LingSync Tutorial

future developments

Joel Dunham

LSA 2015 Annual Meeting

January 8
Features & components under development
LingSync Spreadsheet
simple & easy to use
• more features
• import
• better export
• works offline
• powerful search
• consistency-promoting features
• text and research paper creation
• create morphological parsers using data in the database
• can deal with phonological opacity
LingSync Prototype

LingSync Spreadsheet

Morphological Parser Creator

Dative
Welcome to Dative, an application for linguistic fieldwork.
Welcome to Dative, an application for linguistic fieldwork.
LingSync Prototype

- online & offline with syncing
LingSync Prototype

- ELAN
- XML
- CSV
- .txt

- LaTeX
- JSON
- CSV

• import & export
LingSync Prototype

- customizable data structure
- encryption
LingSync Prototype

- online & offline with syncing
- import & export
- customizable data structure
- encryption
Bullshields Dunham 2008-05-01

Joel Dunham

entered: 6 years ago

last modified: 3 years ago

(1) áïssiikawaattsi
á-sä-i-ka-attsi
IMPF-wipe-foot-???
‘he is washing his feet’
Major OLD requirements

- create consistent data
- create higher-level artifacts
- retrieve data easily, quickly & precisely
Fieldwork features

• morpheme cross-referencing (consistency)
• configurable validation (consistency)
• text creation (higher-level artifacts)
• search (data retrieval)
morphologically complex form

Nitsiitsinoaayaawa  niki  aakiikoaiksi
nit-iit-ino-aa-yi-aawa  ann-iksi  aakkioan-iksi
1-loc-see-dir-3pl-pro  dem-an.pl  girl-an.pl
   na  Maanikapi  otaisinoi'sskipaiksi
ann-wa  Maanikapi  ot-a-sinoi'sskip-aa-iksi
dem-an.sg  Maanikapi  3-imp-kiss-dir-an.pl
  otsipiissa
  ot-ipii-hsi-aawa
  3-enter-conj-pro

‘I saw those girls who Maanikapi was kissing.’

lexical form

sonai'sskip
sonai'sskip
kiss
‘kiss’

no links indicate no matches

lexical form

aa
aa
DIR
‘direct suffix:’
Warning when invalid character sequence attempted

Graphical keyboard generated from orthography
Useful for endangered languages fieldwork where multiple orthographic conventions may co-exist.

User chooses an orthography for entering, viewing, and searching data.

Inputs are translated to and outputs are translated from representations in the designated storage orthography.
<table>
<thead>
<tr>
<th>Form Reference</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) ámo</td>
<td>a long time ago there were three people camping</td>
</tr>
<tr>
<td>(2) ámo, ninaa, ki, ámo, aakii, ki, ámo</td>
<td>there was a man, and a woman, and a little girl, and a dog</td>
</tr>
<tr>
<td>(3) ma, ninaa, álisina'satoom', oó'ahsistsi</td>
<td>the man went hunting and brought food back for them to eat</td>
</tr>
<tr>
<td>(4) ki, ma, aakii, mi</td>
<td>and the woman would clean up their camp and she would go after firewood and go after water</td>
</tr>
</tbody>
</table>
Creation Story

entered by: Joel Dunham
exported by: Joel Dunham

Mar 07, 2011

'a long time ago there were three people camping'  (BB, Jul 04, 2007)
The preferred interpretation of unmarked eventive predicates is reference to a past event (ref[742], ref[1065], ref[1049]).

(1)  
<table>
<thead>
<tr>
<th>$kən$</th>
<th>$q'$iy?ilx</th>
</tr>
</thead>
<tbody>
<tr>
<td>$kn$</td>
<td>$q'$y-ilx</td>
</tr>
<tr>
<td>1SG.ABS</td>
<td>dance-PR.TO</td>
</tr>
</tbody>
</table>

'I danced'

(2)  
<table>
<thead>
<tr>
<th>$k'ru$</th>
<th>$kiliŋtəx^w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k'ru$</td>
<td>$k-l-nt-x^w$</td>
</tr>
<tr>
<td>1SG.ABS</td>
<td>chase-CTR-2SG.ERG</td>
</tr>
</tbody>
</table>

'You chased me'
Creation Story

UBC Blackfoot Group

entered: 6 years ago

last modified: 3 years ago

Associated File 195: Creation Story Animation (0 bytes)

Creation Story Animation

Associated File 141: creat_story_edited_mar19.wav (20.5 MB)

creat_story_edited_mar19.wav

play audio | link to audio

Blackfoot Creation Story Animation from Joel Dunham on Vimeo.

video slideshow illustrating the story with audio, subtitles and translations

large audio recording of entire story
Give me all forms that:

[
"and",
[
[
"Form",
"syntactic_category",
"name",
"=",
"S"
],
[
"not",
[
"Form",
"grammaticality",
"=",
"*
]
],
[
"or",
[
[
"Form",
"morpheme_break",
"regex",
"-wa( \- | $)"
],
[
"Form",
"morpheme_gloss",
"regex",
"-(PROX|prox|Prox|PRX|prx)"
]
]
]
Give me all forms that:

• are sentences and ...
Give me all forms that:

• are sentences and ...

• are not ungrammatical and ...

```json
["and", [["Form", "syntactic_category", "name", "=", "S"],
  ["not", [["Form", "grammaticality", "=", "*"],
     ["or", [ [["Form", "morpheme_break", "regex", 
                "-wa( |-|$)"],
       ["Form", "morpheme_gloss", "regex", 
        "-(PROX|prox|Prox|PRX|prx)"]]]]]]]
```
Give me all forms that:

- are sentences \textit{and} ... 
- are not ungrammatical \textit{and} ... 
- either 
  - contain the morpheme shape \textit{/wa/} \textit{or} ...
Give me all forms that:

- are sentences and ...
- are not ungrammatical and ...
- either
  - contain the morpheme shape /wa/ or ...
  - contain a proximate-like gloss
### morpheme 1

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

### morpheme 2

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PRX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

### morpheme 3

<table>
<thead>
<tr>
<th>shape</th>
<th>iksi</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>iksi</td>
</tr>
</tbody>
</table>
### Scenario: you want to search for all forms that contain morpheme 1.

**morpheme 1**

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

**morpheme 2**

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PRX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

**morpheme 3**

<table>
<thead>
<tr>
<th>shape</th>
<th>iksi</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>iksi</td>
</tr>
</tbody>
</table>
Scenario: you want to search for all forms that contain morpheme 1.
<table>
<thead>
<tr>
<th>morpheme 1</th>
<th>morpheme 2</th>
<th>morpheme 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>wa</td>
<td>iksi</td>
</tr>
<tr>
<td>gloss</td>
<td>PROX</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
<td>PROX</td>
</tr>
</tbody>
</table>

/-wa( |-|$)/
**morpheme 1**

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

**morpheme 2**

<table>
<thead>
<tr>
<th>shape</th>
<th>wa</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PRX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>wa</td>
</tr>
</tbody>
</table>

**morpheme 3**

<table>
<thead>
<tr>
<th>shape</th>
<th>iksi</th>
</tr>
</thead>
<tbody>
<tr>
<td>gloss</td>
<td>PROX</td>
</tr>
<tr>
<td>category</td>
<td>NUM</td>
</tr>
<tr>
<td>sh-gl-cat</td>
<td>iksi</td>
</tr>
</tbody>
</table>
Give me all forms that have a TP or an IP node that immediately dominates a DP, which itself dominates an AP.
Morphological Parser Creator
Desiderata

• automate the morphological parsing task

• automate the evaluation of analyses
Automate parsing
Automate parsing

nítsspiyi
/nit-ihpiyi/
I-dance
AGR-VAI
Automate evaluation
Automate evaluation

nitésspiyi
/nit-ihpiyi/
I-dance
AGR-VAI

Warning!
The morphological analysis you have specified is incompatible with your morphophonological model.
Morphological Parser Creator

- Phonological rewrite rules
- Lexicon corpus
- Morphotactics corpus
- N-gram LM corpus
- Parser
Parser Architecture

{ /n-it-ihpiyi/, /nit-ihpiyi/, /nit-ihp-yi-yi/, ... }

(ranker)

[ñítsspiyi] → morphophonology

morphology → phonology

morphotactics → lexicon


Implementation

- morphophonology
  - phonology
  - morphology
  - morphotactics
  - lexicon

FST

\{ /n-it-ihpiyi/, /nit-ihpiyi/, /nit-ihp-yi-yi/, ...
\}

N-gram language model

ranker

\{ /nit-ihpiyi/, /n-it-ihpiyi/, /nit-ihp-yi-yi/, ...
\}
Specification

Extracted from OLD corpora

Ordered phonological rules

morphophonology

phonology

morphology

morphotactics

lexicon

[nítsspiyi] →

{ /n-it-ihpiyi/,
  /nit-ihpiyi/,
  /nit-ihp-yi-yi/,
  ...
}

ranker

( /nit-ihpiyi/,
  /n-it-ihpiyi/,
  /nit-ihp-yi-yi/,
  ...
)
# Tests from Phonological Rules Appendix B of Frantz's Grammar

#

#test nit-waanist-ok-wa -> nitaanikka
#test nit-waanist-aa-wa -> nitaanistaawa
#test nit-siksip-aa-wa -> nitssiksipaawa
#test nit-siksip-aa-wa -> nitssiksipaawa
#test nit-áak-yo'kaa -> nitáakso'kaa
#test nit-ssikópii -> nitssssikópii
#test á-sínaaki-wa -> áísínaakiwa
#test n-ikáá-ssikópii -> nikáíssikópii
#test káta'-simi-wa -> kátaí'simiwa
#test áak-oto-apinnii-wa -> áakotaapinniiwa
#test áak-oto-apinnii-wa -> áakotapinniiwa
# Phonological Rules (cf. Frantz 1991, pp. 152--155)

# 1. C1-C2 -> C2C2
# Gemination Original
# Note that this rule is actually quite contrived since it requires that the
# adjacent plosives be MADE adjacent by a previously applying rule. If they
# were underlyingly adjacent, then the so-called ``stop-stop epenthesis'' rule
# would insert /oh/ or /i/ between them. Note: geminationGeneral is not being
# used.

define geminationGeneral [  
  [ plosives "-" -> p || _ p ] .o.  
  [ plosives "-" -> t || _ t ] .o.  
  [ plosives "-" -> k || _ k ] ];

define gemination [  
  [ an ist -> an ik || _ "-" k ] .o.  
  [ t -> k || _ "-" k ] ];

# 2. Iit -> Ist
# s-Insertion (assumes that "breaking I" is a phoneme)
# Note: since the dictionary does not use /I/, this rule is not being used.
# Instead it is assumed that words like waanIt 'say to' are represented
# phonemically as /waanist/ and there are special lexical rules to deal with
# Blackfoot Parser Results

<table>
<thead>
<tr>
<th>Component</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>17%</td>
</tr>
<tr>
<td>F-score</td>
<td>0.4</td>
</tr>
<tr>
<td>morphophonology</td>
<td>60%</td>
</tr>
<tr>
<td>ranker</td>
<td>28%</td>
</tr>
</tbody>
</table>
# Blackfoot Parser Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>17%</td>
</tr>
<tr>
<td>F-score</td>
<td>0.4</td>
</tr>
<tr>
<td>morphophonology</td>
<td>60%</td>
</tr>
<tr>
<td>ranker</td>
<td>28%</td>
</tr>
</tbody>
</table>
MPC Next Steps

- GUI for *creating* parsers in Dative
- GUI for *using* parsers in Dative
- ranked autocomplete suggestions
- Improve parsers for Blackfoot
- Build parsers for other languages
Review of in-development Dative features

- Desktop-like GUI
- import/export for interoperability
- text creation: narratives, research papers, etc.
- better search: morphological, syntactic, regular expression macros
- Morphological Parser Creator integration
- consistency: inventory-based validation, orthography conversion
Appeal for developer contributions
Building web applications is fun!

- CoffeeScript is JavaScript improved à la Python and Ruby
- Testable Coffee/JavaScript
- Task runners for compiling, testing, building, etc.
- Chrome developer tools for debugging
```coffeescript
# LoginDialogView
#
# This is a dialog box for authentication, i.e., logging in, out, and
# requesting a new password. It is a draggable box created using jQueryUI's
# .dialog()

class LoginDialogView extends BaseView

template: loginDialogTemplate

initialize: ->
  @listenTo Backbone, 'authenticate:fail', @_authenticateFail
  @listenTo Backbone, 'authenticate:end', @_authenticateEnd
  @listenTo Backbone, 'authenticate:success', @_authenticateSuccess
  @listenTo Backbone, 'loginDialog:toggle', @toggle
  @listenTo Backbone, 'logout:success', @logoutSuccess
  @listenTo @model, 'change:loggedIn', @_disableButtons

  @activeServerView = new ActiveServerView
    model: @model, width: 139, label: 'Server = '

  events:
    'keyup .dative-login-dialog-widget .username': 'validate'
    'keydown .dative-login-dialog-widget .username': 'submitWithEnter'
    'keyup .dative-login-dialog-widget .password': 'validate'
    'keydown .dative-login-dialog-widget .password': 'submitWithEnter'

render: ->
  @$el.append template()
  @renderActiveServerView()
  @$source = @$ '.dative-login-dialog' # outer DIV from template
  @$target = @$ '.dative-login-dialog-target' # outer DIV to which jQueryUI dialog appends
  @dialogify()
  @_disableButtons()

  @

renderActiveServerView: ->
  @activeServerView.setElement @$('li.active-server').first()
  @activeServerView.render()
  @rendered @activeServerView
```
Thank you
Acknowledgements

- Alan Bale (Concordia)
- Gina Cook (iLanguage Lab Ltd)
- Jessica Coon (McGill)
- M.E. Cathcart (U Delaware)
- Theresa Deering (Visit Scotland, iLanguage Lab Ltd)
- Josh Horner (Amilia, iLanguage Lab Ltd)
- Yuliya Manyakina (McGill)
- Elise McClay (McGill)
- Gretchen McCulloch (McGill)
- Hisako Noguchi (Concordia)
- Michael Wagner (McGill)
- Jesse Pollak (Pomona College)
- Tobin Skinner (Acquisio, iLanguage Lab Ltd)
- Xianli Sun (Miami University)
- More...
References

Images

- Bridget Holmes: a Nonagenarian Housemaid. John Riley. Public domain, via Wikimedia Commons
- WAV audio icon. By zeus (www.zeusbox.org) GPL (http://www.gnu.org/licenses/gpl.html), via Wikimedia Commons