UrduGram: Towards a Deep, Large-Coverage Grammar for Urdu and Hindi

Tafseer Ahmed, Tina Bögel, Miriam Butt, Annette Hautli, Ghulam Raza, Sebastian Sulger and Veronika Walther

Universität Konstanz

FB Kolloquium, May 2010
1. Urdu & the UrduGram Project
2. Urdu Transliterator
3. Syntax
4. Semantics
Urdu is
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 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 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)
Urdu and Hindi

The two languages are regarded as *structurally identical*:
Urdu and Hindi

The two languages are regarded as *structurally identical*:

- syntax/morphology are practically identical
Urdu and Hindi

The two languages are regarded as *structurally identical*:

- syntax/morphology are practically identical
- vocabulary is practically identical (Urdu: borrowed from Persian/Arabic; Hindi: borrowed from Sanskrit)
The two languages are regarded as *structurally identical*:

- syntax/morphology are practically identical
- vocabulary is practically identical (Urdu: borrowed from Persian/Arabic; Hindi: borrowed from Sanskrit)
- main difference is in the script
The two languages are regarded as *structurally identical*:

- syntax/morphology are practically identical
- vocabulary is practically identical (Urdu: borrowed from Persian/Arabic; Hindi: borrowed from Sanskrit)
- main difference is in the script

→ We are developing a single grammar and lexicon for both of the languages!
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:
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:

- English, German, French, Japanese, Norwegian, Welsh, Georgian, Hungarian, Turkish, Chinese, Indonesian, Urdu (among many others)
The ParGram Grammar Architecture

Semantic Representation

C-structure Rules
- F-structure annotations
- Possibly other annotations

Morphological Analyzer
- Finite-State Morphophonology
- Stem Lexicon
- F-structure annotations

Tokenization (& Transliteration)
- Identifying Words, Punctuation

Finite-State Transducer

Generated String

Input String

C-structure Representation

(statistical preferences
OT-style constraints
Lexicon:
1) irregular or special items
2) Subcategorization frames

(Equivalent to Quasi-Logical Forms)

XFR/Transfer Rules
The ‘Parallel’ in ParGram

Analysis for transitive sentence in English ParGram grammar (F-Structure, “Functional Structure”):
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]
  NTYPE [NSEM [PROPER [NAME-TYPE first_name, PROPER-TYPE name]]
    NSYN proper
    CASE nom, GEND-SEM female, HUMAN +, NUM sg, PERS 3]

SUBJ
  PRED 'book'
  CHECK [_LEX-SOURCE countnoun-lex]
  NTYPE [NSEM [COMMON count]]
    NSYN common
  SPEC [DET PRED 'the']
    CASE obl, NUM sg, PERS 3
  CHECK [SUBCAT-FRAME V-SUBJ-OBJ]

OBJ
  TNS-ASP [MOOD indicative, PERF --, PROG --, TENSE past]
  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”):
Analysis for the same transitive sentence in Urdu ParGram grammar (F-Structure, "Functional Structure"):  

"nAdiyah nE kitAb dEkHI"

```
PRED  'dEkH<[1:nAdiyah], [20: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
  |   | 20 CASE nom, GEND fem, NUM sg, PERS 3
  |  
  |    CHECK  [VMORPH [MTYPE infl]
  |    |    _RESTRICTED -, _SUBCAT-FRAME V-SUBJ-OBJ, _VFORM perf
  |    |    LEX-SEM [AGENTIVE +]
  |    |    TNS-ASP [ASPECT perf, MOOD indicative]
  |    | 42 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”):

"nAdiyah nE kitAb dEkHI"

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"

→ Analyses are kept parallel where possible
→ Features are kept parallel where possible
The ‘Parallel’ in ParGram (cont.)

Demo: Large-Scale English ParGram Grammar
Computational Grammars - What For?

The Motivation behind ParGram
The Motivation behind ParGram

The ParGram project is working on *Deep Grammars*
Computational Grammars - What For?

The Motivation behind ParGram

The ParGram project is working on *Deep Grammars*

- Provide detailed syntactic and semantic analyses
The Motivation behind ParGram

The ParGram project is working on *Deep Grammars*

- Provide detailed syntactic and semantic analyses
- Encode grammatical functions, tense, number etc.
Computational Grammars - What For?

The Motivation behind ParGram

The ParGram project is working on *Deep Grammars*

- Provide detailed syntactic and semantic analyses
- Encode grammatical functions, tense, number etc.
- Linguistically motivated
The Motivation behind ParGram

The ParGram project is working on *Deep Grammars*

- Provide detailed syntactic and semantic analyses
- Encode grammatical functions, tense, number etc.
- Linguistically motivated
- Usually manually constructed (→ linguistic intuition)
Computational Grammars - What For?

Possible Applications
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
Computational Grammars - What For?

Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:
- Meaning-Sensitive Applications
- Web-Search
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
  - Web-Search
  - Question-Answering
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
  - Web-Search
  - Question-Answering
  - Knowledge Representation
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
  - Web-Search
  - Question-Answering
  - Knowledge Representation
- Text Summarization
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
  - Web-Search
  - Question-Answering
  - Knowledge Representation
- Text Summarization
- Machine Translation
Possible Applications

Large-Coverage, Deep Computational Grammars can be useful for:

- Meaning-Sensitive Applications
  - Web-Search
  - Question-Answering
  - Knowledge Representation
- Text Summarization
- Machine Translation
- Computer-Assisted Language Learning
Computational Grammars - What For?

powerset.com
Computational Grammars - What For?

powerset.com

- “Semantic search engine”
Computational Grammars - What For?

powerset.com

- “Semantic search engine”
- Uses large-scale English LFG
Computational Grammars - What For?

- "Semantic search engine"
- Uses large-scale English LFG
- Works on English Wikipedia
powerset.com

- “Semantic search engine”
- Uses large-scale English LFG
- Works on English Wikipedia
- Parses query and matches with parsed corpus

![Image of Powerset interface]

**Wikipedia Articles**

- [List of mergers and acquisitions by Microsoft](https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Microsoft) - Microsoft is an American multinational technology company based in Redmond, Washington. Of the 105 Microsoft has acquired, 99 were based in the United States.

- [Criticism of Microsoft](https://en.wikipedia.org/wiki/Criticism_of_Microsoft) - Criticism of Microsoft has followed various aspects of its business practices. Microsoft has acquired several companies and products, including some that competed with earlier Microsoft products.

- [History of Microsoft](https://en.wikipedia.org/wiki/History_of_Microsoft) - Activity grew quickly as developers from around the world participated, and by early 2007 commercial open source companies, such as OpenOffice, offered enterprise open source software exclusively on the Microsoft platform. Microsoft wanted to purchase Yahoo (first completely, later partially) in order to strengthen its search engine market vis-à-vis Google.

- [List of assets owned by Microsoft Corporation](https://en.wikipedia.org/wiki/List_of_assets_owned_by_Microsoft_Corporation) - Microsoft has interests in various areas: ... List of companies acquired by Microsoft Corporation
Computational Grammars - What For?

powerset.com

- “Semantic search engine”
- Uses large-scale English LFG
- Works on English Wikipedia
- Parses query and matches with parsed corpus

→ Can give better results than regular search engines
Computational Grammars - What For?

powerset.com

- “Semantic search engine”
- Uses large-scale English LFG
- Works on English Wikipedia
- Parses query and matches with parsed corpus

→ Can give better results than regular search engines

(Example: ‘X was bought by Y’ vs. ‘Y acquired X’)

Wikipedia Articles

- List of mergers and acquisitions by Microsoft
- Criticism of Microsoft
- History of Microsoft
- List of assets owned by Microsoft Corporation
Our Overall Architecture

Our parsing architecture currently looks like this:
Our Overall Architecture

Our parsing architecture currently looks like this:

tokenizer
Our Overall Architecture

Our parsing architecture currently looks like this:

tokenizer

↓

transliterator (Urdu & Hindi to Roman script)
Our Overall Architecture

Our parsing architecture currently looks like this:

```
tokenizer
↓
transliterator (Urdu & Hindi to Roman script)
↓
morphology (FST)
```
Our Overall Architecture

Our parsing architecture currently looks like this:

tokenizer
↓
transliterator (Urdu & Hindi to Roman script)
↓
morphology (FST)
↓
syntax (c- and f-structure) (XLE)
Our Overall Architecture

Our parsing architecture currently looks like this:

```
tokenizer
↓
transliterator (Urdu & Hindi to Roman script)
↓
morphology (FST)
↓
syntax (c- and f-structure) (XLE)
↓
semantics (XFR ordered rewriting)
```
Our Overall Architecture

Our parsing architecture currently looks like this:

```
tokenizer
  ↓
transliterator (Urdu & Hindi to Roman script)
  ↓
morphology (FST)
  ↓
syntax (c- and f-structure) (XLE)
  ↓
semantics (XFR ordered rewriting)
```

**XLE** is the overall development platform, with the other modules (FST and XFR) being plugged into it.
Overview

Overall Architecture

tokenizer
↓
transliterator (Urdu & Hindi to Roman script)
↓
morphology (FST)
↓
syntax (c- and f-structure) (XLE)
↓
semantics (XFR ordered rewriting)
Aim of the transliterator

Our aim is to build and integrate a transliterator that allows for both, Urdu and Hindi, to be parsed AND generated with the same grammar.

→ Right now we are working on the Urdu-Roman transliterator.
An excerpt from our scheme table:

<table>
<thead>
<tr>
<th>Unicode Urdu character</th>
<th>Latin letter in transliteration scheme</th>
<th>Phoneme</th>
</tr>
</thead>
<tbody>
<tr>
<td>ب</td>
<td>b</td>
<td>/b/</td>
</tr>
<tr>
<td>پ</td>
<td>p</td>
<td>/p/</td>
</tr>
<tr>
<td>ت</td>
<td>t</td>
<td>/t/</td>
</tr>
<tr>
<td>ت</td>
<td>T</td>
<td>/t/</td>
</tr>
<tr>
<td>ج</td>
<td>j</td>
<td>/j/</td>
</tr>
<tr>
<td>چ</td>
<td>c</td>
<td>/ṭ/w/</td>
</tr>
</tbody>
</table>
Basic idea of the transliterator

- use finite state transducer to allow for generation and parsing.
use finite state transducer to allow for generation and parsing.

Urdu script: ﮣ

ASCII: bA

↑ generating

parsing ↓
Basic idea of the transliterator

- use finite state transducer to allow for generation and parsing.

Urdu script: 

<table>
<thead>
<tr>
<th>Parsing</th>
<th>Generating</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII:</td>
<td>bA</td>
</tr>
</tbody>
</table>

- The same concept will be used to create a transliterator for Hindi/Devanagari
use finite state transducer to allow for generation and parsing.

The same concept will be used to create a transliterator for Hindi/Devanagari

This way we can parse Urdu script and generate Hindi script (and vice versa)
Position of the transliterator

- the transliterator is composed with the tokenizer
  (separates the words within a sentence)
Position of the transliterator

- the transliterator is composed with the tokenizer (separates the words within a sentence)
- tokenizer and transliterator are placed in front of the morphology
Position of the transliterator

- The transliterator is composed with the tokenizer (separates the words within a sentence).
- Tokenizer and transliterator are placed in front of the morphology.

<table>
<thead>
<tr>
<th>Transliterator</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td></td>
<td>Input</td>
<td>kitAb</td>
</tr>
<tr>
<td></td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>kitAb</td>
</tr>
<tr>
<td></td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>kitAb + Noun + Fem + Sg + Count</td>
</tr>
<tr>
<td>Morphology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLE</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
→ The transliterator at this position works quite well:

(1) laRkE kI kitAb
boy GEN book
‘The boy’s book’

→ Problem: long sentences or highly ambiguous words (when looking at script) need some time to parse.
Problems of the script - an example

The problem of the vowels ...

- Diacritics represent short vowels

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>Roman script</th>
</tr>
</thead>
<tbody>
<tr>
<td>بّ</td>
<td>ba</td>
</tr>
<tr>
<td>بٰ</td>
<td>bi</td>
</tr>
<tr>
<td>ب١</td>
<td>bu</td>
</tr>
</tbody>
</table>
Problems of the script - an example

The problem of the vowels ...

- Diacritics represent short vowels

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>Roman script</th>
</tr>
</thead>
<tbody>
<tr>
<td>بٽ</td>
<td>ba</td>
</tr>
<tr>
<td>بٽ</td>
<td>bi</td>
</tr>
<tr>
<td>بٽ</td>
<td>bu</td>
</tr>
</tbody>
</table>

(2) nAdyA nE yasIn kO kitAb dEkHnE dI
Nadya ERG Yasin DAT see let
‘Nadya let Yassin see the book’

ناديًا يٍي يٍييسٍن كٍو كٍيتٍاب دٍٍيكهٍنٍ دٍي
Problems of the script - an example

The problem of the vowels ...

- Diacritics represent short vowels

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>Roman script</th>
</tr>
</thead>
<tbody>
<tr>
<td>بّ (ba)</td>
<td></td>
</tr>
<tr>
<td>بٰ (bi)</td>
<td></td>
</tr>
<tr>
<td>بُ (bu)</td>
<td></td>
</tr>
</tbody>
</table>

Unfortunately, these diacritics tend to be left out.

نَادِیَا نِی یِسِین کُو کِتاب ڈیکھنی دی
Consequences

- If the input is without diacritics, e.g. كتاب ...

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>letter combination</th>
<th>representation</th>
<th>translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>كتاب</td>
<td>ktAb</td>
<td>kitAb</td>
<td>‘book’</td>
</tr>
</tbody>
</table>
Consequences

- If the input is without diacritics, e.g. كتاب ...

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>letter combination</th>
<th>representation</th>
<th>translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>كتاب</td>
<td>ktAb</td>
<td>kitAb</td>
<td>‘book’</td>
</tr>
</tbody>
</table>

- .. then there are all kinds of possible combinations:
  - kitAb, kutaAb, kitAbu, ikatAubi, ukitAbia, akatAbu, aukatAib ....
If the input is without diacritics, e.g. كتاب ...

<table>
<thead>
<tr>
<th>Urdu script</th>
<th>letter combination</th>
<th>representation</th>
<th>translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>كتاب</td>
<td>ktAb</td>
<td>kitAb</td>
<td>‘book’</td>
</tr>
</tbody>
</table>

.. then there are all kinds of possible combinations:

kitAb, kutaAb, kitAbu, ikatAubi, ukitAbia, akatAbu, aukatAib ....

(demo)
Solution

In order to restrict this overgeneration the possible letter combinations need to be constrained:
Solution

In order to restrict this overgeneration the possible letter combinations need to be constrained:

- which vowels are actually allowed to cooccur?
  → $ai$, but not $ia$?
In order to restrict this overgeneration the possible letter combinations need to be constrained:

- which vowels are actually allowed to cooccur?  
  \( \rightarrow \) \textit{ai}, but not \textit{ia}?

- which consonants are actually allowed to cooccur?  
  \( \rightarrow \) initial \textit{kr}, but not \textit{gr}?
In order to restrict this overgeneration the possible letter combinations need to be constrained:

- which vowels are actually allowed to cooccur?
  - $ai$, but not $ia$?

- which consonants are actually allowed to cooccur?
  - initial $kr$, but not $gr$?

- certain combinations with semi-vowels or consonants are not allowed:
  - a short vowel followed by $v$ may not be followed by $u$ or $i$
In order to restrict this overgeneration the possible letter combinations need to be constrained:

- which vowels are actually allowed to cooccur?
  \[ \rightarrow ai, \text{ but not } ia? \]

- which consonants are actually allowed to cooccur?
  \[ \rightarrow \text{ initial } kr, \text{ but not } gr? \]

- certain combinations with semi-vowels or consonants are not allowed:
  \[ \rightarrow \text{ a short vowel followed by } v \text{ may not be followed by } u \text{ or } i \]

- certain positions are prohibited:
  \[ \rightarrow \text{ a word can never end in a short vowel or begin with a short vowel that is only represented with a diacritic} \]
Solution

- write rules and filters out of these constraints and apply them to the transliterator

  (demo)
Solution

- write rules and filters out of these constraints and apply them to the transliterator

  (demo)

- Problem: these “rules” cannot be found in the literature - they are a product of extensive manual labor
Solution

- write rules and filters out of these constraints and apply them to the transliterator
  
  (demo)

- Problem: these “rules” cannot be found in the literature - they are a product of extensive manual labor

- However, the transliterator works quite well now

  → Some sentences are still a little slow (but I keep looking for possible restrictions)

  → continue with generation of Urdu and the Hindi transliterator
Overview

Overall Architecture

tokenizer
↓
transliterator (Urdu & Hindi to Roman script)
↓
morphology (FST)
↓
syntax (c- and f-structure) (XLE)
↓
semantics (XFR ordered rewriting)
syntax component is at the core of Urdu grammar
• syntax component is at the core of Urdu grammar
• theoretical background: LFG
syntax component is at the core of Urdu grammar
theoretical background: LFG
well-studied (∼ 30 years) framework with computational usability
Syntax component is at the core of Urdu grammar
theoretical background: LFG
well-studied (∼ 30 years) framework with computational usability
c- and f-structures used for syntactic representation
Syntax

- syntax component is at the core of Urdu grammar
- theoretical background: LFG
- well-studied (∼ 30 years) framework with computational usability
- c- and f-structures used for syntactic representation
  - c-structure: basic constituent structure ("tree") and linear precedence (∼ what parts belong together)
Syntax component is at the core of Urdu grammar

- theoretical background: LFG
- well-studied (~ 30 years) framework with computational usability
- c- and f-structures used for syntactic representation
  - c-structure: basic constituent structure ("tree") and linear precedence (~ what parts belong together)
  - f-structure: encodes syntactic functions and properties
Syntax

CS 1:  ROOT "nAdiyah hansI"

<table>
<thead>
<tr>
<th>PRED</th>
<th>'hans&lt;1:nAdiyah&gt;'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRED 'nAdiyah'</td>
</tr>
<tr>
<td></td>
<td>NTYPE NSEM [PROPER [PROPER-TYPE name]]</td>
</tr>
<tr>
<td></td>
<td>SUBJ NSYN proper</td>
</tr>
<tr>
<td></td>
<td>SEM-PROP [SPECIFIC 1]</td>
</tr>
<tr>
<td></td>
<td>CHECK _VMORPH [MTYPE inf]</td>
</tr>
<tr>
<td></td>
<td>_RESTRICTED - , _SUBCAT-FRAME V-SUBJ , _VFORM perf</td>
</tr>
<tr>
<td></td>
<td>LEX-SEM VERB-CLASS unerg</td>
</tr>
<tr>
<td></td>
<td>TNS-ASP ASPECT perf , MOOD indicative</td>
</tr>
<tr>
<td></td>
<td>18 CLAUSE-TYPE decl , PASSIVE - , VTYPE main</td>
</tr>
</tbody>
</table>

KP VCmain

<table>
<thead>
<tr>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

NP V

<table>
<thead>
<tr>
<th>nAdiyah</th>
</tr>
</thead>
<tbody>
<tr>
<td>hansI</td>
</tr>
<tr>
<td>nAdiyah</td>
</tr>
</tbody>
</table>

CLause-TYPE  decl, PASSIVE - , VTYPE main
current size: 53 phrase-structure rules, annotated for syntactic function (usual size of large-scale grammars: 350–400 rules)
current size: 53 phrase-structure rules, annotated for syntactic function (usual size of large-scale grammars: 350–400 rules)

coverage: basic clauses with free word order, NP syntax, tense and aspect, causative verbs, complex predicates, relative clauses, passives, semantically-based case marking
Discontinuous NPs in Urdu

1. Well known discontinuities
2. NP-internal discontinuity in Urdu
3. LFG implementation
4. Conclusion
Extraction from DP

(2) a. Er hat **viele Bücher über Logik** gekauft.
   He has many books on logic bought
   ‘He has bought many books about logic.’

b. **Bücher über Logik** hat er **viele** gekauft.

c. **Über Logik** hat er **viele Bücher** gekauft. (German)

(3) mantiq=par nidA=nE Ek kitAb
    logic=Loc.on Nida=Erg one book.F.3Sg
    xarId-l he.
    buy-Perf be.Pres
    ‘Nida has purchased a book on logic.’ (Urdu)
Quantifier Float

(4) a. **They all** have bought a car.
   b. **They** have **all** bought a car.

(5)

*Am al₂nE bahut kHA-E mango.Pl Ali=Erg many eat-Perf*  
‘Ali ate many mangoes.’ (Urdu)
Constituent-level discontinuities in Urdu

NP-internal discontinuity
- Discontinuous NP
- Discontinuous AP
When NP-internal discontinuity occurs in Urdu

The NP-internal discontinuity in Urdu can occur when the argument-taking noun is modified by:

- ARGUMENT-TAKING ADJECTIVES
- ARGUMENT-TAKING SPECIFIER NOUNS
## Argument-taking adjectives in Urdu

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Type of Argument</th>
<th>Example of Adjective Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>Dative Marked</td>
<td>sadr=kO hAsil president=Dat possessed ‘possessed by the president’</td>
</tr>
<tr>
<td>(ii)</td>
<td>Ablative Marked</td>
<td>adliyah=sE xAif courts=Abl afraid ‘afraid of courts’</td>
</tr>
<tr>
<td>(iii)</td>
<td>Locative Marked</td>
<td>buxAr=mEN mubtalA fever=Loc.in suffered ‘suffered with fever’</td>
</tr>
<tr>
<td>(iv)</td>
<td>Adpositional</td>
<td>sihat=kE liyE muzir health=Gen for harmful ‘harmful for health’</td>
</tr>
</tbody>
</table>
Simple examples of argument-taking nouns

(6) a. istisnA
   ‘immunity’

b. muqaddamAt=sE istisnA
court-case.Pl=Abl immunity
   ‘immunity from court-cases’

c. muqaddamAt=sE AInI istisnA
court-case.Pl=Abl constitutional immunity
   ‘constitutional immunity from court-cases’
Simple examples of argument-taking nouns

(7) a. barIfiNg
    ‘briefing’

b. salAmtI=par barIfiNg
   security=Loc briefing
    ‘briefing on security’

c. salAmtI=par tafsI=barIfiNg
   security=Loc detailed briefing
    ‘detailed briefing on security’
Simple examples of argument-taking nouns

(8) a. mutAlbA
   ‘demand’

b. 
   Arml-clf=sE mutAlbA
   army-chief=Abl demand
   ‘demand to the army-chief’

c. 
   Arml-clf=sE qAnUnI mutAlbA
   army-chief=Abl legal demand
   ‘legal demand to the army-chief’
Examples of discontinuous NPs

(9) a1. sadr=kO₁ hAsil₁ muqaddamAt=sE₂
   president=Dat possessed court-cases=Abl
   AInl istisnA₂
   constitutional immunity
   ‘Constitutional Immunity from court-cases possessed by the president’

a2. \([\text{NP}[\text{AP}[\text{KP} \text{sadr=kO}] \text{hAsil}][\text{KP} \text{muqaddamAt=sE}] \text{AInl istisnA}]\)

b. muqaddamAt=sE₂ sadr=kO₁ hAsil₁ AInl istisnA₂

c. sadr=kO₁ muqaddamAt=sE₂ hAsil₁ AInl istisnA₂

d. \(*\text{hAsil}_1 \text{muqaddamAt}=\text{sE}_2 \text{sadr}=\text{kO}_1 \text{AInl istisnA}_2\)
Hierarchical structure of AP in NP

Figure: Hierarchical structure of AP in NP
Examples of discontinuous NPs

(10) a1.

\[\text{Arml-clf}=\text{sE}_2 \quad \text{salAmtl}=\text{par}_1 \quad \text{barIfiNg}_1=\text{kA \ mutAlbA}_2 \]

\[\text{army-chief}=\text{Abl \ security}=\text{Loc.on \ briefing}=\text{Gen \ demand} \]

‘The demand to the army chief for briefing on security’

a2. \[\text{NP}[\text{KP \ Arml-clf}=\text{sE}][\text{KP}[\text{NP}[\text{KP \ salAmtl}=\text{par} \ \text{barIfiNg}]=\text{kA}]] \]

\[\text{mutAlbA} \]

b. \[\text{salAmtl}=\text{par}_1 \ \text{Arml-clf}=\text{sE}_2 \ \text{barIfiNg}_1=\text{kA \ mutAlbA}_2 \]
Examples of discontinuous NPs

(11) \[ NP[KP \text{ ArmI-clf=sE}] \quad [KP[NP[KP \text{ mulkI salAmtI=par}]
\text{army-chief=Abl of-country security=Loc.on tafsIlI barIfiNg=kA] qAnUnI mutAlbA}]
\text{detailed briefing=Gen legal demand}

‘The legal demand to the army chief for a detailed briefing on security of the country’
LFG implementation of NP-internal discontinuity

Figure: Word Order in Noun Phrases of Urdu
Implementation Issues

- Free word order in an NP
- Relating arguments with corresponding heads
- Head last constraint
LFG instruments used

- **Shuffle operator (‘,’):**
  To accommodate free word order of different elements in the noun phrases.

- **Non-deterministic operator (‘$’):**
  Relating the corresponding arguments to the corresponding heads.

- **Head Precedence Operator (‘<h’):**
  To make it sure that the head must not precede its arguments in the noun phrases.
An excerpt from Grammar Rules

NP $\rightarrow$ KP*: { (^ ADJUNCT $\ OBL)$)= !
  | (ADJUNCT $\ OBJ-GO)$)= !
  | (^ OBL) = !
  | (^ OBJ-GO) = ! }
  , “for scrambling”
AP*: ! $ (\ ^ \ ADJUNCT )$
N : ^ = !

KP*: { (^ ADJUNCT $\ OBL)$)= !
  (^ ADJUNCT) <h (^ ADJUNCT $\ OBL)$
  | ...... }
  ...... 

**Figure:** Grammar Rules
C-structure for a discontinuous NP

**Figure:** C-structure
F-structure for a discontinuous NP

"s3adr kO muqaddamAt sE h2As3il AInI istis2nA"

```
PRED 'istas2nA<[34:muqaddamah]>'
PRED 'muqaddamah'
OBL CHECK [_NMORPH obl]
34 CASE inst, GEND masc, NUM pl
  PRED 'h2As2il<[1:s3adr]>'
    PRED 's3adr'
    CHECK [_NMORPH obl]
  OBJ-GO NTYPE NSEM [COMMON count]
    NSYN common
  1 CASE dat, GEND masc, NUM sg, PERS 3
ADJUNCT
  CHECK [_RESTRICTED -]
  LEX-SEM [GOAL +]
  39 ATYPE attributive
    PRED 'AInI'
    ATYPE attributive
    44 <s [39:h2As2il]
```

**Figure:** F-structure
Summary

Urdu is a typical language in which discontinuous NPs are found both at:
- Clause-level
- Constituent-level

Constituent-level discontinuity in Urdu can be implemented in LFG framework by making use of:
- Shuffle operator (‘,’)
- Non-deterministic operator (‘$’)
- Head-precedence operator (‘<$h$’)

Overall Architecture

tokenizer

↓

transliterator (Urdu & Hindi to Roman script)

↓

morphology (FST)

↓

syntax (c- and f-structure) (XLE)

↓

semantics (XFR ordered rewriting)
Semantics

Intro

Aim: a large-coverage computational semantic analyzer on the basis of a deep syntactic analysis

- use f-structures as starting point
- apply XFR semantic rules → from f-structure facts to a semantic representation (Crouch and King, 2006)
- judgment on the semantic well-formedness of a sentence
  - The girl laughs. → semantically well-formed
  - #The tree laughs. → semantically ill-formed
- we need lexical information about the words in a sentence

1 lexical resource for Urdu verbs
  - more information on the verb and its arguments
2 general lexical resource for Urdu nouns, adjectives etc.
F-structure for \textit{nAdiyah hansI} (Nadya laughed).

\begin{verbatim}
"nAdiyah hansI"

  [PRED 'hans<1:nAdiyah>']
  [PRED 'nAdiyah']
  [SUBJ [NTYPE [NSEM [PROPER [PROPER-TYPE name]]]]]
  [SEM-PROP [SPECIFIC +]]
  [CASE nom, GEND fem, NUM sg, PERS 3]
  [CHECK [_VMORPH [_MTYPE inf]]]
  [RESTRICTED -, _VFORM perf]
  [LEX-SEM [VERB-CLASS unerg]]
  [TNS-ASP [ASPECT perf, MOOD indicative]]
  [CLAUSE-TYPE decl, PASSIVE -, VTYPE main]

XFR semantic rule:

  PRED(%1, hans), SUBJ(%1, %subj), -OBJ(%1, %obj)
  =>
  word(%1, hans, verb), role(Agent, %1, %subj).
\end{verbatim}
Developing an Urdu VerbNet (1)

- following the methodology of the English VerbNet (Kipper-Schuler 2006)
  - categorization of English verbs in 250 classes
  - information on event structure and argument structure of verbs
  - provides the general architecture for a VerbNet in any language
  - e.g. parts of the entry for ‘laugh’ in the English VerbNet

```
MEMBER: laugh

THEMROLE: Agent [+animate]
THEMROLE: Theme [+communication]
THEMROLE: Cause
THEMROLE: Recipient [+animate]
FRAME: [NP V // Basic Intransitive]
  EXAMPLE: Paul laughed.
  SYNTAX: %Agent V
FRAME: [NP V NP // TransitiveCognate Object]
  EXAMPLE: Paul laughed a cheerful laugh.
  SYNTAX: %Agent V %Theme
FRAME: [NP V PP.recipient // PPRecipient-PP]
  EXAMPLE: Paul laughed at Mary.
  SYNTAX: %Agent V {{+dest_dir}} %Recipient
```
Developing an Urdu VerbNet (2)

**Difficulty:** resource sparseness of Urdu

**Approach 1:**
- translating the entries in the English VerbNet to Urdu
- figure out problematic cases

**Approach 2:**
- fully rely on corpus work
- extend tool for automatic subcategorization extraction (Ghulam, 2010)

Can we benefit from a Hindi lexical resource?
Hindi WordNet

**Facts:**

- inspired in methodology and architecture by the English WordNet (Fellbaum 1998)

- **S:** (n) **book** (a written work or composition that has been published (printed on pages bound together)) "I am reading"
  - **direct hyponym / full hyponym**
  - **part meronym**
  - **has instance**
    - **direct hypernym / inherited hypernym / sister term**
  - **S:** (n) **publication** (a copy of a printed work offered for distribution)
    - **S:** (n) **work, piece of work** (a product produced or accomplished through the effort or activity or as one of his more memorable works"; "the symphony was hailed as an ingenious work"; "he was in work of an active imagination"; "erosion is the work of wind or water over time"
    - **S:** (n) **product, production** (an artifact that has been created by someone or some process) "the most of their agricultural production"
    - **S:** (n) **creation** (an artifact that has been brought into existence by someone)
      - **S:** (n) **artifact, artefact** (a man-made object taken as a whole)
      - **S:** (n) **whole, unit** (an assemblage of parts that is regarded as a single entity "the team is a unit"
    - **S:** (n) **object, physical object** (a tangible and visible entity; an entity "balls and other objects"
      - **S:** (n) **physical entity** (an entity that has physical existence)
      - **S:** (n) **entity** (that which is perceived or known or inferred nonliving)
Hindi WordNet

- developed at the Indian Institute of Technology, Bombay, India
- separated into four independent “semantic nets”
  - verbs, nouns, adjectives and adverbs
- about 3.900 verbs, 57.000 nouns, 13.700 adjectives and 1.300 adverbs
- words are grouped according to their meaning similarity (“synsets”)
Hindi WordNet

Issues

- far less specific concepts than in the English WordNet

**Hindi WordNet:**

TOP ⟩ Noun ⟩ Inanimate ⟩ Object ⟩ Artifact ⟩ kitAb

TOP ⟩ Noun ⟩ Inanimate ⟩ Object ⟩ Artifact ⟩ mez

**English WordNet:**

entity ⟩ physical entity ⟩ object ⟩ whole unit ⟩ artifact ⟩ creation ⟩ product ⟩ piece of work ⟩ publication ⟩ book

entity ⟩ physical entity ⟩ object ⟩ whole unit ⟩ artifact ⟩ instrumentality ⟩ furnishing ⟩ piece of furniture ⟩ table
Benefits for an Urdu VerbNet

Preliminary experiments for Urdu/Hindi verbs

- Resources that we have:
  - the database from Hindi WordNet
  - a list of Urdu verbs

- out of 3,900 Hindi verbs, we have found 534 verbs in an Urdu verb list (Humayoun, 2006)

- complex predicates are included in Hindi WordNet, but not in the Urdu wordlist

- total of around 700 Urdu verbs $\rightarrow$ more than 2/3 of Urdu verbs are found

- all found verbs seem to be valid

$\rightarrow$ extract verb information from Hindi WordNet for the Urdu VerbNet
Polysemy:
An extreme case - **EAT** expressions in Hindi/Urdu (Hook and Pardeshi, 2009):

- employing 'eat' in idiomatic expressions
- about 160 **EAT** expressions for Hindi/Urdu
- variety of uses due to loan translations from Persian
Urdu Lexical Semantics

h2asan=ne kEk=ko kHAyA
h2asan.Erg cake.Acc eat.Perf.Sg.Masc
'Hasan ate the cake.'

\texttt{eat} = \texttt{\langle Agent, Theme \rangle}

inqilAbI fikar zang kHA jAEgl
revolutionary thought rust eat go.Fut
'Revolutionary thinking will gather rust.'

\texttt{eat \ (gather rust)} = \texttt{\langle Patient, Theme \rangle}

is sAl=kI mandI sheyar-bAzAr kHA gAyI
this year.Gen slowdown.Fem stockmarket eat go.Fut.Fem
'This year's slowdown wrecked (lit. devoured) the stock market.'

\texttt{eat \ (wreck)} = \texttt{\langle Agent, Theme \rangle}
How do we approach polysemy in the computational semantics?

- extensive corpus work to find polysemous verbs
- assign different thematic roles to polysemous verbs?
- put all combinations in the Urdu VerbNet, but mark the “original” use?
- analysis for all sentences, mark idiomatic and semantically ill-formed sentences as such?
Wrap up

What we have talked about:
- architecture of the Urdu LFG Grammar
- ongoing work
  - transliteration
  - discontinuous NPs
  - computational semantics
- challenges ahead

Demo
Thank you!