Iango Documentation

Release 0.1

Michael Young

July 18, 2016

Contents

CHAPTER 1

Installation

1.1 Install package with pip

```
pip install lango
```

1.2 Download Stanford Models and Parser

Make sure you have Java installed for the Stanford parser to work.

Download Stanford Parser

1.3 Set Environment Variables

Set environment variables for STANFORD_PARSER and STANFORD_MODELS to where you downloaded the parser.

```
import os
os.environ['STANFORD_PARSER'] = 'stanford-parser-full-2015-12-09'
os.environ['STANFORD_MODELS'] = 'stanford-parser-full-2015-12-09'
```

CHAPTER 2

Matching

Matching is done by comparing a set rules and matching it with a parse tree. You can see parse trees for sentences from examples/parser_input.py.

The set of rules is recursive and can match multiple parts of the parse tree.

Rules can be broken down into smaller parts: - Tag - Token - Token Tree - Rules

2.1 Tag

A tag is a POS (part of speech) tag to match. A list of POS tags used by the Stanford Parser can be found here.

```
Format:
tag = string

Example:
'NP'
'VP'
'PP'
```

2.2 Token

A token is a string comprising of a tag with modifiers/options.

```
Format:
token = tag:match_label-opts

Example:
'NP:subject-o'
'VP'
'NP:np'
```

2.3 Token Tree

A token tree is a recursive tree of tokens. The tree matches the structure of a parse tree.

```
Format:
token_tree = ( token token_tree token_tree ... )

Examples:
'( NP ( DT ) ( NP:subject-o ) )'
'( NP )'
'( PP ( TO=to ) ( NP:object-o ) )'
```

2.4 Rules

Rules are a dictionary of token trees to dictionaries of matching labels to a nested set of rules.

```
Format:
rules = {token_tree: {match_label: rules}}

Example:
{
    '( S ( NP:np ) ( VP ( VBD:action-o ) ( PP:pp ) ) )': {
        'np': {
            '( NP:subject-o )': {}
        },
        'pp': {
            '( PP ( TO=to ) ( NP:to_object-o ) )': {},
            '( PP ( IN=from ) ( NP:from_object-o ) )': {},
        }
    },
}
```

When matching a rule to a parse tree, the token tree is first matched. Then, all matching tags are matched to nested rules corresponding to their matching label.

All nested match labels must have a subrule match or the rules will not match.

The first rule to match is returned so the order of match is based on key ordering (use OrderedDict if order matters). Once a rule is matched, it calls the callback function with the context as arguments.

2.5 Example

Suppose we have the sentence "Sam ran to his house" and we wanted to match the subject ("Sam"), the object to ("his house") and the action ("ran").

Sample parse tree for "Sam ran to his house" from the Stanford Parser.

```
(S
  (NP
    (NNP Sam)
    )
  (VP
    (VBD ran)
      (PP
        (TO to)
        (NP
          (PRP$ his)
          (NN house)
          )
```

```
        )
    )
  )
```

Simplified image of tree:

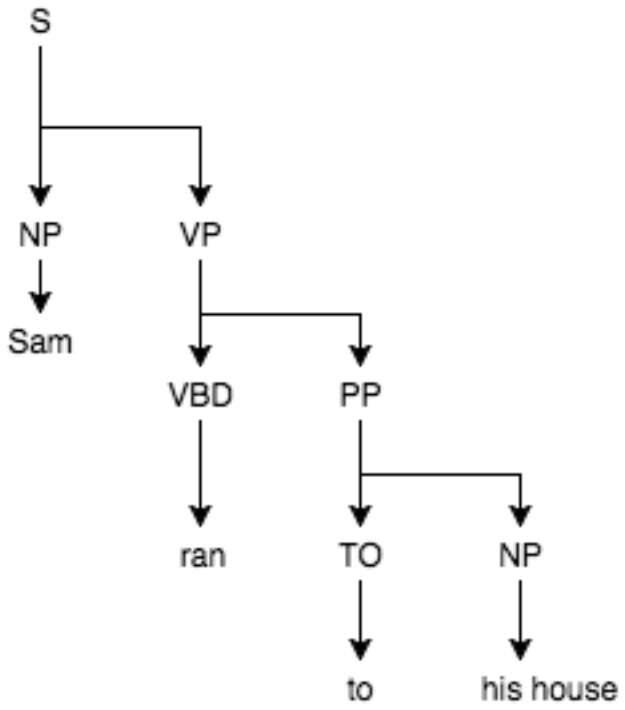


Fig. 2.1: tree

```
Matching:
Parse Tree:
(S (NP (NNP Sam) ) (VP (VBD ran) (PP (TO to) (NP (PRP$ his) (NN house))))

Matched token tree: '( S ( NP:np ) ( VP ( VBD:action-o ) ( PP:pp ) ) )'
Matched context:
  np: (NP (NNP Sam))
  action-o: 'ran'
  pp: (PP (TO to) (NP (PRP$ his) (NN house)))
```

Rule for '(S (NP:np) (VP (VBD:action-o) (PP:pp)))':

Matching 'NP' matches the whole NP tree and converts to a word:

```
Matched token tree for np: '( NP:subject-o )'
Matched context:
  subject-o: 'Sam'
```

Matching 'PP' requires matching the nested rules:

```
Match token tree for pp: '( PP ( TO=to ) ( NP:to_object-o ) )'
Match context:
  object-o: 'his house'
```

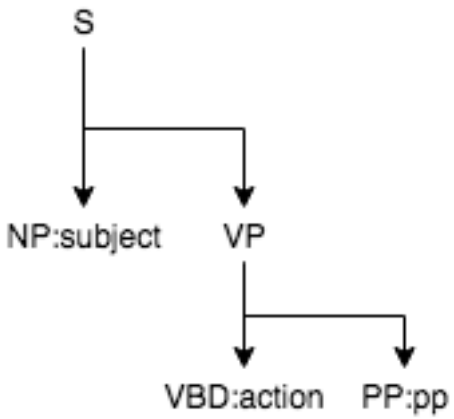


Fig. 2.2: tree

```
Match token tree for pp: '( PP ( IN=from ) ( NP:from_object-o ) )'
No match found
```

PP of the sample sentence:

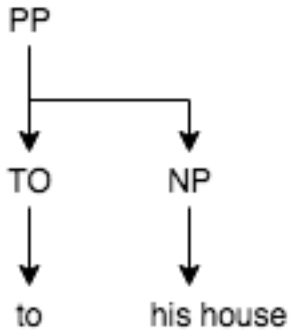
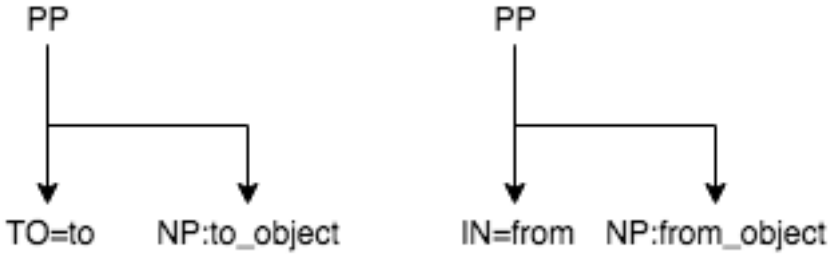


Fig. 2.3: tree

Nested PP rules:



Only the first rule matches for 'PP'.

Now that we have a match for all nested rules, we can return the context:

```
Returned context:
  action: 'ran'
  subject: 'sam'
  to_object: 'his house'
```

Full code:

```
from lango.parser import StanfordLibParser
from lango.matcher import match_rules

parser = StanfordLibParser()

rules = {
  '( S ( NP:np ) ( VP ( VBD:action-o ) ( PP:pp ) ) )': {
    'np': {
        '( NP:subject-o )': {}
    },
    'pp': {
        '( PP ( TO=to ) ( NP:to_object-o ) )': {},
        '( PP ( IN=from ) ( NP:from_object-o ) )': {}
    }
  }
}

def fun(subject, action, to_object=None, from_object=None):
    print "%s,%s,%s,%s" % (subject, action, to_object, from_object)

tree = parser.parse('Sam ran to his house')
match_rules(tree, rules, fun)
# output should be: sam, ran, his house, None

tree = parser.parse('Billy walked from his apartment')
match_rules(tree, rules, fun)
# output should be: billy, walked, None, his apartment
```

CHAPTER 3

Reference

3.1 lango package

3.1.1 Submodules

lango.matcher module

`lango.matcher.`**`get_object`**(*tree*)

Get the object in the tree object.

Method should remove unnecessary letters and words:

```
the
a/an
's
```

Parameters **tree** (*Tree*) – Parsed tree structure

Returns Resulting string of tree `(Ex: "red car")`

`lango.matcher.`**`get_proper_word`**(*tree*)

Get unmodified words in the tree object

Parameters **tree** (*Tree*) – Parsed tree structure

Returns Resulting string of tree `(Ex: "The red car")`

`lango.matcher.`**`get_tokens`**(*tokens*)

Recursively gets tokens from a match list

Parameters **tokens** – List of tokens ['(', 'S', '(', 'NP', ')', ')']

Returns Stack of tokens

`lango.matcher.`**`get_word`**(*tree*)

Get the exact words in lowercase in the tree object.

Parameters **tree** (*Tree*) – Parsed tree structure

Returns Resulting string of tree `(Ex: "the red car")`

`lango.matcher.`**`match_rules`**(*tree*, *rules*, *fun*)

Matches a Tree structure with the given query rules.

Query rules are represented as a dictionary of template to action. Action is either a function, or a dictionary of subtemplate parameter to rules:

```
rules = { 'template' : { 'key': rules } }
      | { 'template' : {} }
```

Parameters

- **tree** (*Tree*) – Parsed tree structure
- **rules** (*dict*) – A dictionary of query rules
- **fun** – Function to call

Returns Result of function call with context or None if nothing matched

`lango.matcher.`**`match_rules_context`**(*tree*, *rules*, *parent_context={}*)

Recursively matches a Tree structure with rules and returns context

Parameters

- **tree** (*Tree*) – Parsed tree structure
- **rules** (*dict*) – See match_rules
- **parent_context** (*dict*) – Context of parent call

Returns Context matched dictionary of matched rules or None if no match

Return type dict

`lango.matcher.`**`match_template`**(*tree*, *template*, *args=None*)

Check if match string matches Tree structure

Parameters

- **tree** (*Tree*) – Parsed Tree structure of a sentence
- **template** (*str*) – String template to match. Example: "(S (NP))"

Returns If they match or not

Return type bool

`lango.matcher.`**`match_tokens`**(*tree*, *tokens*, *args*)

Check if stack of tokens matches the Tree structure

Special matching rules that can be specified in the template:

```
':label': Label a token, the token will be returned as part of the context with key 'label'.
'-@': Additional single letter argument determining return format of labeled token. Valid option
    '-w': Return token as word
    '-o': Return token as object
'=word|word2|....|wordn': Force match the token words
'$': Force match the number of tokens
```

Parameters

- **tree** – Parsed tree structure
- **tokens** – Stack of tokens

Returns Boolean if they match or not

lango.parser module

class `lango.parser.`**`OldStanfordLibParser`**

Bases: *`lango.parser.Parser`*

For StanfordParser < 3.6.0

`parse`(*line*)

Returns tree objects from a sentence

Parameters **`line`** – Sentence to be parsed into a tree

Returns Tree object representing parsed sentence

class `lango.parser.`**`Parser`**

Abstract Parser class

`parse`(*sent*)

class `lango.parser.`**`StanfordLibParser`**

Bases: *`lango.parser.OldStanfordLibParser`*

For StanfordParser == 3.6.0

3.1.2 Module contents

Lango is a natural language framework for matching parse trees and modeling conversations.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

l

G

L

M

O

P

S