Tree-Adjoining Grammar Parsing and Applications

Jungo Kasai

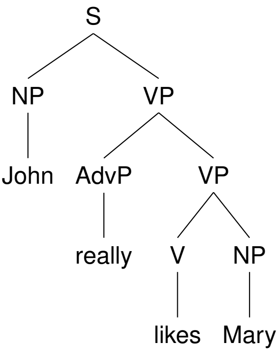
Yale University

April 27, 2018

Outline

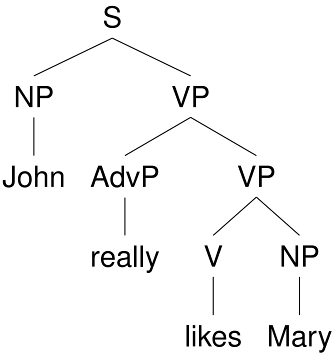
Outline

Syntactic Parsing



- Why do we need parsing?
- Does John love Mary? Does Mary love John?
- Understanding of a sentence depends on the structure

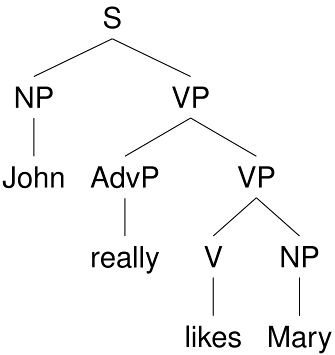
Context Free Grammars



$$S \rightarrow NP\ VP$$
$$VP \rightarrow AdvP\ VP$$
$$AdvP \rightarrow really$$
$$VP \rightarrow V\ NP$$
$$NP \rightarrow Mary$$
$$NP \rightarrow they$$
$$NP \rightarrow John$$
$$V \rightarrow like$$
$$V \rightarrow likes$$

- These production rules generate sentences

Context Free Grammars



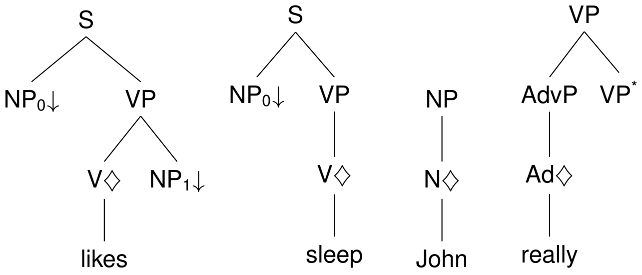
$$S \rightarrow NP\ VP$$
$$VP \rightarrow AdvP\ VP$$
$$AdvP \rightarrow really$$
$$VP \rightarrow V\ NP$$
$$NP \rightarrow Mary$$
$$NP \rightarrow they$$
$$NP \rightarrow John$$
$$V \rightarrow like$$
$$V \rightarrow likes$$

- **Fundamental problem**: constraints are distributed over separate rules
  How do we choose $V \rightarrow$ *like* or $V \rightarrow$ *likes*?

Tree-Adjoining Grammar

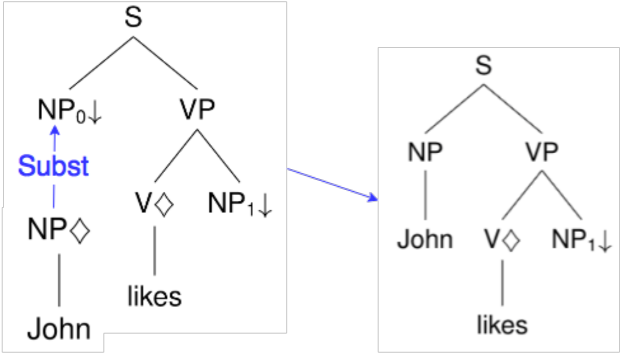
Tree-Adjoining Grammar (TAG) localizes grammatical constraints

- Finite set of lexicalized elementary trees
- Finite set of operations (Substitution and Adjunction) are used to combine elementary trees



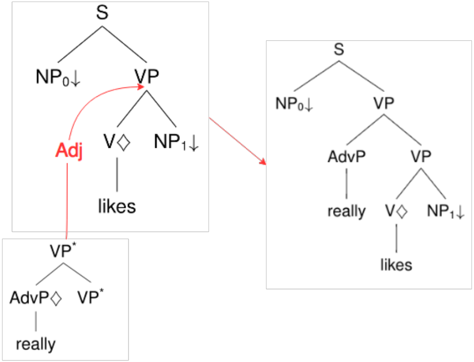
Tree-Adjoining Grammar

Substitution

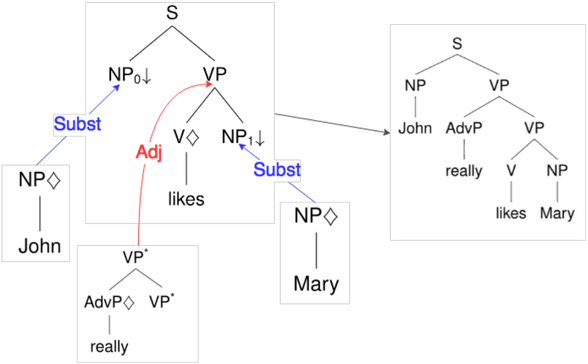


Tree-Adjoining Grammar

Adjunction



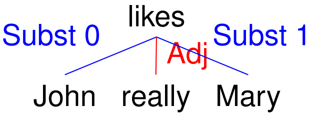
Tree-Adjoining Grammar

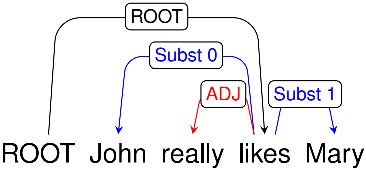


Adjunction allows for unbounded recursion while still enforcing agreement.
John smartly occasionally really only likes Mary...

Derivation Tree

Derivation tree records the operations.
Forms a dependency tree (each token has exactly one parent)

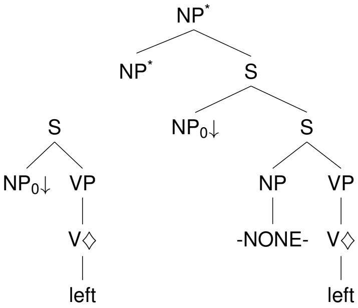




Two Steps in TAG Parsing

Now the reverse process.

- *Supertagging*
  Assign elementary trees (supertags) to each token. Similar to POS tagging.
- *Parsing*
  Predict operations on the elementary trees.



Outline

Supertagging is a bottleneck

Supertagger	Parser	Stag Acc	UAS	LAS
Gold	Chart (MICA)	100.00	97.60	97.30
Maxent (MICA)	Chart (MICA)	88.52	87.60	85.80

- Supertagging is *almost parsing*
- There are about 5,000 supertags in the grammar
- About half of them occur only once in the training data (PTB WSJ Sections 1-22).

BiLSTM Supertagging

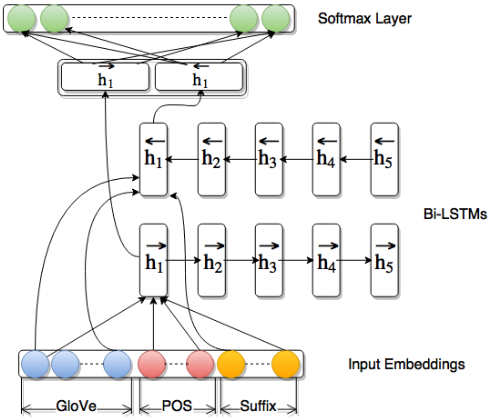


Figure: BiLSTM Supertagger Architecture.

Supertagging is still a bottleneck

Supertagger	Parser	Stag Acc	UAS	LAS
Maxent (MICA)	Chart (MICA)	88.52		
BiLSTM	Chart (MICA)	**89.32**		

Supertagging is still a bottleneck

Supertagger	Parser	Stag Acc	UAS	LAS
Gold	Chart (MICA)	100.00	97.60	97.30
Maxent (MICA)	Chart (MICA)	88.52	87.60	85.80
BiLSTM	Chart (MICA)	**89.32**	**90.05**	**88.32**

- We can compensate for supertagging errors by exploiting structural similarities across elementary trees.
- Similarities across supertags are not utilized by the chart parser.
- We use two alternative families of parsing algorithms

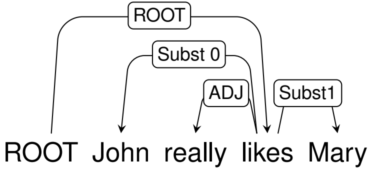
Outline

Parsing Models

- Prior Work: Unlexicalized Chart-Parser (MICA) [Bangalore et al., 2009]
- Unlexicalized Transition-based Parser [Kasai et al., 2017, Friedman et al., 2017]
- Graph-based Parser [Kasai et al., 2018]

Transition-based Parsing

Arc-Eager System (MALT) [Nivre et al., 2006]



Transition-based Parsing

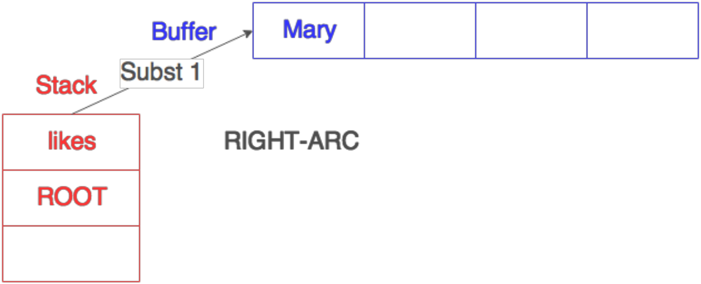
Transition-based Parsing

Transition-based Parsing

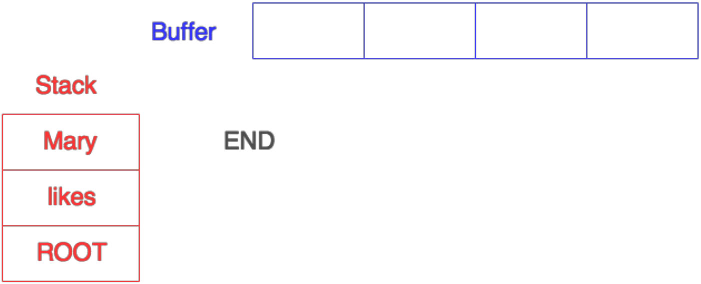
Transition-based Parsing

Transition-based Parsing

Transition-based Parsing



Transition-based Parsing



Transition-based TAG Parsing

How do we learn?

- Represent the configuration by the top k elements from stack and buffer: $\{s_i, b_i\}_{i=1}^{k}$ [Chen and Manning, 2014].
- Represent s_i (b_i) by the TAG elementary tree and the derived substitution operations performed into s_i.
- Encode the TAG elementary trees and the substitution operations with dense vectors.

NN Transition-based Parsing Model

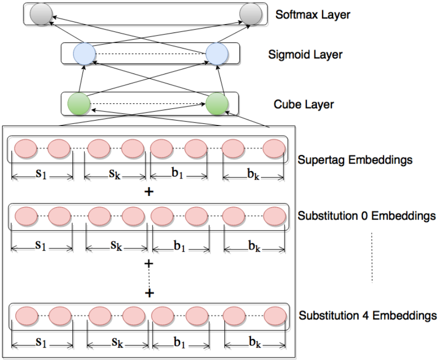
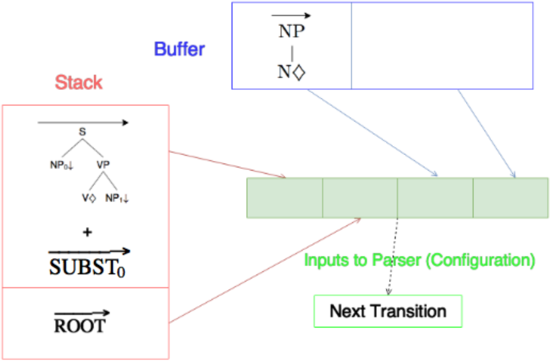


Figure: Transition-based Parser Neural Network Architecture.

Example

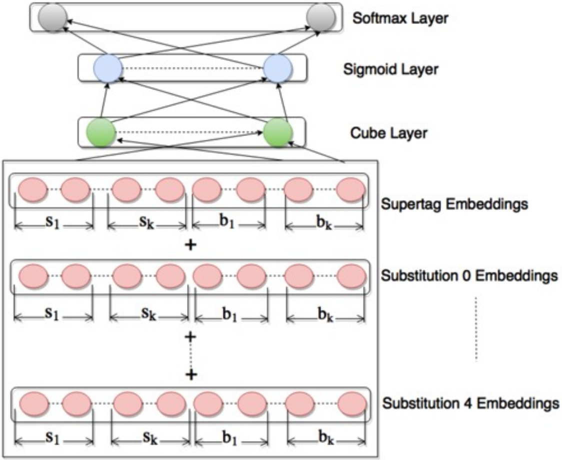
John really likes Mary

Stack	Buffer	Relations	Action
ROOT likes	Mary	$\{(ROOT, likes, ROOT), (likes, John, 0) \cdots\}$	RIGHT:1



Outline

Embeddings for Elementary Trees



PCA Plots of Vector Representations

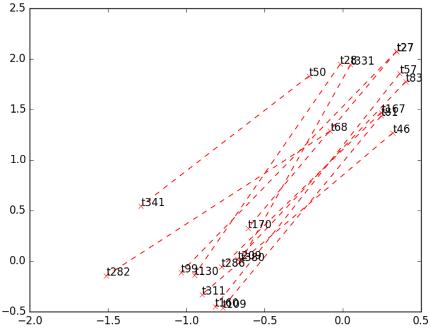


Figure: Declarative/subject relative alignment (Atomic embeddings)
ex: *the man* ***sneezed*** vs. *the man who* ***sneezed***

PCA Plots of Vector Representations

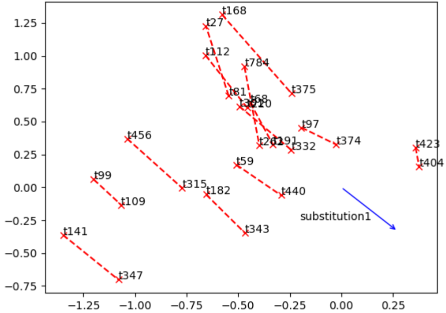


Figure: Transitive/intransitive alignment (Atomic embeddings).
ex: *the man who **devoured** the pizza* vs. *the man who **sneezed***

Analogy Test Results

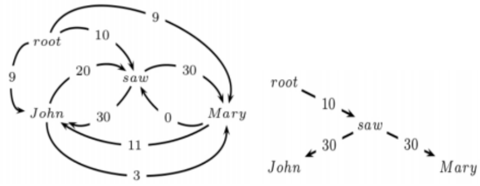
n	# equations	% correct	Avg. position
300	246	50.40	7.98
4724	57220	4.62	289.48

Table: Analogy task results. (n is a number used to restrict which supertags are considered: For a given n, only equations for which all supertags are among the n most common supertags are considered.)

- **% correct**: Percent of equations for which the left hand side's closest cosine neighbor was the right hand side.
- **Avg. position**: The position of the correct right hand side in the list of supertag embeddings ranked by cosine distance from the left hand side.

Outline

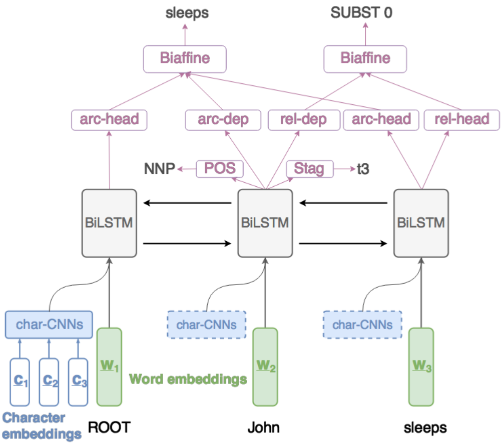
Graph-based Parsing



[McDonald et al., 2005]

- Score n^2 directed edges (n potential parents for each of the n tokens in a sentence) using features
- Find the maximum spanning tree (greedy + cycle fix)

Graph-based Parsing TAG Parsing



Comparison btw transition-based and graph-based

Rich feature representations with parse history v.s. global training/inference

- Transition-based parsers have parse history that naturally relates supertags that differ only by a certain operation (e.g. transitive/intransive)
- Transition-based parsers suffer from global error propagation
- Graph-based parsers assign scores independently
- Graph-based parser with BiLSTM feature representations (still no history)

New Results

Parser	UAS	LAS
MICA Chart	86.66	84.90
Transition-based Parsing	90.97	89.68
Joint Graph Parsing (POS+Stag)	**93.26**	**91.89**

Table: Parsing results on the test set.

Outline

Syntactically-oriented Textual Entailment

[Xu et al., 2017]

E.g. *The guy who left the room saw a squirrel*

$\Longrightarrow$ *The guy left the room*

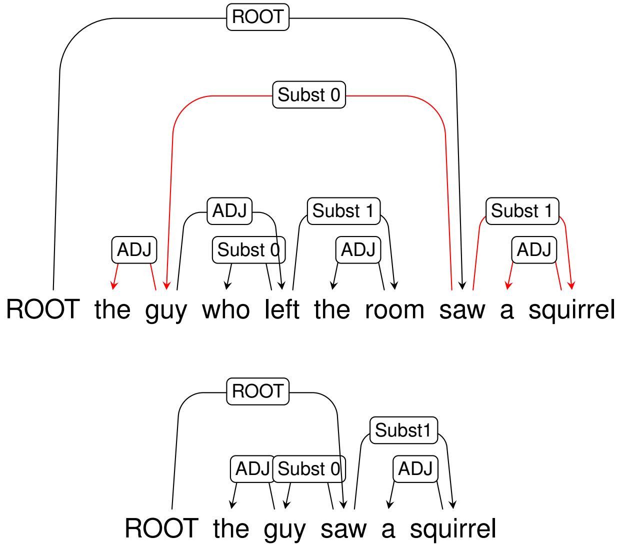
$\Longrightarrow$ *The guy saw a squirrel*

$\not\Longrightarrow$ *The room saw a squirrel*

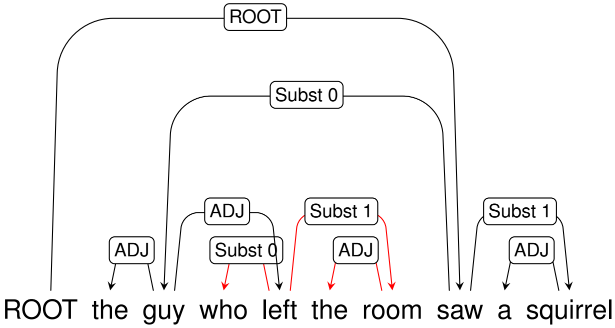
$\not\Longrightarrow$ *The guy saw an animal* (not pure syntactic)

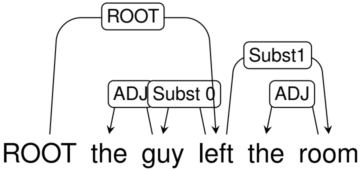
- Parse the original sentence and hypothesis
- Transform the parses using properties of supertags
- If the original sentence parse subsumes the hypothesis one, YES.

Syntactically-oriented Textual Entailment



Syntactically-oriented Textual Entailment



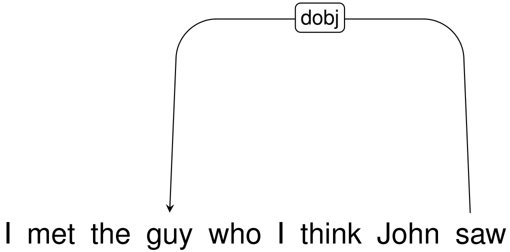


Syntactically-oriented Textual Entailment

System	%A	%P	%R	F1
[Rimell and Clark, 2010]	72.4	79.6	62.8	70.2
[Ng et al., 2010]	70.4	68.3	80.1	73.7
[Lien, 2014]	70.7	88.6	50.0	63.9
Transition-based TAG Parsing	72.4	85.4	56.4	68.0
Graph-based Method	**78.1**	86.3	68.6	**76.4**

Table: PETE test results. Precision (P), recall (R), and F1 are calculated for "entails."

Unbounded Dependency Recovery



System	obRC	obRd	sbRC	free	obQ	rnr	sbEm	Total
C&C	59.3	62.6	80.0	72.6	**72.6**	**49.4**	22.4	53.6
Enju	47.3	65.9	82.1	76.2	32.5	47.1	32.9	54.4
Stanford	22.0	1.1	74.7	64.3	41.2	45.4	10.6	38.1
MST	34.1	47.3	78.9	65.5	41.2	45.4	37.6	49.7
MALT	40.7	50.5	**84.2**	70.2	31.2	39.7	23.5	48.0
Joint	**72.5**	**78.0**	81.1	**85.7**	56.3	47.1	**49.4**	**64.9**

Acknowledgement

Thank you!

- Robert Frank, Dan Friedman, Tom McCoy, William Merrill, Alexis Nasr, Dragomir Radev, Owen Rambow, and Pauli Xu

Bangalore, S., Boullier, P., Nasr, A., Rambow, O., and Sagot, B. (2009).
MICA: A Probabilistic Dependency Parser Based on Tree Insertion Grammars.
In *NAACL HLT 2009 (Short Papers)*.

Chen, D. and Manning, C. (2014).
A fast and accurate dependency parser using neural networks.
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Friedman, D., Kasai, J., McCoy, R. T., Frank, R., Davis, F., and Rambow, O. (2017).
Linguistically rich vector representations of supertags for tag parsing.

In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131, Umeå, Sweden. Association for Computational Linguistics.

Kasai, J., Frank, R., McCoy, R. T., Rambow, O., and Nasr, A. (2017).
TAG Parsing with Neural Networks and Vector Representations of Supertags.
In *Proceedings of EMNLP*.

Kasai, J., Frank, R., Xu, P., Merrill, W., and Rambow, O. (2018).
End-to-end Graph-based TAG Parsing with Neural Networks.
In *Proceedings of NAACL*.

Lien, E. (2014).
Using minimal recursion semantics for entailment recognition.

In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, page 7684, Gothenburg, Sweden.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005).
Non-projective dependency parsing using spanning tree algorithms.
In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ng, D., Constable, J. W., Honnibal, M., and Curran, J. R. (2010).
SCHWA: PETE using CCG dependencies with the C&C parser.

In *Proceedings of the 5th International Workshop on Semantic Evaluation*, page 313316.

Nivre, J., Hall, J., and Nilsson, J. (2006).
Maltparser: A data-driven parser-generator for dependency parsing.
In *LREC*.

Rimell, L. and Clark, S. (2010).
Cambridge: Parser evaluation using textual entailment by grammatical relation comparison.
In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 268–271.

Xu, P., Frank, R., Kasai, J., and Rambow, O. (2017).
Tag parser evaluation using textual entailments.
In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 132–141, Umeå, Sweden. Association for Computational Linguistics.