open source toolkit or domain embedded language for morphology development in Haskell
  - Functional Programming language, High level of abstraction, Higher-order functions, type classes, polymorphism
  - These features: good for capturing linguistic generalizations
- Idea: Dealing with grammars as reusable software libraries
- Functional Morphology treats
  - The part of speech (word classes) as **data types**
  - Their Inflection as **finite functions**
- Tools (API functions, Analyzer, Synthesizer, Exporter)
Morphology + Orthography

GUI Application

Urdu Script (Unicode enabled Urdu)

Transliteration

ASCII / Roman Urdu

Language Dependant (Urdu)

Morphology (Types, Rules, Lexicon)

FM API

Dictionary format

Language Independent Module

Analyzer

Synthesizer

Exporter

Functional Morphology Toolkit

• XFST and LexC
• GF (Grammatical Framework)
• XML
• SQL
• Full-form lexicon, tables and LATEX

Urdu Script

Transliteration

ASCII / Roman Urdu

Language Dependant (Urdu)

Morphology (Types, Rules, Lexicon)

FM API

Dictionary format

Language Independent Module

Analyzer

Synthesizer

Exporter

Functional Morphology Toolkit

• XFST and LexC
• GF (Grammatical Framework)
• XML
• SQL
• Full-form lexicon, tables and LATEX
Nouns in Urdu: type system

- Urdu Noun Inflects in **Number** (*Singular, Plural*) and **Case** (*Nominative, Oblique, Vocative*)

  ```
  data Number = Singular | Plural
  deriving (Show, Eq, Enum, Ord, Bounded)
  ```

  ```
  data Case = Nominative | Oblique | Vocative
  ```

  ```
  data NounForm = NF Number Case
  ```

  ```
  type Noun = NounForm → Str
  ```

- Inherent parameter: **Gender** (*Masculine, Feminine*)

  ```
  data Gender = Masculine | Feminine
  deriving (Show, Eq, Enum, Ord, Bounded)
  ```
Nouns in Urdu

- Nouns are divided into 15 groups based on their inflection.

- A group as running example:
  - Singular masculine nouns ending with (ا, ә), (ہ, h) and (ع, e / ʔ / ə).

- Making:
  - If a word ends with letter (ا, ә) or (ہ, h) then:
    - Plural nominative, singular oblique: last letter is replaced by (ے, e).
    - Plural oblique: the last letter is replaced by (ں, on).
    - Plural vocative: last letter is replaced by (و, o).
  - If a word ends with (ع, e / ʔ / ə): above mentioned letters will be added at the end without replacing any existing letter.
Nouns in Urdu: inflection engine

Example Noun: (Іרךa, فلک, boy)

<table>
<thead>
<tr>
<th></th>
<th>Nominative</th>
<th>Oblique</th>
<th>Vocative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Singular</strong></td>
<td>Íרךa</td>
<td>Íרךکے</td>
<td>Íרךکے</td>
</tr>
<tr>
<td><strong>Plural</strong></td>
<td>Íרךکے</td>
<td>Íطرقکے</td>
<td>Íطرقکو</td>
</tr>
</tbody>
</table>

noun_ІЍrka :: DictForm → Noun

noun_ІЍrka ІЍrka nf = **mkNoun** sg pl sg_obl pl_obl sg_voc pl_voc nf

where

sg = ІЍrka
pl = Іږک ++ "E"
sg_obl = pl
pl_obl = Іږک ++ "wN"
sg_voc = pl
pl_voc = Іږک ++ "w"
Іږک = if (end =="e") then ІЍrka else (tk 1 ІЍrka)
end = dp 1 ІЍrka
Nouns in Urdu: inflection engine

An general function for the Inflection table of nouns

\[ mk\text{Noun}::\text{String} \rightarrow \text{String} \rightarrow \text{String} \rightarrow \text{String} \rightarrow \text{String} \rightarrow \text{Number} \rightarrow \text{Case} \rightarrow \text{String} \]

\[ mk\text{Noun} \ sg \ pl \ sg\_\text{Obl} \ pl\_\text{Obl} \ sg\_\text{Voc} \ pl\_\text{Voc} \ n \ c = \]

\[
\text{case n of}
\]

Singular → case c of
- Nominative → sg
- Oblique → sg\_Obl
- Vocative → sg\_Voc

Plural → case c of
- Nominative → pl
- Oblique → pl\_Obl
- Vocative → pl\_Voc

<table>
<thead>
<tr>
<th></th>
<th>Nom</th>
<th>Obl</th>
<th>Voc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sg</td>
<td>lɑɽka</td>
<td>lɑɾke</td>
<td>lɑɾke</td>
</tr>
<tr>
<td>Pl</td>
<td>lɑɾke</td>
<td>lɑɾko</td>
<td>lɑɾko</td>
</tr>
</tbody>
</table>
An interface function for this group of nouns

\[ n1 :: DictForm \rightarrow Entry \]
\[ n1\ df = \text{masculine (noun}\_\text{IRka df}) \]

*DictForm*: a string type

*masculine*: a function for masculine words

Defined in the Lexicon:

\[ n1\ \text{I(a)R'ka} \quad (ləɽkɑ, \ ləɽkɑ) \]

- ləɽkɑ
- N
- Masc
- NF Sg Nom: ləɽkɑ
- NF Sg Obl: ləɽke
- NF Sg Voc: ləɽke
- NF Pl Nom: ləɽke
- NF Pl Obl: ləɽko
- NF Pl Voc: ləɽko
Nouns in Urdu: inflection engine

- An interface function for this group of nouns

\[
n1 :: \text{DictForm} \rightarrow \text{Entry}
\]

\[
n1 \ df = \text{masculine} \ (\text{noun}_\text{IRka} \ df)
\]

*DictForm*: a string type

*masculine*: a function for masculine words

- Defined in the Lexicon:

  \[
  \text{n1 l(a)R'ka} \quad (\text{ləɾkə, لَرْكَا})
  \]

Some other noun groups in the Lexicon:

- \text{n2 k(o)n'waN} (Well, \text{kɵnwaŋ}، گنوان)
- \text{n3 m(a)r'd} (Man, \text{məɾd}، مَرْد)
- \text{n4 k(o)r'sy} (Chair, \text{kʊrsi}, كرسي)
- \text{n5 maN} (Mother, \text{mɑɳ}, مان)
- \text{n6 g(o)R'ya} (Doll, \text{ɡʊɾiyɑ}, گریا)
- \text{n7 KwX'bw} (Fragrance, \text{xʊʃbʊ}, خوشبو)
- \text{n8 k(i)tab} (book, \text{kɪt̪ɑb}, کتاب)

..........

........
# Urdu Verbs

We divide verbs in the following categories:

- Basic stem form, direct & indirect causatives exist
- Only Basic stem form exists
- Basic stem form & direct causative form exist
- Basis stem form & indirect causative form exist
- **6 groups** have been implemented for verbs

<table>
<thead>
<tr>
<th>Category</th>
<th>Root</th>
<th>Infinitive</th>
<th>Oblique</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intransitive / Transitive / Ditransitive etc</strong></td>
<td>bən بن</td>
<td><strong>bənna بننا</strong>&lt;br&gt;<code>to build (by unknown)</code></td>
<td>bənne بننے</td>
</tr>
<tr>
<td><strong>Direct Causative</strong></td>
<td>bənɑ بنا</td>
<td><strong>bənɑna بنانا</strong>&lt;br&gt;<code>to build (by self)</code></td>
<td>bənɑne بنانے</td>
</tr>
<tr>
<td><strong>Indirect Causative</strong></td>
<td>bənvɑ بنوا</td>
<td><strong>bənvɑna بنوانا</strong>&lt;br&gt;<code>to build (by third person)</code></td>
<td>bənvɑne بنوانے</td>
</tr>
</tbody>
</table>
Urdu Verbs

- Urdu verb inflects in:
  - Gender, Number
  - Person (First, Second \{casual, familiar, respectful\}, Third \{near, distant\})
  - Tense (Subjunctive, Perfective, Imperfective)
Urdu Verbs: type system

**Category:** *Basic stem form, direct & indirect causatives exist*

type Verb = VerbForm → Str

data VerbForm =

VF Tense Person Number Gender | Caus1 Tense Person Number Gender | Caus2 Tense Person Number Gender |
Inf | Caus1_Inf | Caus2_Inf |
Inf_Fem | Caus1_Inf_Fem | Caus2_Inf_Fem |
Inf_Obl | Caus1_Inf_Obl | Caus2_Inf_Obl |
Root | Caus1_Root | Caus2_Root |

data Person = Pers1 |
Pers2_Casual | Pers2_Familiar | Pers2_Respect |
Pers3_Near | Pers3_Distant |

data Tense = Subj | Perf | Imperf
Urdu Verbs: inflection engine

mkVerbCaus12 :: String -> String -> String -> Verb
mkVerbCaus12 vInf caus1_inf caus2_inf =
    mkGenVerb root r1 r2 vInf caus1_inf caus2_inf
    where
        root = (tk 2 vInf)
        r1 = (tk 2 caus1_inf)
        r2 = (tk 2 caus2_inf)

An general function for the Inflection table

mkGenVerb :: DictForm -> DictForm -> DictForm -> DictForm -> DictForm -> DictForm -> Verb
mkGenVerb root r1 r2 vf caus1 caus2 (Root1) = root
mkGenVerb root r1 r2 vf caus1 caus2 (VF1 t p n g) = mkVAnalysis root t p n g

mkVAnalysis :: String -> Tense -> Person -> Number -> Gender -> String
mkVAnalysis root tense p n g =
    case tense of
        Subjunctive -> case p of
            Pers1 -> case n of
                Singular -> case g of
                    Masculine -> mkEnding b root "w^N" "wN"

where
t = dp 1 root
b = inStr t ["A","a","w"]
**Urdu Verbs: in lexicon**

- **v4 bnna bnana bnwana**
  
  (بنوانا بنا نا بنا
  
  Root1: ُبن
  Inf1: ُبنانا
  Inf_Obl1: ُبننے
  Inf_Fem1: ُبننی

  VF Subj Pers1 Sg Masc/Fem: ُبنون
  VF Subj Pers1 Pl Masc/Fem: ُبنن
  VF Subj Pers2_Casual Sg Masc/Fem: ُبن
  VF Subj Pers2_Casual Pl Masc/Fem: ُبنو

  ....

  Caus1 Root: ُبن
  Caus1 Inf: ُبنانا
  Caus1 Inf Obl: ُبنانے

  Caus1 Subj Pers1 Sg Masc/Fem: ُبنانون
  Caus1 Subj Pers1 Pl Masc/Fem: ُبنان
  Caus1 Subj Pers2_Casual Sg Masc/Fem: ُبن
  Caus1 Subj Pers2_Casual Pl Masc/Fem: ُبنو

  ....

  Caus2 Root: ُبنوا
  Caus2 Inf: ُبنوانا
  Caus2 Inf Obl: ُبنوانے

  Caus2 Subj Pers1 Sg Masc/Fem: ُبنوانو
  Caus2 Subj Pers1 Pl Masc/Fem: ُبنوان
  Caus2 Subj Pers2_Casual Sg Masc/Fem: ُبنوان
  Caus2 Subj Pers2_Casual Pl Masc/Fem: ُبنوانو

  ....
Other word classes

- Adjectives
- Adverbs
- The closed classes
  - Pronouns, PostPositions, Particles, Interjunctions, Conjunctions, Negations, Questions and Numerals
The Lexicon

- A wide-coverage lexicon is a key part of any morphological implementation.
- Aim: to build a lexicon automatically with minimal human efforts.
- A tool *extract* is used which is provided with the Functional Morphology.
- It requires a *paradigm file* and a *corpus*.
- To build a *corpus*:
  - A reasonable amount of Urdu Unicode text was collected from the web (news and literature domain).
  - All the html tags & other non-related information were thrown away by a tool (developed with this work) and save the file as text file.
  - Urdu Unicode text is then converted into ASCII Urdu by transliteration tool.
- The *lexicon*: extracted by applying *paradigms* on *corpus*.
The Lexicon: Problems

- Urdu is commonly written without or with a variant number of diacritic marks
  - A fundamental limitation to get a fully vocalized corpus
- Problem: having more versions per word with different diacritics
  - e.g. (کتاب, کتاب) and (كتاب, كتاب) for word (کتاب, book)
  - Point: We should save only one version per word with full diacritics
- Tokens with different diacritics are not always same words
  - e.g. (تیر, تیر, to swim) and (تیر, تیر, arrow)
  - Point: We should save all such words with full diacritics
The Lexicon Extraction - Results

To assure the correctness:

- Manually re-checking of the lexicon from word to word
- Incorrect entries thrown away

A fundamental limitation

- The missing diacritics on partly vocalized words are not applied

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (Words)</td>
<td>Extracted lexicon 9,126</td>
</tr>
<tr>
<td>1,520,000 (1.5 million)</td>
<td>Words containing Diacritics: 632</td>
</tr>
<tr>
<td>Words containing Diacritics: 23,696</td>
<td>Clean lexicon 4,816 (52.8%)</td>
</tr>
<tr>
<td>Unique tokens: 63,700 (4.1%)**</td>
<td>Words containing Diacritics: 415</td>
</tr>
</tbody>
</table>

**This conforms well to our intuition that high frequent items (postpositions, auxiliaries, particles and pronouns), account for most tokens in Urdu text.
The Lexicon Extraction - Results

- Why so many incorrect entries?
- The strictness of rules in paradigm file: normal
  - Trade-off: quality vs. coverage
- Spelling mistakes:
  - Original Typos
  - Lack of spaces between words
  - Extra spaces inside words
  - Possible Reason: The use of Urdu on web is relatively new
- Foreign words:
  - Arabic – The verses of Holy Quran in religious text
  - Persian – Poetry in slightly old literary text
  - Lot of proper nouns and English words in the news domain
The Syntax

- Urdu an **SOV** (Subject Object Verb) language
- Relatively **free word order**
- A small fragment of syntax as a separate component on top of morphology by using **Grammatical Framework**

**Grammatical Framework (GF):**
- A logical Framework
- Programming language for defining grammars (formal + natural)
- Grammar = The Abstract syntax and Concrete syntax

**In our Implementation: A sentence:**
- Combination a noun phrase (NP) and a verb phrase (VP)
- Combination of two sentences by adding a conjunction in between
The Syntax

- Abstract Syntax:
  \[
  \text{fun UsePresS: NP} \rightarrow \text{VP} \rightarrow \text{S};
  \]

- Concrete Syntax:
  \[
  \text{UsePresS np vp =}
  \{s = \text{np.s ! Nom ++ vp.s ! Present ! np.p ! np.n ! np.g}\}
  \]

  is ko k̄t̄aben̄ leni hen̄, ےہ دو کتابوں، these two books

<table>
<thead>
<tr>
<th>DemPron → Num → CN → NP</th>
<th>ye ḏo k̄t̄aben̄, یہ دو کتابوں</th>
<th>wo Ali, ہو علی</th>
<th>these two books, that Ali</th>
</tr>
</thead>
<tbody>
<tr>
<td>DemPron → PN → NP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP → PostP → CN → NP</td>
<td></td>
<td></td>
<td>is ko k̄t̄aben̄, ےہ دو یہ کتابوں</td>
</tr>
</tbody>
</table>

| Verb_Aux → VP           | hen̄, ہیں | are         |
| Verb → Verb_Aux → VP    | leni ʈhɨn̄, لینی تھین | was suppose to take |
A Complete Example

Transliteration: a(i)s kw ktabyN lyny hyN

< a(i)s >

yih_6 +DemPron - Sg Obl - Pers3_Near
mayN_8 +PersPron - Sg Pers3_Near Obl

< kw >

kw_18 +PostP

< ktabyN >

ktab_824 +N - Pl Nom - Fem

< lyna >

lyna_2 +Verb - Inf_Fem

< hyN >

hwna_0 +Verb_Aux - Present Pers1 Pl Masc

Syntactic parsing:

UsePresS
   +-------------+--------------+
   UseNP      UseVP
   +------------+----------+ +-------+------+
   UsePron kw_18
   UseN ktab_824
   (UseVP lyna_2 hwna_0)

Syntax tree
The Overall Picture

Syntax in GF

GF Morphology + UTF-8 Lexicon
(auto generated code) + Preprocessing

Orthography Component
(GUI Application, tools &Transliteration)

Morphology Component in FM

Lexicon
Conclusion

- **Merits:**
  - FM proved to be a very good choice for implementing Urdu morphology
  - A *comprehensive, reusable & elegant* implementation for Urdu that covers the linguistic abstraction (morphology) adequately

- **Limitations:**
  - A partly vocalized Lexicon
  - Run time system requires an *exact match*:
    - One cannot check if there exist orthographically different versions of a word
Future Work

- A component that matches the partly vocalized input words with the canonical words in the lexicon
- Algorithms to add missing diacritics on partly vocalized words
- A bigger lexicon
- A comprehensive implementation for syntax
- Implementation of Hindi, by adding a lexicon and a transliteration scheme
Some screen shots
Dictionary loaded: DF = 4807 and WF = 137078.

[ "كاتبة", k(i)tab >
14. {كاتبة, k(i)tab} (1074) N - NF Sg Nom - Fem
15. {كاتبة, k(i)tab} (1074) N - NF Sg Obl - Fem
16. {كاتبة, k(i)tab} (1074) N - NF Sg Voc - Fem ]
Urdu Morphology

* Functional Morphology v1.10
* (c) Markus Forsberg & Ramu Ranta 2004
* under GNU General Public License.

Implementation for Urdu
* (Muhammad Humayoun 2006)
* Chalmers University of Technology

Dictionary loaded: DF = 4907 and WF = 137078.

14. {كتب, k(i)tab} (1074) N - NF Sg Nom - Fem
15. {كتب, k(i)tab} (1074) N - NF Sg Obl - Fem
16. {كتب, k(i)tab} (1074) N - NF Sg Voc - Fem

Tagger Mode: Please be patient! It will take some time to build the data structure first time.
Urdu Script

Roman Transliteration

Aj ky bat

Unicode

\u622\u62c\u0020\u06a9\u06cc\u0020\u0628\u0627\u062a

Urdu Transliterator: Running...
Thanks for your attention

Questions / Comments

Homepage of the project

http://www.lama.univ-savoie.fr/~humayoun/UrduMorph/
Additional slides
Urdu Orthography

- Forty one non-aspirated letters:
- Fifteen aspirated letters:
- Three hamzah carrier (The glottal stop):
- The Vowels & other diacritics (Aerab / Harkat, اعراب/حركات):
Urdu Orthography - Transliteration

- Transliteration is a strict, reversible, one to one string mapping from one system of writing into another.
- Each Unicode value of Urdu alphabet is mapped with a unique Roman string
- An attempt to make transliteration as phonetic as possible
- An open source API (ICU4J) is used which is developed by IBM
- It provides a *Transliterator class* for this purpose
public class **UrduUnicode**
{
    public static final char alif='\u0627';
    public static final char bay='\u0628';
    public static final char pay='\u067e';
    ....
}

public class **UrduRoman**
{
    public static final String alif = "a";
    public static final String bay = "b";
    public static final String pay = "p";
    ..
}

private static final String **unicode_to_Roman_rules** =
    UrduUnicode.alif + "->" + UrduRoman.alif + "," +
    UrduUnicode.bay + "->" + UrduRoman.bay + "," +
    ....

public static **Transliterator roman_to_unicode** =
Transliterator.createFromRules("RomanUrdu-Unicode", roman_to_Unicode_rules, 0);

String **romanText** =
Transliterator_ur.unicode_to_roman.transliterate("Unicode Text");
### Urdu Orthography - Transliteration

#### Examples:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>Transliteration</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>كِتاب</td>
<td>k(i)tab</td>
<td>book</td>
</tr>
<tr>
<td>گوشش</td>
<td>k(a)wX(i)X</td>
<td>struggle</td>
</tr>
<tr>
<td>بُلاو</td>
<td>b(o)law^</td>
<td>to call</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sound</th>
<th>Transliteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>کے</td>
<td>k</td>
</tr>
<tr>
<td>ہ</td>
<td>(a)</td>
</tr>
<tr>
<td>ی</td>
<td>(i)</td>
</tr>
<tr>
<td>ش</td>
<td>X</td>
</tr>
<tr>
<td>ت</td>
<td>(o)</td>
</tr>
<tr>
<td>ا</td>
<td>w^</td>
</tr>
<tr>
<td>ب</td>
<td>b</td>
</tr>
</tbody>
</table>
Other word classes

- **Adjectives**
  - **Marked** (Inflects in number, case and gender)
    - Ends with (अ, a): निला, निली, निले: Blue
    - Ends with (अ, an): दायन, दायन: Right
  - **Unmarked**
    - No inflection: khush: Happy
    - Inflects in degree (Persian's style)
      - bad, worse, worst

- **Adverbs**

- **The closed classes**
  - Pronouns, PostPosition, Particle, Interjunction, Conjunction, Negations, Questions and Numerals
Lexicon Extraction – Paradigms

- Singular Feminine nouns not ending with (۰, a), (ن, N), (و, w)
  (کتاب, Kitāb, book), (گاجر, gadʒər, carrot)

regexp Not_awN = char* (char- ("a" | "N" | "w"));

paradigm n9 [x:Not_awN] =
  x { (x &
      (x"yN" | x"wN" | x"w")
   )
  };

<table>
<thead>
<tr>
<th></th>
<th>Nom</th>
<th>Oblique</th>
<th>Voc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sg</td>
<td></td>
<td>كتاب</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kitāb</td>
<td></td>
</tr>
<tr>
<td>Pl</td>
<td></td>
<td>كتابین</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kitābenә</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(yN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>كتابون</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kitābonә</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(wN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>كتابو</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kitābo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>