

# 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](mailto:anton@singularitynet.io)



OpenCog

<https://opencog.org/>



SingularityNET

<https://singularitynet.io>



<http://www.hansonrobotics.com/>

# Grammar Learning from scratch - programmatically

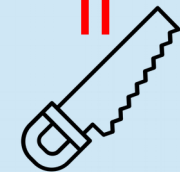


**PRONOUN**



**VERB**

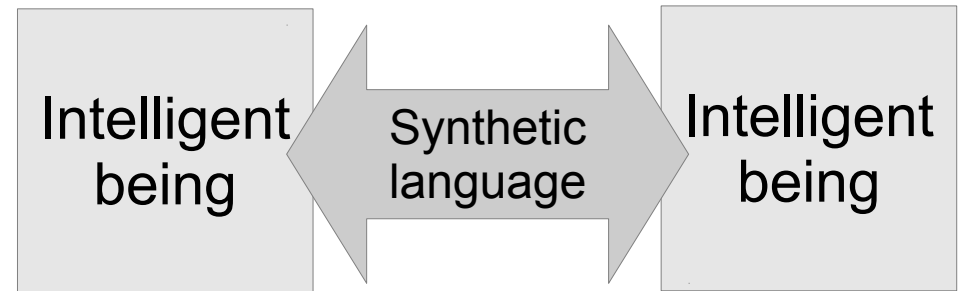
**ARTICLE**



**NOUN**

# Language as beneficial evolutionary property of generic intelligence

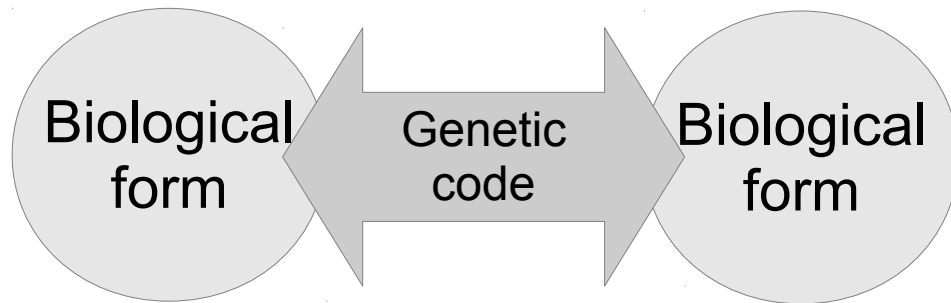
## Social transfer of information



Speed of light

**Learnable Language**

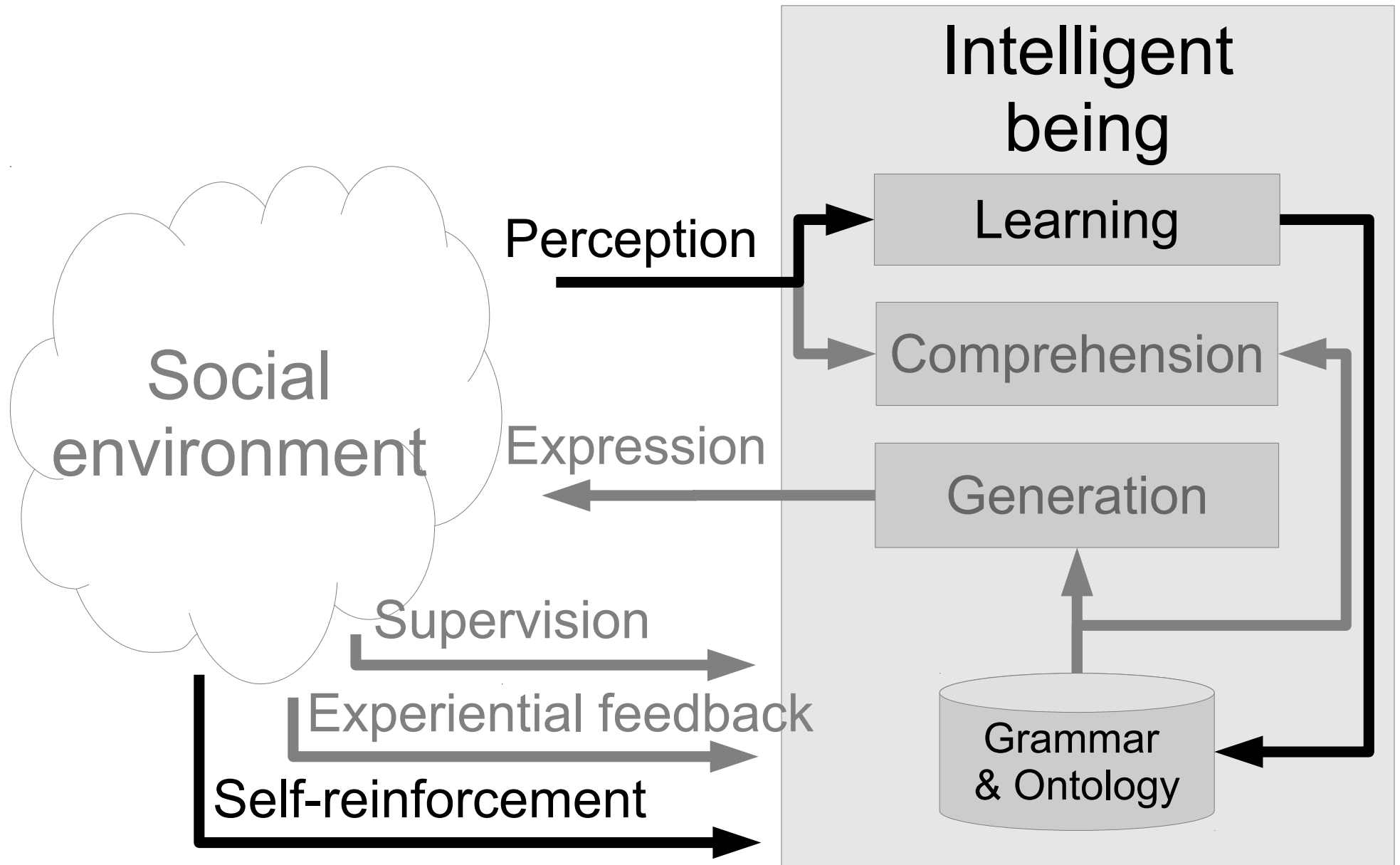
## Biological transfer of information



Change of generations

**Hardcoded by evolution**

# Language Learning Environment



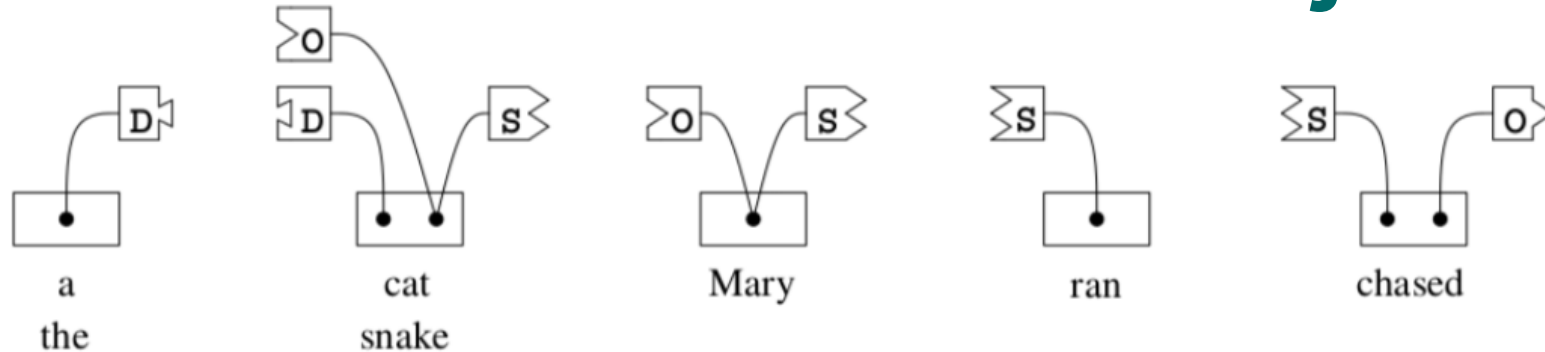
# Project goal and applications

- 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**

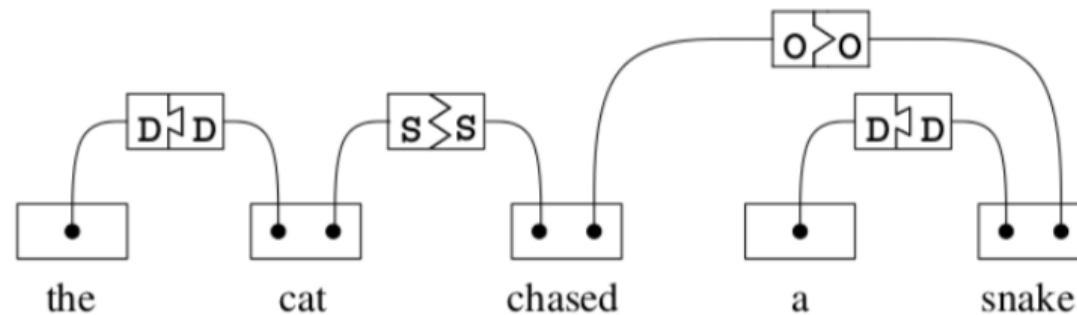
# Link Grammar and Disjuncts



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

```
a the: D+;
cat snake: D- & (S+ or O-);
Mary: O- or S+;
ran: S-;
chased S- & O+;
```

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:

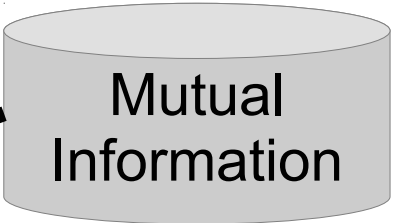
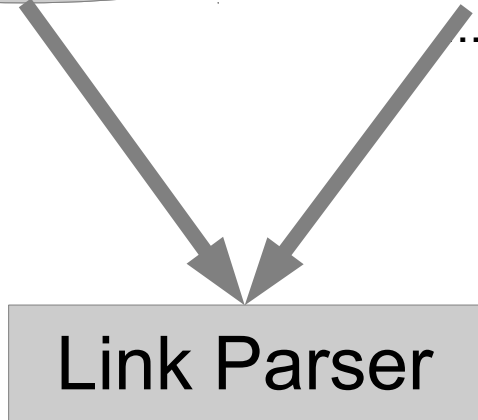


B. Goertzel,  
L. Vepstas,  
2014

# 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-----+   +-----Os-----+
|         +Ds**c+---Ss---+   +Ds**c+
|         |         |         |         |
LEFT-WALL the  cat.n chased.v- a  snake.n
```

MST Parse:

```
LEFT-WALL 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<sup>1</sup>
- Pre-processed and MST-parsed

LEFT-WALL Me very hungry ...

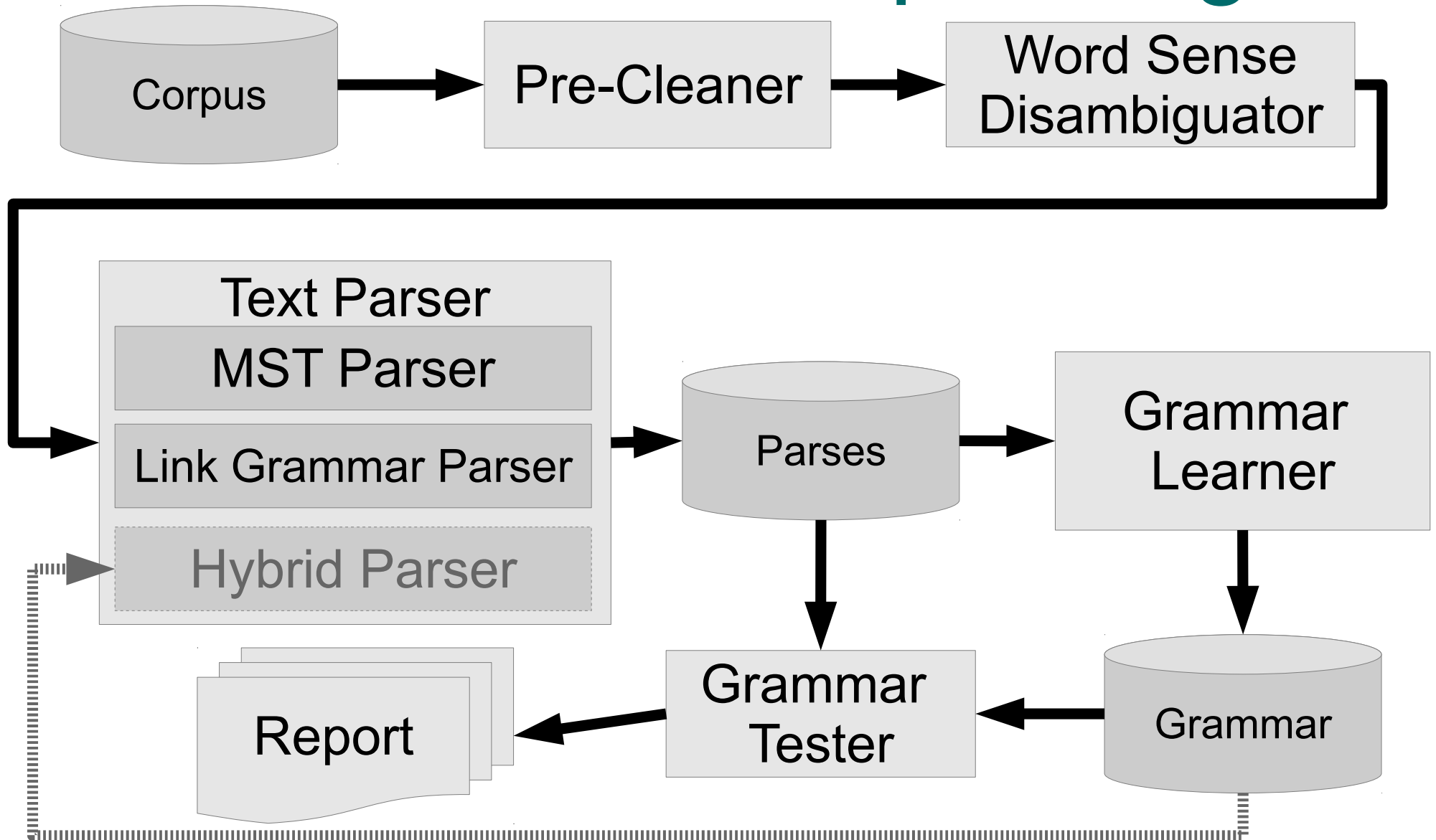
```
graph TD; Root[ ] --- L[LEFT-WALL]; Root --- N1[ ]; N1 --- Me[Me]; N1 --- N2[ ]; N2 --- very[very]; N2 --- N3[ ]; N3 --- hungry[hungry]; N3 --- dots[...];
```

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

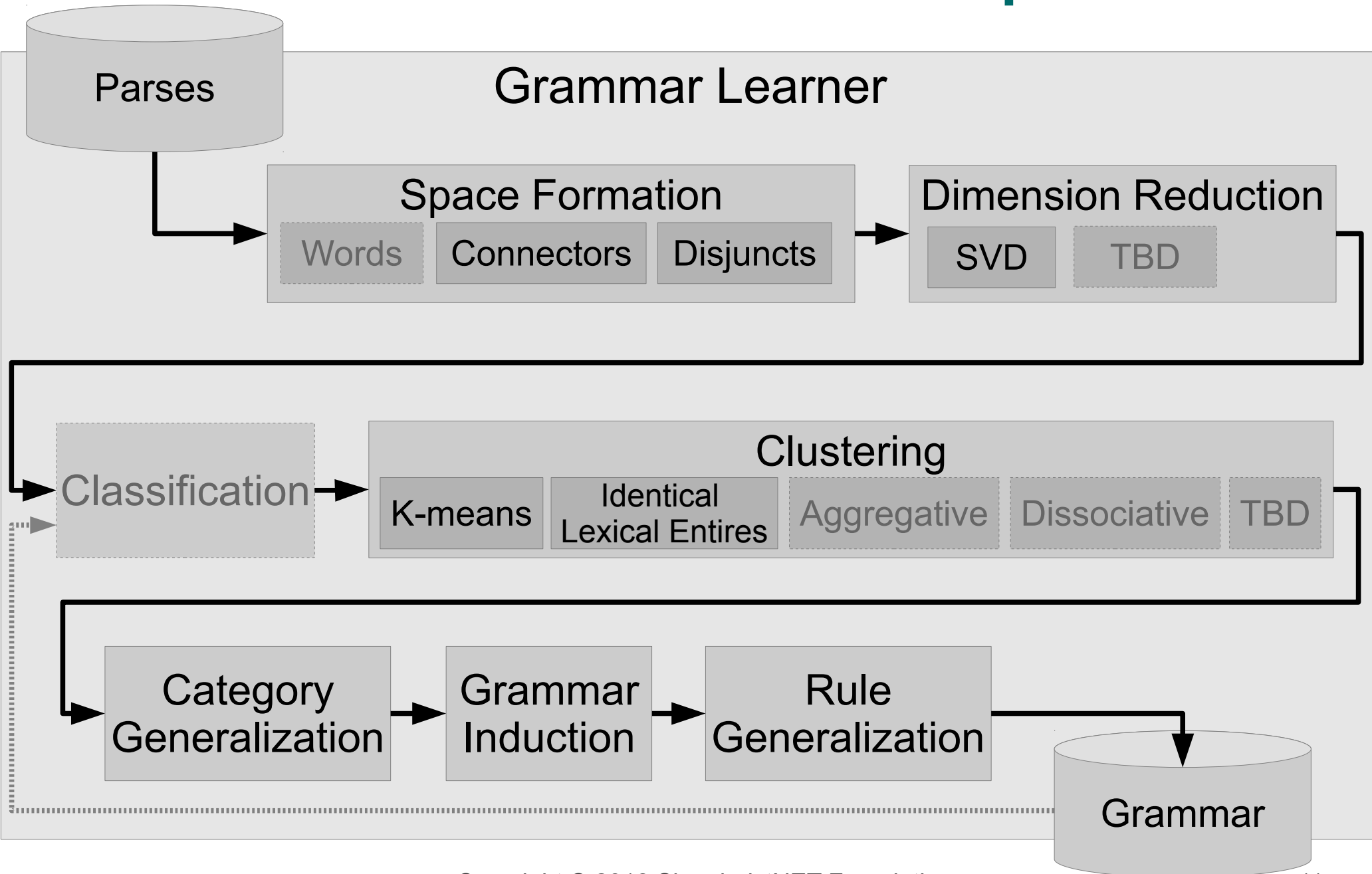
```
graph TD; Root[ ] --- L[LEFT-WALL]; Root --- N1[ ]; N1 --- Haha[Haha]; N1 --- N2[ ]; N2 --- comma[,]; N2 --- N3[ ]; N3 --- can[can]; N3 --- N4[ ]; N4 --- go[go]; N4 --- N5[ ]; N5 --- for[for]; N5 --- N6[ ]; N6 --- one[one]; N6 --- N7[ ]; N7 --- whole[whole]; N7 --- N8[ ]; N8 --- stretch[stretch]; N8 --- dots[...];
```

<sup>1</sup> [https://www.kaggle.com/rtatman/the-national-university-of-singapore-sms-corpus#smsCorpus\\_en\\_2015.03.09\\_all.json](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



# Results: Word-Sense Disambiguation

Using AdaGram<sup>1</sup> we disambiguate our POC-English corpus without supervision.

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

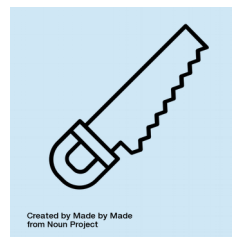


Created by iconstock from Noun Project



Created by b fariss from Noun Project

board



Created by Made by Made from Noun Project



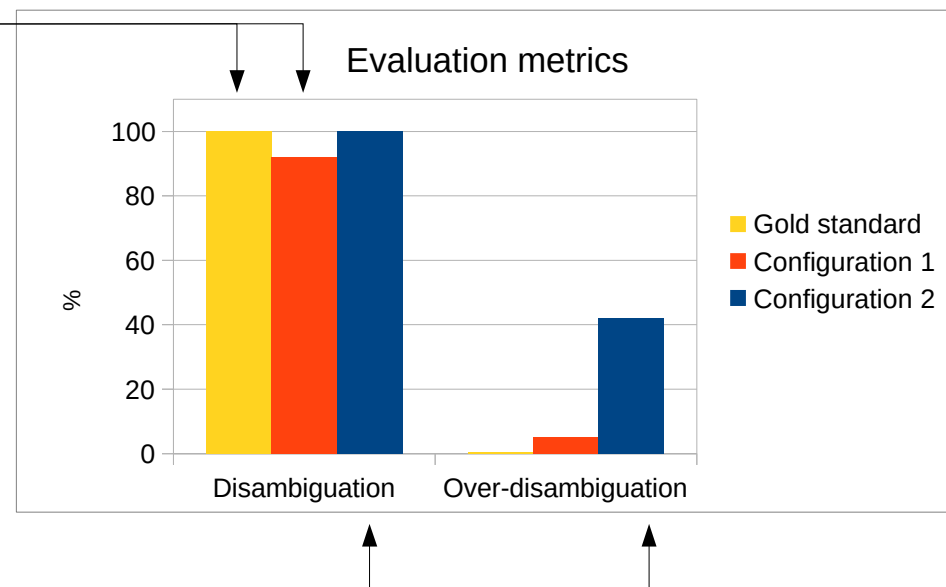
Created by Filippo Gianessi from Noun Project

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 .



<sup>1</sup> [https://github.com/glicerico/AdaGram/tree/take\\_sentences](https://github.com/glicerico/AdaGram/tree/take_sentences)

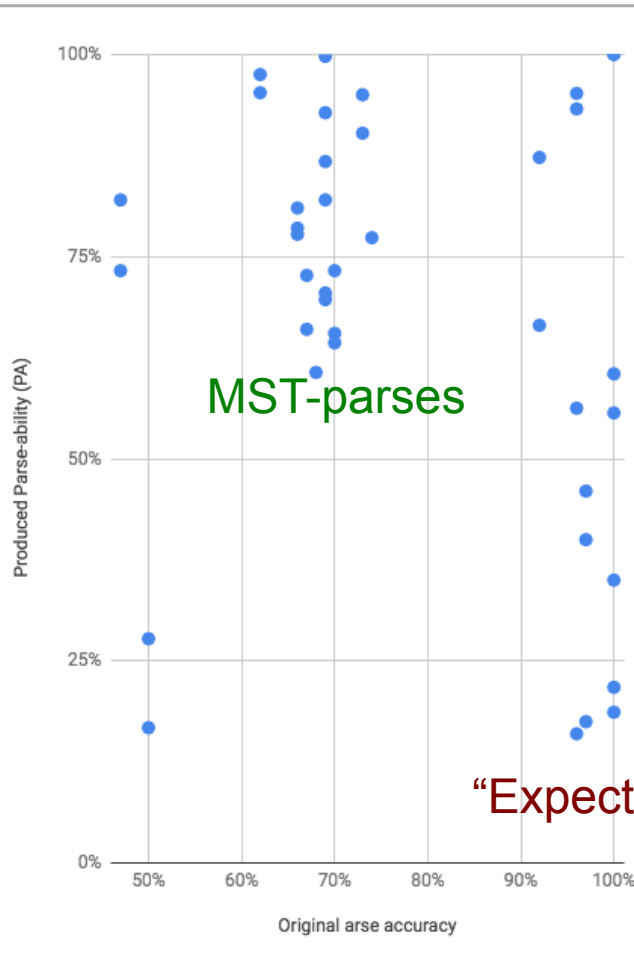
# Results: Corpora with PA & PQ

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

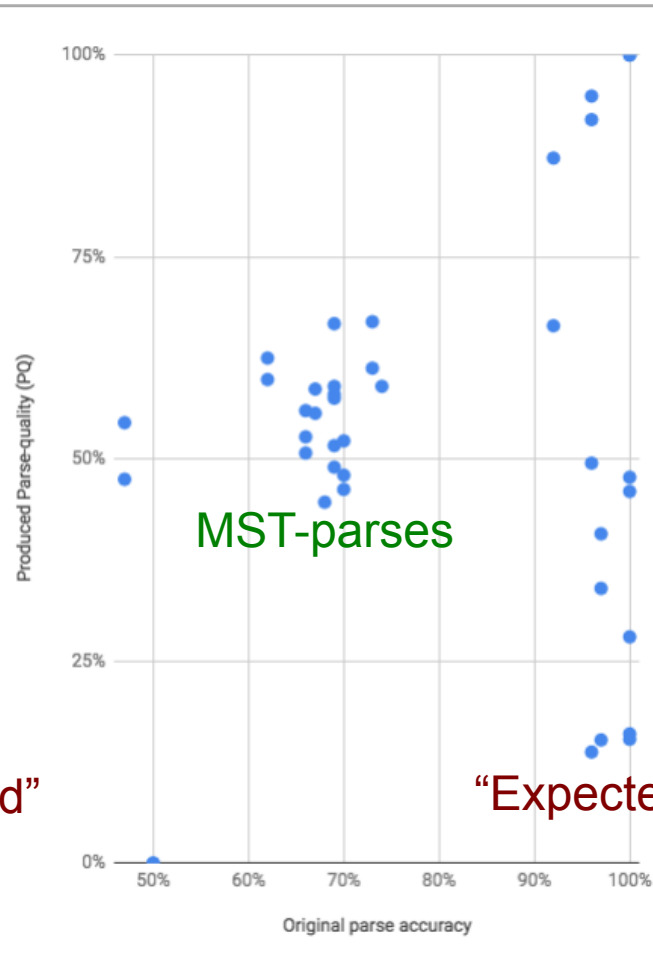
Across corpora with the best found configurations, artificially learned grammar makes it possible to **parse 75-100% of text (Parse-ability or PA), having 60-100% of it parsed properly (Parse-quality or PQ), with properly parsed fraction of parsed text above 63%**

# Results: MST Parses

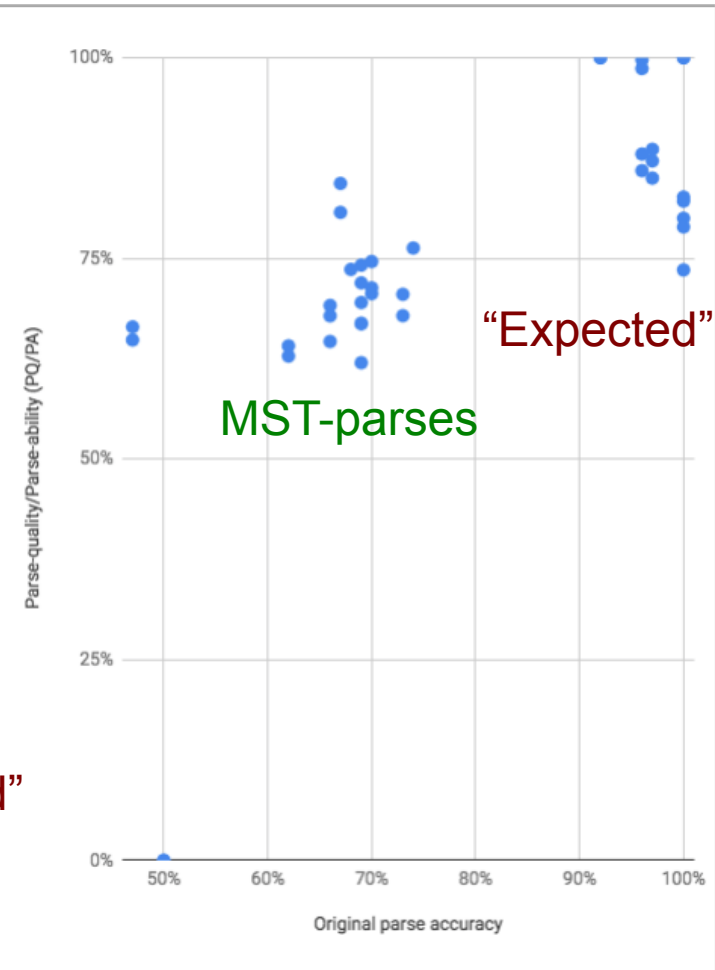
## Parse-ability (PA)



## Parse-quality (PQ)

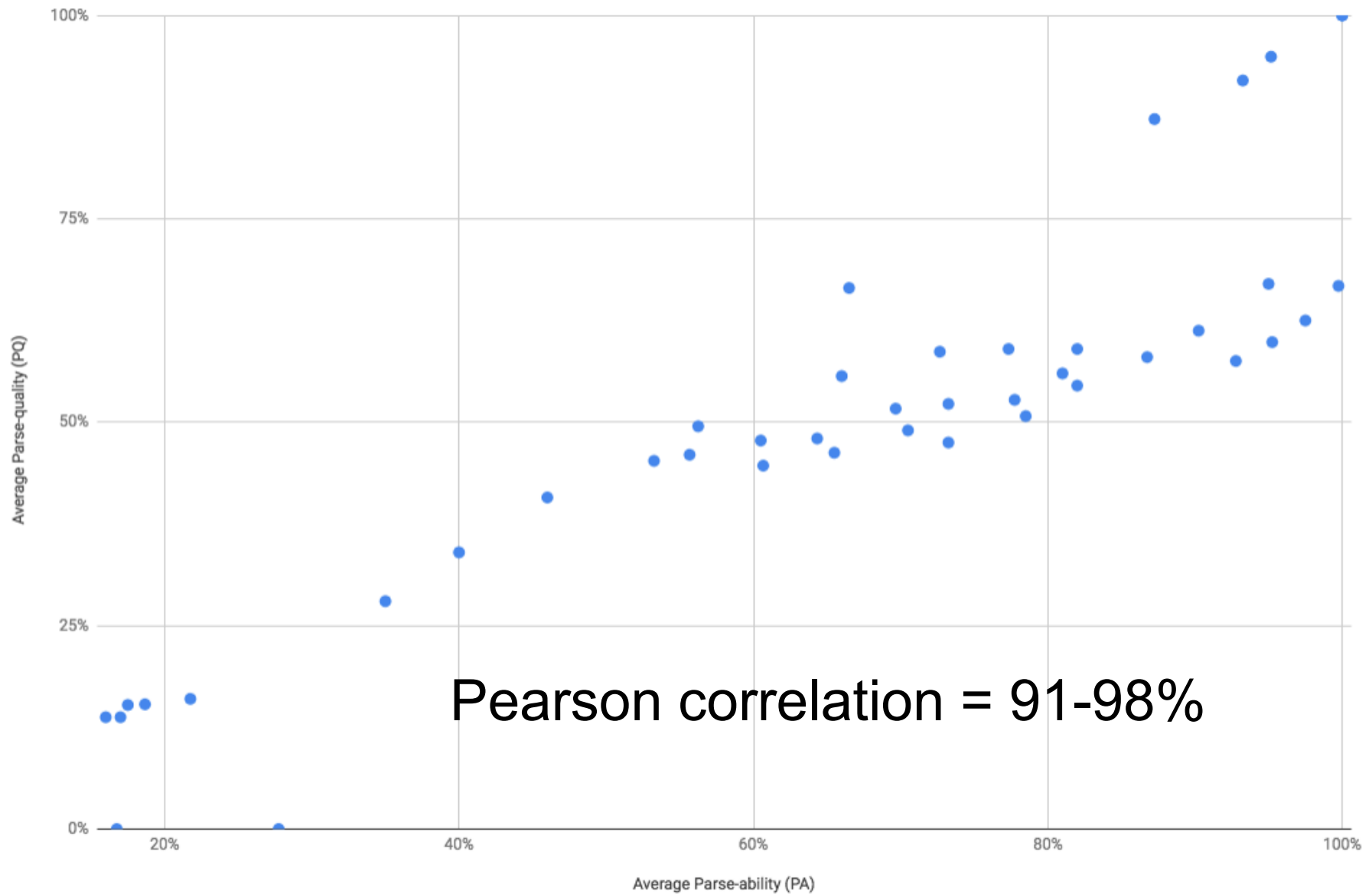


## PQ/PA



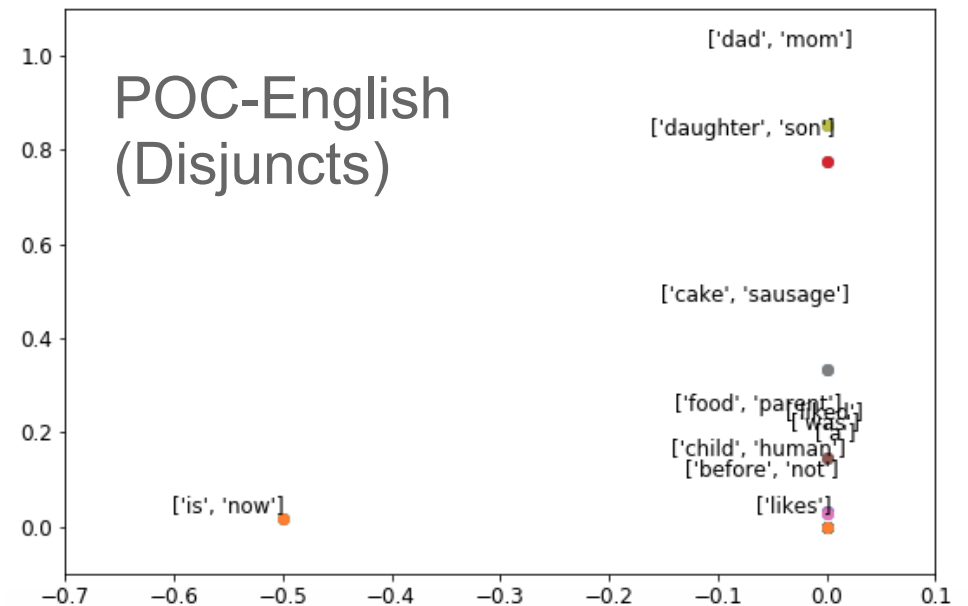
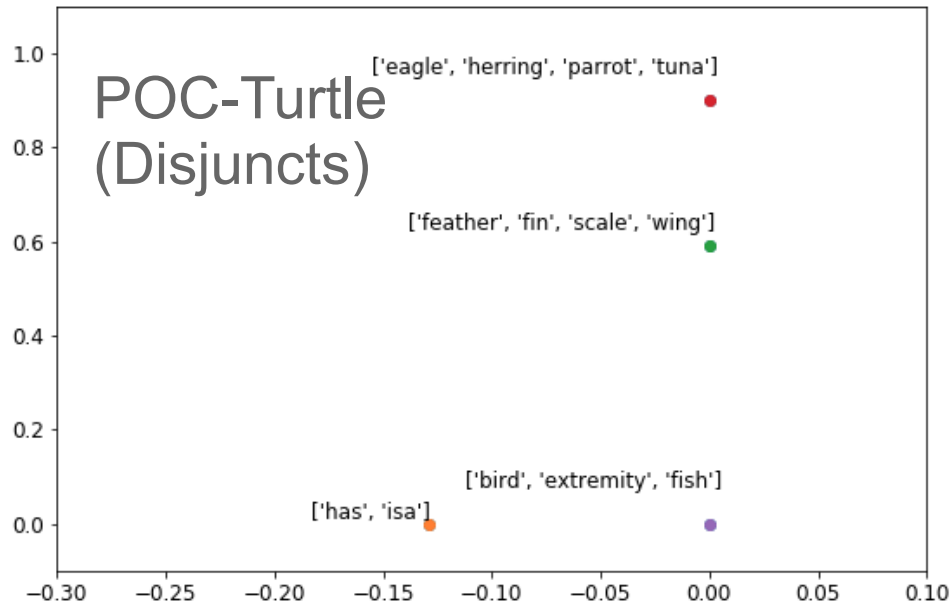
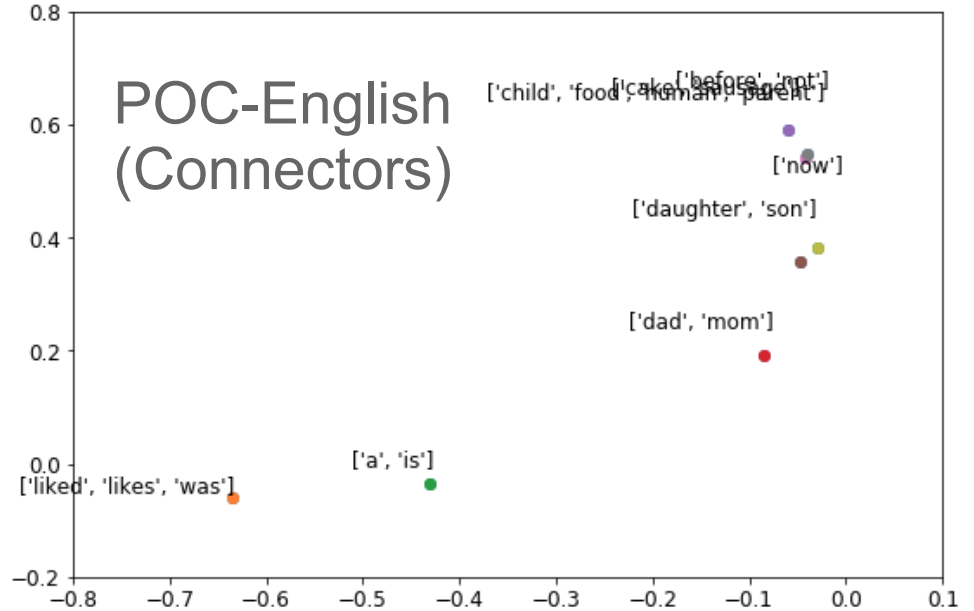
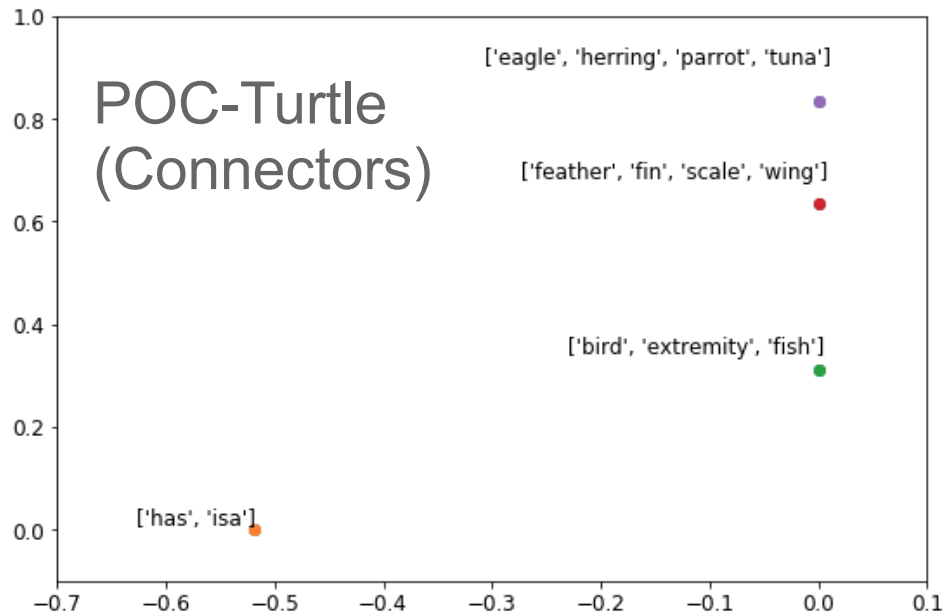
- Better parses – better results (generally, for MST parses - especially)
- Good parses – does not mean good results (for “expected” parses)

# Results: PA and PQ correlation



**Better Parse-ability (PA) implies better Parse-quality (PQ)**

# Results: Categorical spaces (POC)

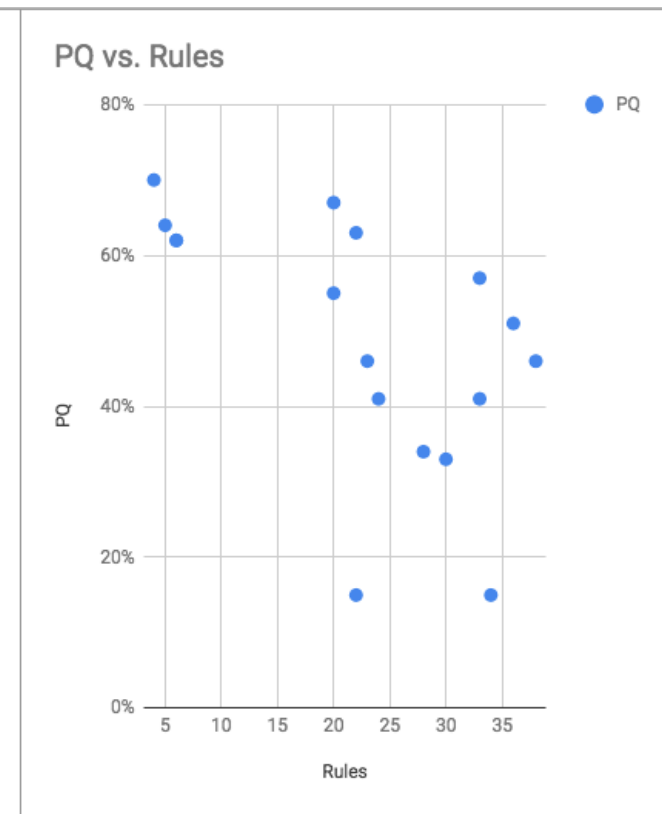
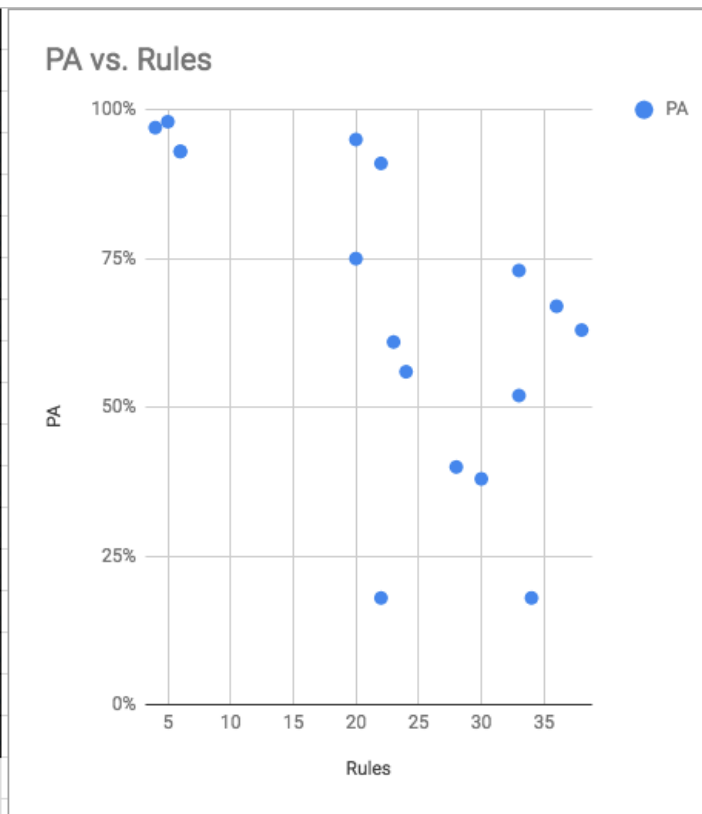






# Results: Categories and PA and PQ

Parsing	Gene	Space	Rules	PA	PQ
MST-fixed-manually	none	dDRKd	30	38%	33%
MST-fixed-manually	rules	dDRKd	28	40%	34%
LG-English	none	dDRKd	34	18%	15%
LG-English	rules	dDRKd	22	18%	15%
R=6-Weight=6:R-mst-weight=+1:R	none	dDRKd	6	93%	62%
R=6-Weight=6:R-mst-weight=+1:R	rules	dDRKd	6	93%	62%
R=6-Weight=6:R-mst-weight=+1:R	none	dDRKd	38	63%	46%
R=6-Weight=6:R-mst-weight=+1:R	rules	dDRKd	36	67%	51%
R=6-Weight=6:R-mst-weight=+1:R	rules	dDRKd	4	97%	70%
R=6-Weight=1-mst-weight=+1:R	none	dDRKd	20	75%	55%
R=6-Weight=1-mst-weight=+1:R	none	dDRKd	24	56%	41%
R=6-Weight=1-mst-weight=+1:R	rules	dDRKd	23	61%	46%
R=6-Weight=1-mst-weight=+1:R-ac	rules	dDRKd	33	52%	41%
LG-ANY-all-parses	none	dDRKd	5	98%	64%
LG-ANY-all-parses	none	dDRKd	22	91%	63%
LG-ANY-all-parses	rules	dDRKd	20	95%	67%
LG-ANY-all-parses-adagram	rules	dDRKd	33	73%	57%
<b>Pearson correlation with N of Rules</b>				<b>-0.6</b>	<b>-0.6</b>

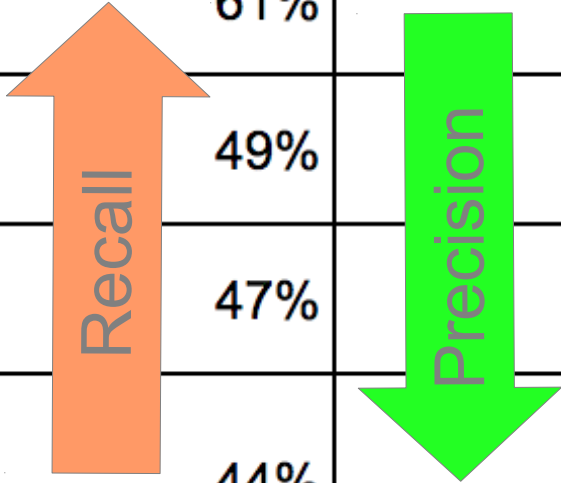


- Fewer categories – better Parse-ability (PA) and Parse-quality (PQ)
- Number of categories vary from upper limit of natural number of identical lexical entries (tens to thousands) to 4-6 basic “parts of speech” - randomly due to uncontrolled nature of K-means clustering which has to be replaced with controlled aggregative/dissociative clustering

# Results: Grammar Learning Algorithms

Comparing across all corpora

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



- 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://languarn.singularitynet.io/>

## Stay in touch:

Ben Goertzel

[ben@singularitynet.io](mailto:ben@singularitynet.io)

Cassio Pennachin

[cassio@singularitynet.io](mailto:cassio@singularitynet.io)

Anton Kolonin

[anton@singularitynet.io](mailto:anton@singularitynet.io)



OpenCog

<https://opencog.org/>



SingularityNET

<https://singularitynet.io>



<http://www.hansonrobotics.com/>