Python toolkit for subregular languages

Alëna Aksënova

Linguistics department @ SBU

IACS Research Day

IACS @ SBU

April 4, 2018
Today, I will…

- talk about the subregular hierarchy
  very useful for linguistics and not only
- explain what is the kist toolkit
  provides different subregular tools
- update on what has been done so far
  learners, scanners, generators for several language classes
Outline

1. Preliminaries
2. Subregular approach
3. Subregular toolkit
4. Summary
For linguistics, it is important to be able to formalize patterns and to study properties of the system.
For linguistics, it is important to be able to formalize patterns and to study properties of the system.

Among others, it can give us

- typological predictions;
- language models;
- cognitive insights.
For formal language theory, it is important to know different types of patterns and their complexity.
The side of formal language theory

For formal language theory, it is important to know different types of patterns and their complexity.

Among others, it can give us

- formal properties of the patterns;
- generating devices they require;
- cognitive applicability.
Subregular languages

The formal class of subregular languages is where they meet.
Subregular languages

The formal class of subregular languages is where they meet.

- **FLT**: we know properties of subregular languages.
- **Linguistics**: and we know how to apply them!
The Chomsky Hierarchy of String Languages

- **recursively enumerable**
- **context-sensitive**
- **context-free**
- **regular**
- *(finite)*
The Chomsky Hierarchy of String Languages

- recursively enumerable
- context-sensitive
- context-free
- regular
  - (finite)

Finite state automaton
Push-down automaton
Linear-bounded non-deterministic Turing machine
Turing machine
Regular languages

Regular languages = FSA recognized = MSO definable
Regular languages

Regular languages = FSA recognized = MSO definable

\[ a^+ b^+ \]
Regular languages

Regular languages = FSA recognized = MSO definable

\[ a^+ b^+ \]
Regular languages

Regular languages = FSA recognized = MSO definable

\[
\begin{align*}
& a^+ b^+ \\
& a^n b^n
\end{align*}
\]
Regular languages

Regular languages = FSA recognized = MSO definable

\[ a^+ b^+ \]

\[ a^m b^n \]

1 \[ \rightarrow \] 2 \[ \rightarrow \] 3

Regular
Regular languages

Regular languages = FSA recognized = MSO definable

Regular

Regular languages = FSA recognized = MSO definable

Regular

Regular languages = FSA recognized = MSO definable
Regular languages

Regular languages = FSA recognized = MSO definable

\[ a^+ b^+ \]

... no general FSA can be constructed
The class of regular languages can be decomposed into **subregular hierarchy**

- Introduced by McNaughton & Papert (1971)
- Expanded by numerous researchers
The class of regular languages can be decomposed into **subregular hierarchy**

- Introduced by McNaughton & Papert (1971)
- Expanded by numerous researchers

The most fruitful classes for the NL are:

- **strictly local languages** (SL)
- **tier-based strictly local languages** (TSL)
- **strictly piecewise languages** (SP)
(T)SL intuitions

**Intuition:** TSL is a $n$-gram model on steroids.
Intuition: TSL is a \( n \)-gram model on steroids.

\*ab constraint:

\[ \begin{align*}
\text{\textit{a}} & \quad \text{\textit{a}} & \quad \text{\textit{c}} & \quad \overline{\text{\textit{a}}} & \quad \overline{\text{\textit{b}}} & \quad \text{\textit{a}} & \quad \text{\textit{c}} \\
\text{\textit{a}} & \quad \text{\textit{a}} & \quad \text{\textit{c}} & \quad \text{\textit{c}} & \quad \text{\textit{b}} & \quad \text{\textit{a}} & \quad \text{\textit{a}} \\
\text{\textit{a}} & \quad \text{\textit{a}} & \quad \text{\textit{c}} & \quad \text{\textit{c}} & \quad \text{\textit{b}} & \quad \text{\textit{a}} & \quad \text{\textit{a}}
\end{align*} \]

\( n \)-gram model or SL grammar lists the (im)possible sequences of elements.
**Intuition:** TSL is a \( n \)-gram model on steroids.

\( n \)-gram model or SL grammar lists the (im)possible sequences of elements.

TSL grammar is a SL grammar of a certain subset of the alphabet (*tier alphabet*). Other elements are ignored.
Possible compounding patterns

**Turkish**

1. kapı
   ‘gate’

2. bahçe kapı-şi
   ‘garden gate’

3. türk bahçe kapı-şi
   ‘Turkish garden gate’
Possible compounding patterns

Turkish

1. kapı  
   ‘gate’

2. bahçe kapı-\textit{sı}  
   ‘garden gate’

3. türk bahçe kapı-\textit{sı}  
   ‘Turkish garden gate’

\textit{word-(word}^{+}\textit{-sı)}
Possible compounding patterns

<table>
<thead>
<tr>
<th>Turkish</th>
<th>Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. kapı</td>
<td>1. par</td>
</tr>
<tr>
<td>‘gate’</td>
<td>‘steam’</td>
</tr>
<tr>
<td>2. bahçe kapı-şî</td>
<td>2. par-o-khod</td>
</tr>
<tr>
<td>‘garden gate’</td>
<td>‘steam boat’</td>
</tr>
<tr>
<td>3. türk bahçe kapı-şî</td>
<td>3. par-o-khod-o-voz</td>
</tr>
<tr>
<td>‘Turkish garden gate’</td>
<td>‘steam boat carrier’</td>
</tr>
</tbody>
</table>

word-(word⁺-şî)
Possible compounding patterns

<table>
<thead>
<tr>
<th>Turkish</th>
<th>Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. kapı</td>
<td>1. par</td>
</tr>
<tr>
<td>‘gate’</td>
<td>‘steam’</td>
</tr>
<tr>
<td>2. bahçe kapı-şi</td>
<td>2. par-o-khod</td>
</tr>
<tr>
<td>‘garden gate’</td>
<td>‘steam boat’</td>
</tr>
<tr>
<td>3. türk bahçe kapı-şi</td>
<td>3. par-o-khod-o-voz</td>
</tr>
<tr>
<td>‘Turkish garden gate’</td>
<td>‘steam boat carrier’</td>
</tr>
</tbody>
</table>

\[ \text{word-} (\text{word}^+ - \text{şi}) \]

\[ (\text{word-o})^* - \text{word} \]
Possible compounding patterns

**Turkish**

1. kapı
   ‘gate’
2. bahçe kapı-si
   ‘garden gate’
3. türk bahçe kapı-si
   ‘Turkish garden gate’

\[
\text{word-} (\text{word}^+ - \text{sI})
\]

**Russian**

1. пар
   ‘steam’
2. пар-o-khod
   ‘steam boat’
3. пар-o-khod-o-voz
   ‘steam boat carrier’

\[
(\text{word-o})^* - \text{word}
\]

Both patterns are SL!
Impossible compounding pattern

\[ \text{word-} (\text{word}^+ - \text{sI}) \quad (\text{word-o})^* - \text{word} \]
Impossible compounding pattern

word-(word$^+_{-SI}$) \hspace{1cm} (word-o)$^*$-word

word-(word$^n_{-x^n}$)
Impossible compounding pattern

\[
\text{word-} (\text{word}^+-\text{sI}) \\
\downarrow \\
\text{word-} (\text{word}^n-x^n) \\
\downarrow \\
\text{(word-o)}^*\text{-word}
\]

This pattern is unattested!
Complexity: context-free.
What are the other applications?

Applications

- Linguistics
  - Sounds (Heinz 2010)
  - Words (Aksënova et al. 2016)

- Robotics
  - Sentences (Graf & Heinz 2015)

- Experiments with NN
  - Meaning (Graf 2017)
  - (Rawal et al. 2011)
**kist**: **kist implementing subregular toolkit**

**Motivation**: help researchers to avoid manual burden by providing a toolkit that contains necessary subregular functions.

- Python 3 (will be available via pip)
- Open source
- Available on GitHub
  
  # https://github.com/loisetoil/slp
What is available already?

Languages (positive and negative versions):
- Strictly local
- Tier-based strictly local
- Strictly piecewise
What is available already?

**Languages (positive and negative versions):**
- Strictly local
- Tier-based strictly local
- Strictly piecewise

**Tools available:**
- **learn**
  extracts the grammar
- **generate_sample**
  generates a data sample for a given grammar
- **scan**
  tells whether a string belongs to a language or not
- **fsmize**
  generates a FSA that corresponds to the grammar
- **generate_polarity**
  changes the polarity of the grammar
Subregular toolkit: examples

**Initialization (positive SL)**

```python
>>> a = PosSL()
>>> a.data = ['abab', 'ab']
>>> a.k = 2
```
Subregular toolkit: examples

**Initialization (positive SL)**

```python
>>> a = PosSL()
>>> a.data = ['abab', 'ab']
>>> a.k = 2
```

**Learning (positive SL)**

```python
>>> a.learn()
>>> a.grammar
[(>'a'), ('b', '<'), ('a', 'b'), ('b', 'a')]
```
Subregular toolkit: examples [cont.]

**Scanning (positive SL)**

```plaintext
>>> a.scan('ababab')
True
>>> a.scan('aba')
False
```
Subregular toolkit: examples [cont.]

**Scanning (positive SL)**

```python
>>> a.scan('ababab')
True
>>> a.scan('aba')
False
```

**Changing polarity (positive → negative SL)**

```python
>>> a.change_polarity()
>>> a.grammar
[('a', 'a'), ('b', 'b'), ('>', '<'), ('a', '<'), ('>', 'b')]
```
Subregular toolkit: examples [cont.]

**Learning (negative TSL)**

```python
>>> b = NegTSL()
>>> b.data = ['abaa', 'aab', 'ba', 'b']
>>> b.learn()
>>> b.grammar
[(‘b’, ‘b’), (‘>’, ‘<’)]
>>> b.tier
[‘b’]
```
## Subregular toolkit: examples [cont.]

### Learning (negative TSL)

```python
>>> b = NegTSL()
>>> b.data = ['abaa', 'aab', 'ba', 'b']
>>> b.learn()
>>> b.grammar
[('b', 'b'), ('>', '<')]
>>> b.tier
['b']
```

### Generating sample (negative TSL)

```python
>>> b.generate_sample(3)
>>> b.data_sample
['abaaa', 'b', 'aaba']
```
What is currently happening?

Currently being implemented:

- Probabilistic versions of SL, TSL, SP
- Subregular transductions
  - the grammars discussed above behave as acceptors: they either accept the given string or not;
  - transducers re-write the given string while reading it.
Summary

This subregular toolkit allows:

- test ideas currently available in literature;
- explore new methods to model NL;
- seek out new ways to improve the results.
Thank you!

What I cannot create, I do not understand.

Richard Feynman
Aksènova, Alëna, Thomas Graf and Sedigheh Moradi (2016)
Morphotactics as Tier-Based Strictly Local Dependencies.
In Proceedings of SIGMorPhon 2016.

Avcu, Enes, Chihiro Shibata and Jeffrey Heinz (2017)
Subregular complexity and deep learning.

Graf, Thomas and Jeffrey Heinz (2015)
Commonality in Disparity: The Computational View of Syntax and Phonology.

Heinz, Jeffrey (2010)
Learning long-distance phonotactics.

Graf, Thomas (2017)
The subregular complexity of monomorphemic quantifiers.
Ms., Stony Brook University.

McNaughton, Robert and Seymour Papert (1971)
Counter-Free Automata.

Rawal, Chetan, Herbert Tanner and Jeffrey Heinz (2011)
(Sub)regular Robotic Languages.