Unsupervised Language Learning in OpenCog

Alex Glushchenko, Andres Suarez, Anton Kolonin, Ben Goertzel, Claudia Castillo, Man Hin Leung, Oleg Baskov
Presenter: Anton Kolonin
anton@singularitynet.io

OpenCog
https://opencog.org/

SingularityNET
https://singularitynet.io

Hanson Robotics
http://www.hansonrobotics.com/
Grammar Learning from scratch - programmatically
Language as beneficial evolutionary property of generic intelligence

Biological transfer of information

Social transfer of information

Speed of light

Learnable Language

Hardcoded by evolution
Language Learning Environment

Intelligent being

- Learning
- Comprehension
- Generation

Grammar & Ontology

Social environment

- Perception
- Expression

- Supervision
- Experiential feedback
- Self-reinforcement
Project goal and applications

- Grammar learning from scratch - programmatically
- Grammar extension/customization for specific domains
- Building dictionaries and patterns for NLP applications
- Parsing texts for NLP applications
- Grammar checking (more than spell checking)
Constraints of the currently explored approach

- Controlled corpora
- Using Link Grammar formalism
- Relying on MST parses
- No account for morphology
- Self-reinforcement with parse-ability
- Test against training data
Link Grammar and Disjuncts

An illustration of Link Grammar connectors and disjuncts. The connectors are the jigsaw-puzzle-shaped pieces; connectors are allowed to connect only when the tabs fit together. A disjunct is the entire (ordered) set of connectors for a word. As lexical entries appearing in a dictionary, the above would be written as

\[
\begin{align*}
\text{a the:} & \quad D^+; \\
\text{cat snake:} & \quad D^- \& (S^+ \text{ or } O^-); \\
\text{Mary:} & \quad O^- \text{ or } S^+; \\
\text{ran:} & \quad S^-; \\
\text{chased:} & \quad S^- \& O^+;
\end{align*}
\]

Note that although the symbols ‘‘&’’ and ‘‘or’’ are used to write down disjuncts, these are not Boolean operators, and do not form a Boolean algebra. They do form a non-symmetric compact closed monoidal algebra. The diagram below illustrates puzzle pieces, assembled to form a parse:
MST Parses vs. Link Parses

Corpus:

There is a snake
The boy saw a snake
The dog chased a snake
The cat chased a snake

Link Parse:

```
[linkparser> the cat chased a snake
Found 2 linkages (2 had no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS= 0.00 LEN=9)
       +-------->WV--------+
       +------Wd------+
       | +Ds**Ct----Ss----+ +Ds**Ct
       |                |    |
       |  LEFT-WALL the  |    | a
       |  cat.n chased.v-|    | a
       |   a snake.n  |    | n
```

MST Parse:

```
the cat chased a snake
0 ###LEFT-WALL### 2 cat
0 ###LEFT-WALL### 3 chased
1 the 2 cat
2 cat 3 chased
3 chased 5 snake
4 a 5 snake
```
MST-Parser parsing SMS text

- SMS corpus in English by NUS
- Pre-processed and MST-parsed

LEFT-WALL Me very hungry ...

LEFT-WALL Haha, can go for one whole stretch ...

¹ https://www.kaggle.com/rtatman/the-national-university-of-singapore_sms-corpus#smsCorpus_en_2015.03.09_all.json
Unsupervised language learning framework in OpenCog
Grammar Learner Pipeline

Grammar Learner

Space Formation
- Words
- Connectors
- Disjuncts

Dimension Reduction
- SVD
- TBD

Clustering
- K-means
- Identical Lexical Entires
- Aggregative
- Dissociative
- TBD

Classification

Category Generalization

Grammar Induction

Rule Generalization

Grammar
Results: Word-Sense Disambiguation
Using AdaGram\(^1\) we disambiguate our POC-English corpus without supervision.

Two ambiguous words in corpus, with only two senses each:

```
board
saw
```

After parameter tuning, we found two promising results:

```
mom saw@a dad with a saw@b.
mom@a saw@a dad@b with a@c saw@b.
```

\(^1\) https://github.com/glicerico/AdaGram/tree/take_sentences
## Results: Corpora with PA & PQ

<table>
<thead>
<tr>
<th>Name</th>
<th>Language</th>
<th>Volume (bytes)</th>
<th>Unique Words</th>
<th>Word Instances</th>
<th>Instances per Word</th>
<th>Sentences</th>
<th>Average sentence length</th>
<th>Best PA</th>
<th>Best PQ</th>
<th>Best PQ/PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>POC-Turtle</td>
<td>Turtle</td>
<td>203</td>
<td>13</td>
<td>36</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>POC-English (with no ambiguity)</td>
<td>English</td>
<td>789</td>
<td>25</td>
<td>132</td>
<td>5</td>
<td>36</td>
<td>4</td>
<td>100%</td>
<td>68%</td>
<td>68%</td>
</tr>
<tr>
<td>POC-English (with ambiguity)</td>
<td>English</td>
<td>1,794</td>
<td>55</td>
<td>388</td>
<td>7</td>
<td>88</td>
<td>4</td>
<td>97%</td>
<td>70%</td>
<td>72%</td>
</tr>
<tr>
<td>Child Directed Speech (br-text + brent9mos)</td>
<td>English</td>
<td>633,151</td>
<td>4,717</td>
<td>130,109</td>
<td>27</td>
<td>38,181</td>
<td>3</td>
<td>75%</td>
<td>60%</td>
<td>80%</td>
</tr>
<tr>
<td>Gutenberg Children Books</td>
<td>English</td>
<td>30,118,309</td>
<td>54,054</td>
<td>2,695,151</td>
<td>50</td>
<td>207,130</td>
<td>13</td>
<td>99%</td>
<td>62%</td>
<td>63%</td>
</tr>
</tbody>
</table>

Across corpora with the best found configurations, artificially learned grammar makes it possible to parse 75-100% of text (Parse-ability or PA), having 60-100% of it parsed properly (Parse-quality or PQ), with properly parsed fraction of parsed text above 63%.
Results: MST Parses

- Better parses – better results (generally, for MST parses - especially)
- Good parses – does not mean good results (for “expected” parses)
Results: PA and PQ correlation

Better Parse-ability (PA) implies better Parse-quality (PQ)
Results: Categorial spaces (POC)

POC-Turtle (Connectors)

POC-English (Connectors)

POC-Turtle (Disjuncts)

POC-English (Disjuncts)
Results: Category trees (POC-English)
Results: Categories and PA and PQ

- Fewer categories – better Parse-ability (PA) and Parse-quality (PQ)

- Number of categories vary from upper limit of natural number of identical lexical entries (tens to thousands) to 4-6 basic “parts of speech” - randomly due to uncontrolled nature of K-means clustering which has to be replaced with controlled aggregative/dissociative clustering
## Results: Grammar Learning Algorithms

Comparing across all corpora

<table>
<thead>
<tr>
<th>Grammar Learning Algorithm</th>
<th>Parse-ability (PA)</th>
<th>Parse-quality (PQ)</th>
<th>PQ/PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space of Connectors, SVD, K-means, Rules by Connectors</td>
<td>82%</td>
<td>61%</td>
<td>75%</td>
</tr>
<tr>
<td>Space of Connectors, SVD, K-means, Rules by Disjuncts</td>
<td>64%</td>
<td>49%</td>
<td>76%</td>
</tr>
<tr>
<td>Space of Disjuncts, SVD, K-means, Rules by Disjuncts</td>
<td>61%</td>
<td>47%</td>
<td>77%</td>
</tr>
<tr>
<td>Space of Disjuncts, No dimension reduction, Identical Lexical Entries, Rules by Disjuncts</td>
<td>54%</td>
<td>44%</td>
<td>81%</td>
</tr>
</tbody>
</table>

- Connectors - more generalized, less categories, less strict parsing
- Disjuncts - less generalized, more categories, more precise parsing
The next steps

- Incremental probabilistic assessment of parses, clustering, grammar induction
- Fine/tuning MST-parsing parameters or change parsing approach
- Quality assessment procedure (fitness function) idea tolerant to overfitting
- Pipeline made available as CLI tool, web service and SingularityNET adapter
Thank you and visit us at:
http://langlelearn.singularitynet.io/

Stay in touch:

Ben Goertzel
ben@singularitynet.io
Cassio Pennachin
cassio@singularitynet.io
Anton Kolonin
anton@singularitynet.io

OpenCog
https://opencog.org/

SingularityNET
https://singularitynet.io

Hanson Robotics
http://www.hansonrobotics.com/