

*Syntactic and semantic parsing
for natural language understanding*

2. Syntactic and semantic parsing with CCG

Hiroshi Noji

Topics

- ▶ In the last week, we focused on the grammatical property of CCG
 - Logical forms can be obtained through syntactic derivations
 - The number of rules is small, and much information is encoded in the lexicon
- ▶ Today's topic:
 - how to efficiently obtain CCG derivation on the input sentence
= **syntactic parsing**
 - Application to semantic parsing in textual entailment and QA

Outline

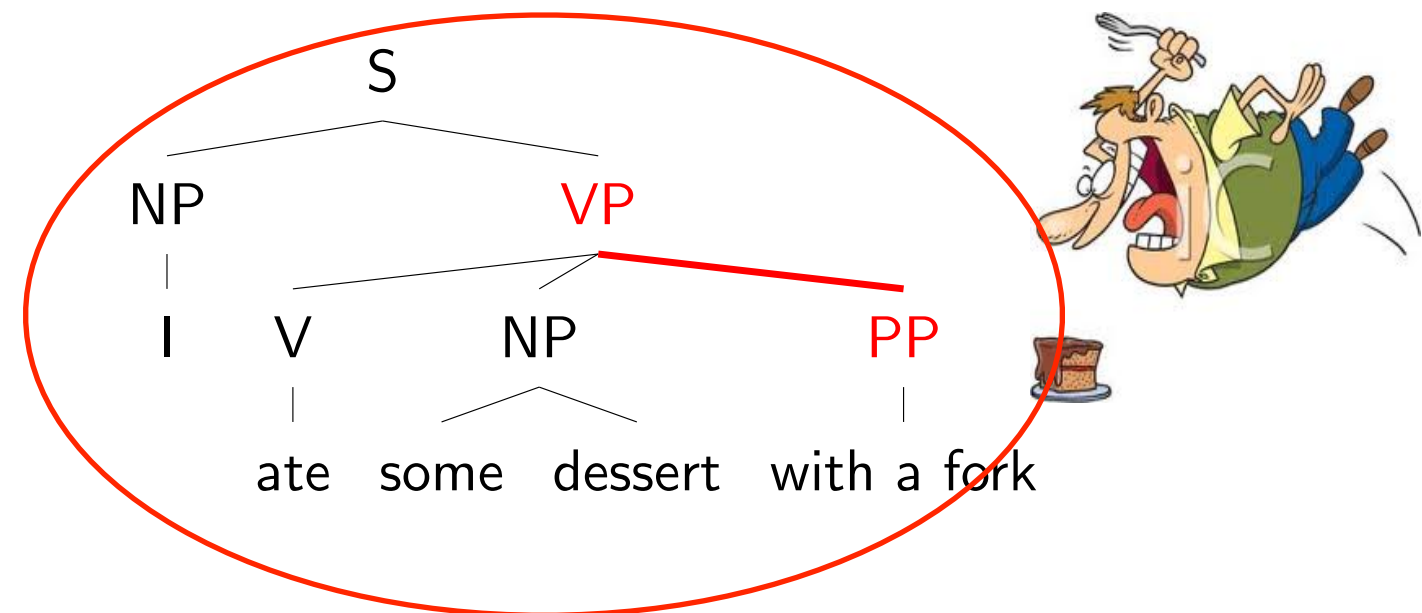
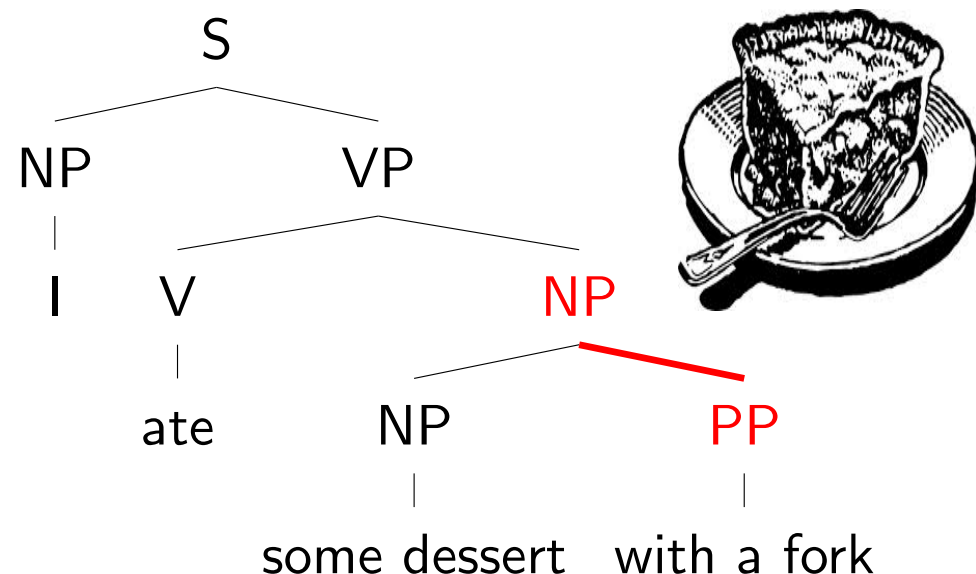
- ▶ Efficient parsing techniques for CCG
- ▶ Bottom-up semantic parsing with CCG
 - application to textual entailment
 - issues in question answering
- ▶ Top-down semantic parsing with CCG for question answering

QA is moved to the next lecture (Thursday)

Goal of parsing: disambiguation

- ▶ What is the correct tree?
 - correct tree is the one **corresponding to human interpretation**
- ▶ Parsing aims at finding a tree that corresponds to human interpretation
- ▶ Example: PP-attachment ambiguity

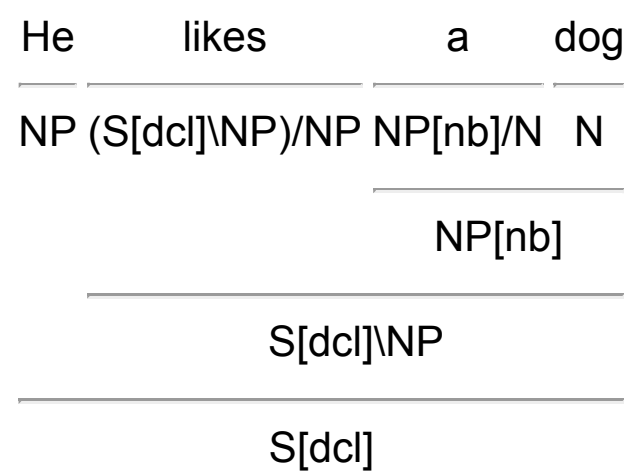
I ate some dessert with a fork



Statistical parsing

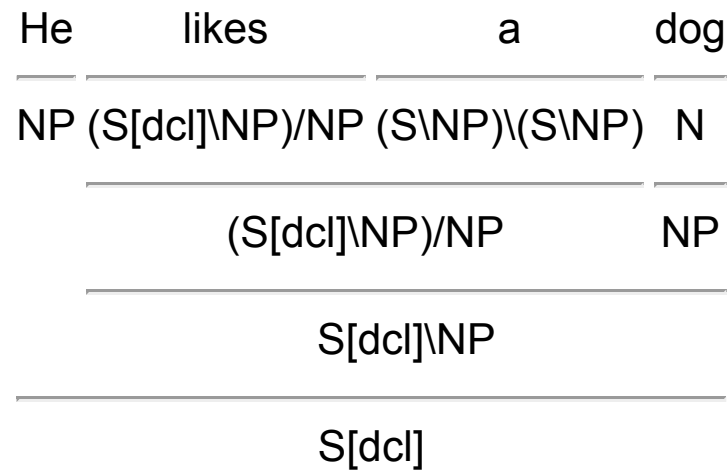
- ▶ Today most practical parsers are statistical parsers with supervised machine learning
 - Typically we use 10,000~30,000 trees to train a parser
- ▶ After training a parser, the essential problem is **search**
 - The parser can assign a score to each tree (structure)
 - How can we find the best tree for the model?
 - The number of trees is exponential to the length of the sentence
- ▶ Our recent work (Yoshikawa et al., 2017) found that CCG parsing **can be more efficient than phrase-structure parsing** by exploiting the property of the grammar effectively

There are so many parses



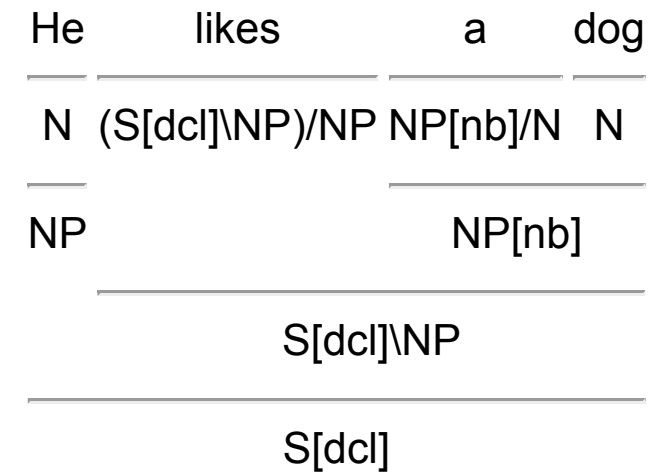
1000

600



200

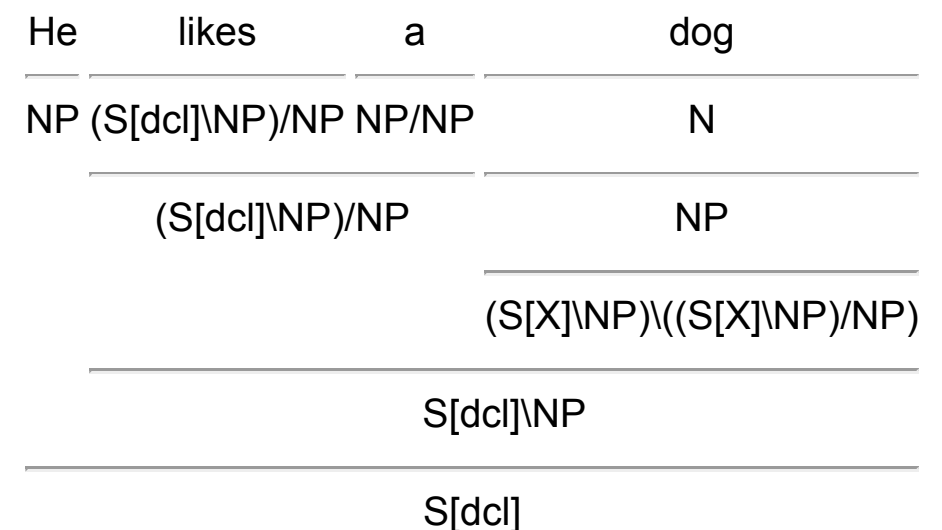
800



180

400

- ▶ Even for a very short sentence, the number of possible trees is large



100

200

- ▶ Problems:

- **Model:** how to obtain the model that gives the highest score to the correct tree?
- **Search:** how to find the best tree for the model?

Model and search

- ▶ Model and search are not independent
- ▶ Generally, more complex model gets more strong (higher expressive power), but the search becomes harder
- ▶ So one critical problem in parsing is to develop a model that best balances the tradeoff between the expressivity of the model and the difficulty of search

Simple model: PCFG

Rule:	Probability:
-------	--------------

$S \rightarrow NP VP$	1.0
-----------------------	-----

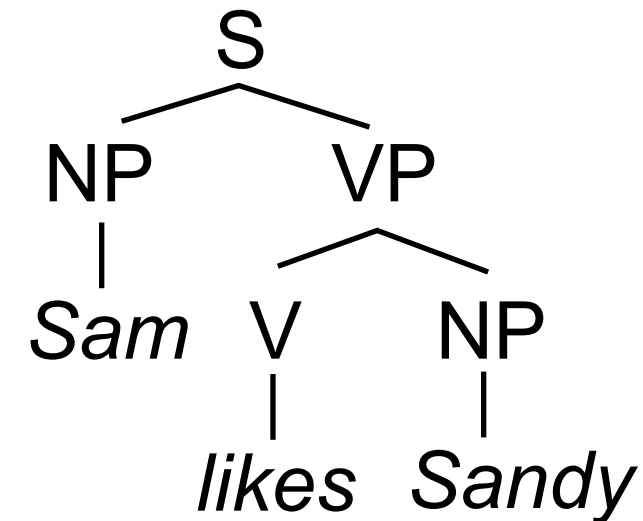
$NP \rightarrow Sam$	0.7
----------------------	-----

$NP \rightarrow Sandy$	0.3
------------------------	-----

$VP \rightarrow V NP$	1.0
-----------------------	-----

$V \rightarrow likes$	0.8
-----------------------	-----

$V \rightarrow hates$	0.2
-----------------------	-----



$$P(\text{Tree}) = 1.0 \times 0.7 \times 1.0 \times 0.8 \times 0.3$$

- ▶ Each rule has a different probability
- ▶ The probability of a tree is the product of all rules on the tree
- ▶ **Search:** we can find the highest score tree $\operatorname{argmax}_t P(t)$ with dynamic programming called CKY in $O(n^3|G|)$
 - $|G|$ is the number of rules in the grammar

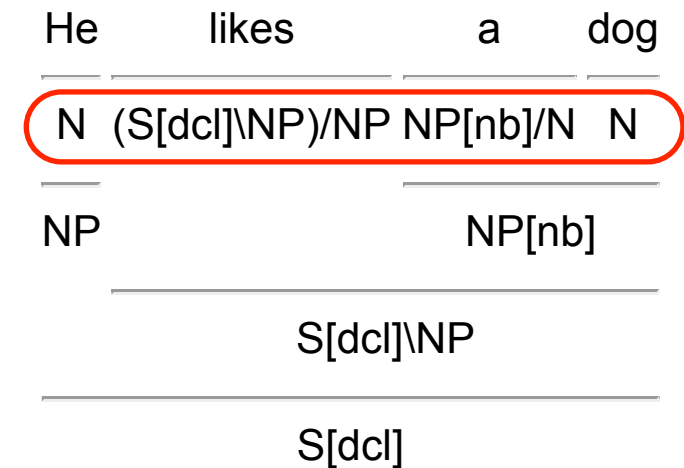
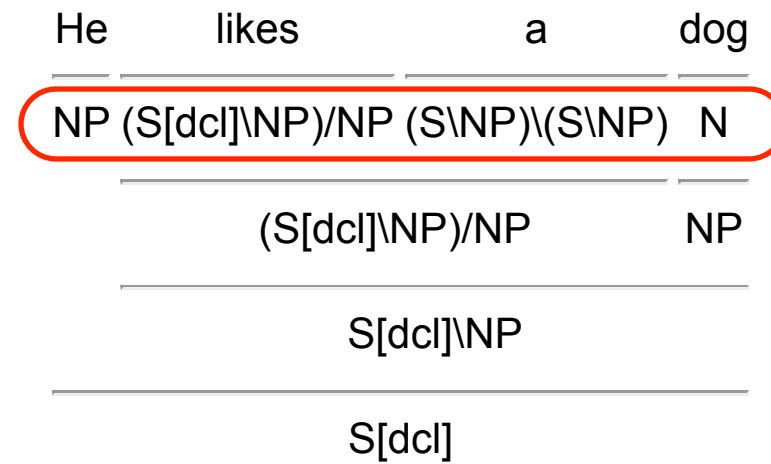
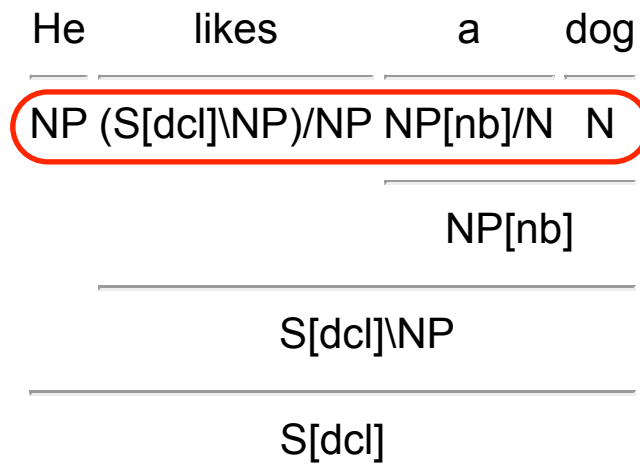
Learning is easy

- ▶ Learning a naive PCFG from the training data (= the collection of trees, called a treebank) is very simple
- ▶ Just maximum likelihood
 - Count the number of occurrences of each rule in the treebank
 - Normalize each score to get probability

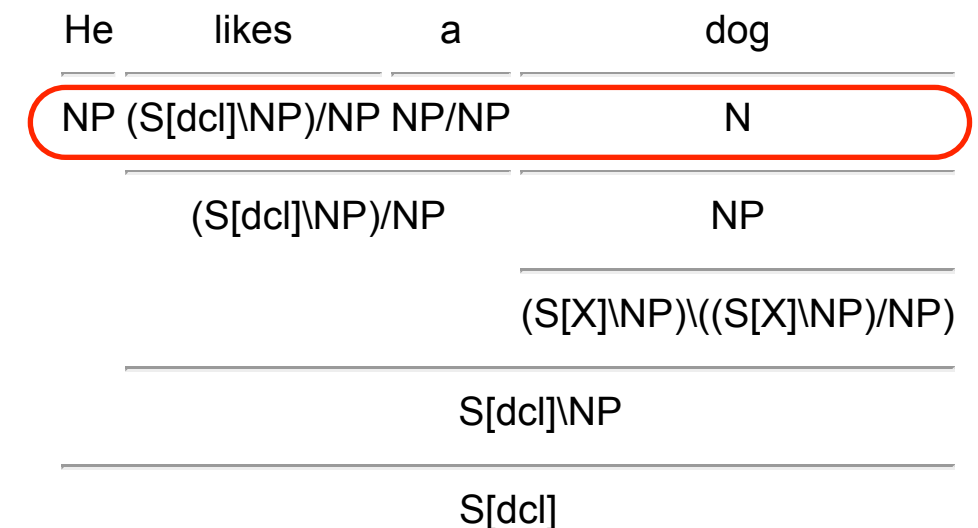
PCFG parsing for CCG?

- ▶ It has been tried at the very beginning of CCG parsing
 - Hockenmaier (2001). *Statistical Parsing for CCG with Simple Generative Models*. ACL student research workshop.
- ▶ Problem:
 - The search space (= size of grammar) is quite huge
 - $r = (S \backslash NP) / NP \rightarrow (S \backslash NP) / (S \backslash NP) \quad S \backslash NP / NP \dots$
 - A number of occurrences of each rule is small = **sparsity issue**
- ▶ The popular parser (Clark and Curran, 2008) employs a log-linear model for scoring each rule
 - $\text{Score}(r) = \text{Score}(\text{top}=(S \backslash NP) / NP) + \text{Score}(\text{Right}= S \backslash NP / NP) + \dots$
 - **breaking independence assumption of each rule**

Can we exploit the property of CCG?



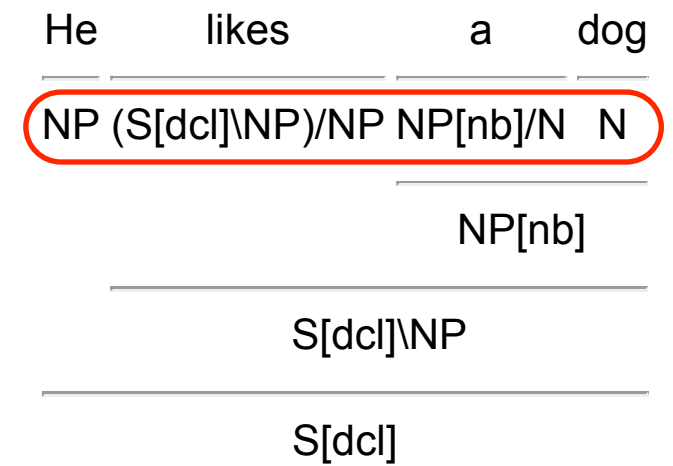
- ▶ All structures differs in **category assignments for words**



- In other words, if we find the correct category assignments, we can find the correct tree
- This is because each CCG category is syntactically highly informative \Rightarrow categories drastically reduce search space

Supertagging

- ▶ CCG categories are highly informative
 - When correct categories are given to all words, random choice from all grammatical trees achieves >95% accuracy
 - So it is essential to assign correct category to every word
- ▶ In previous CCG parsing (e.g., Clark and Curran, 2008), **the process of assigning categories** is called **supertagging**
 - Before scoring a derivation (tree), supertagging restricts the possible categories to each word
 - This reduces the search space, and speeds up the search



Supertagging

He	likes	a	dog
N	N	N	N
NP	NP	NP	NP
S\NP/NP	S\NP/NP	S\NP/NP	S\NP/NP
NP/N	NP/N	NP/N	NP/N
NP/NP	NP/NP	NP/NP	NP/NP
(S\NP)/(S\NP)	(S\NP)/(S\NP)	(S\NP)/(S\NP)	(S\NP)/(S\NP)

- ▶ For longer sentences, this approach is quite effective
- ▶ Supertagger model can be trained by a standard machine learning technique (e.g., linear classifier, neural networks)

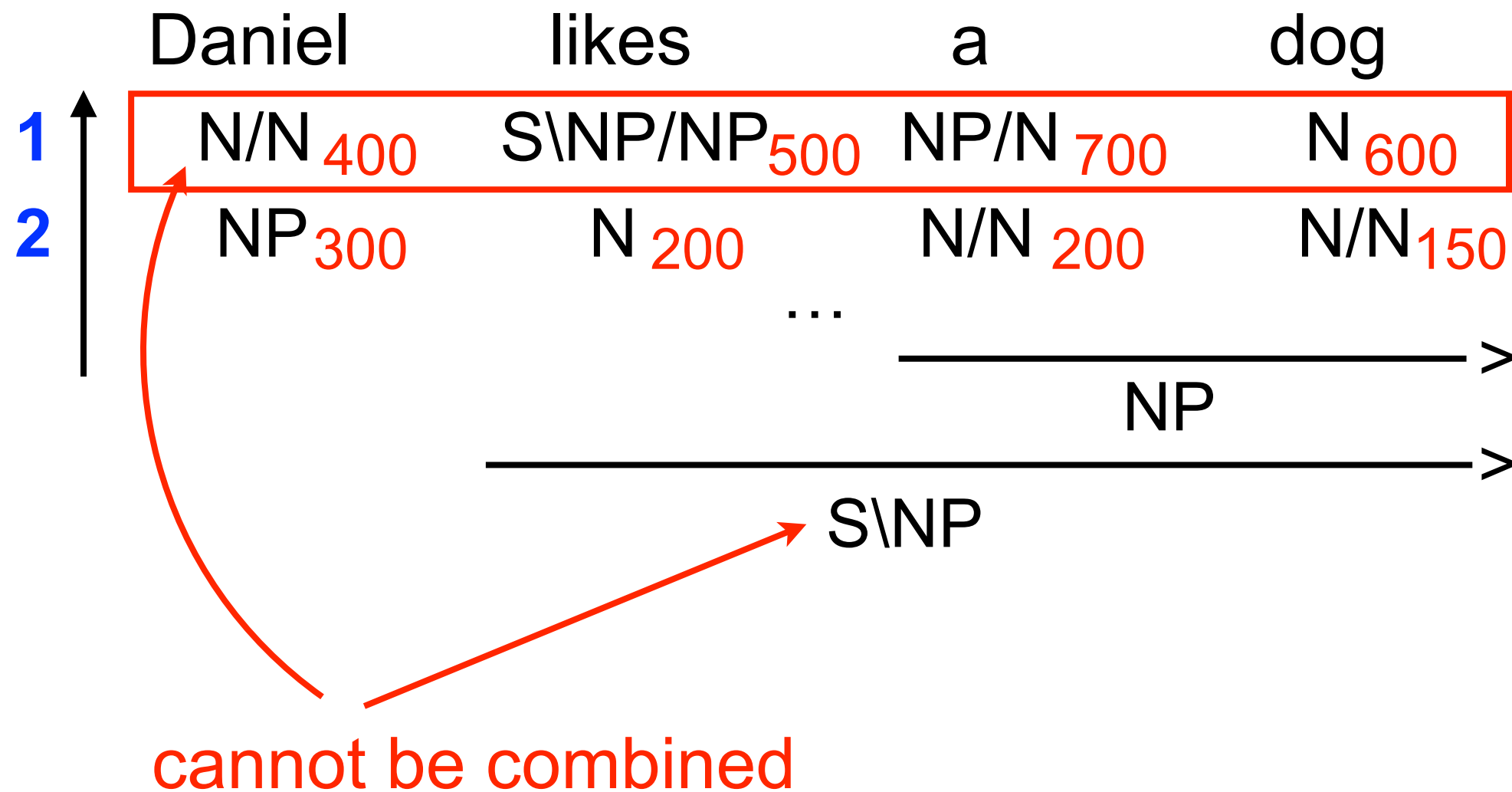
A* CCG parsing

Lewis et al., 2014. *A* CCG Parsing with a Supertag-factored Model*. In EMNLP.

- ▶ Recently, the idea of supertagging was further promoted
- ▶ **New simpler model:**
 - **The score of tree = The score of supertags**
 - Do not score the internal of derivation (applied rules) at all!
 - Lewis et al. found that if the supertagger model is strong (with neural networks), this approach can achieve near state-of-the-art
- ▶ **Advantage of simpler model:** Search gets simpler as well
 - They found that **efficient A* search can be applied** to this model
 - A* search finds the derivation of **a single connected tree that has the highest score**

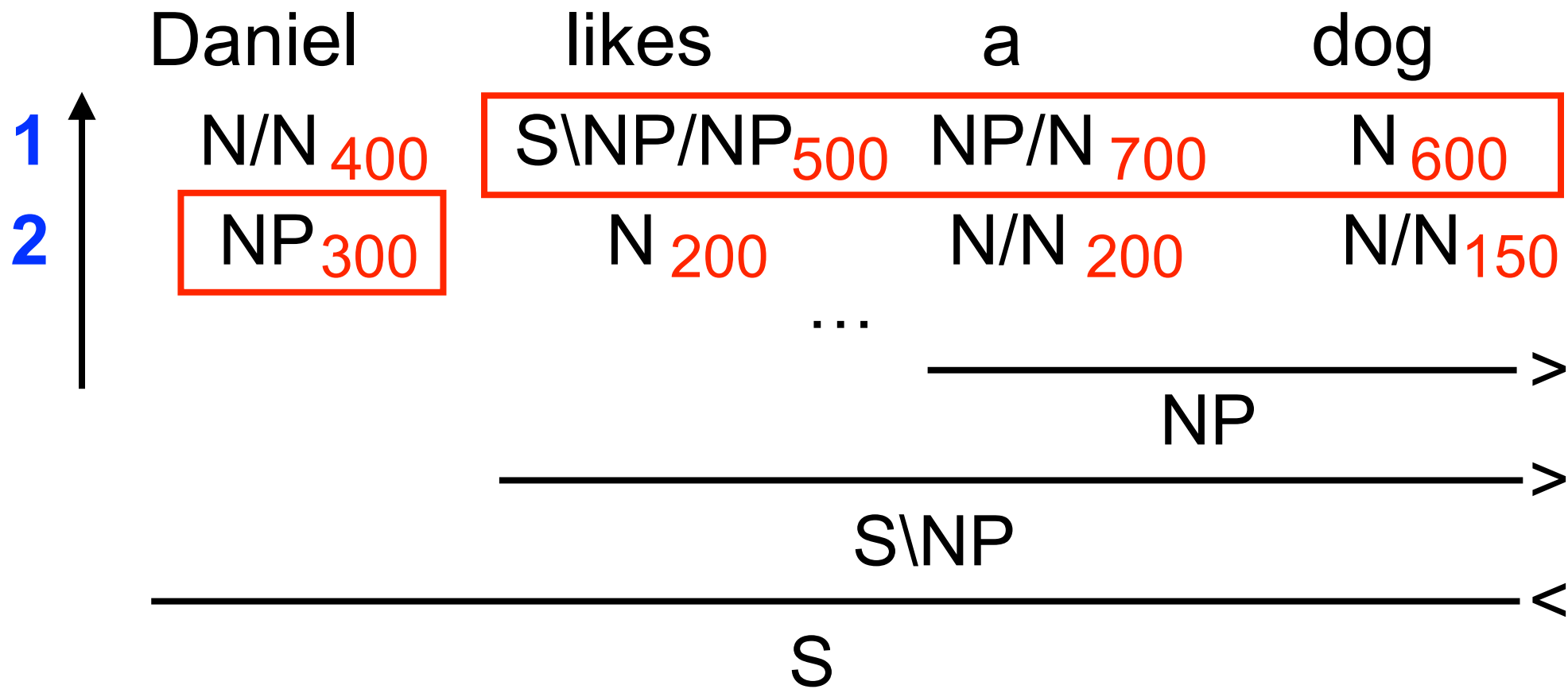
A* CCG parsing

- Note: If we pick up the best supertag on each word, they do not (generally) realize the single connected tree



A* CCG parsing

- ▶ A* search finds the best supertag sequence that realizes a single connected tree



How A* search works?

► A* search expands **the span**, where the best internal structure (= supertag sequence) was found

- **Dynamic programming:** solve the small problems first

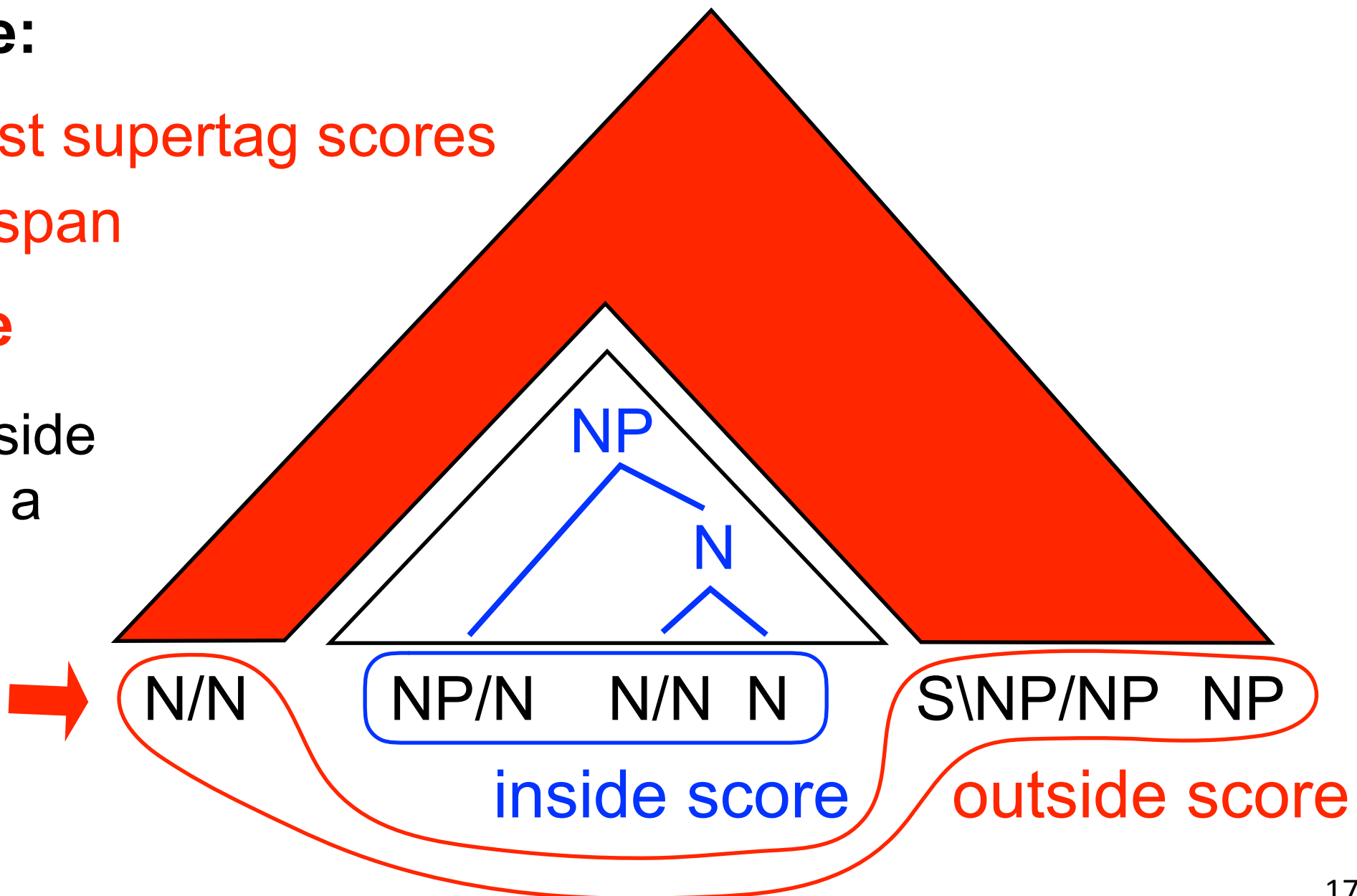
- **Heuristic score:**

- The sum of **best supertag scores outside of the span**

- **Outside score**

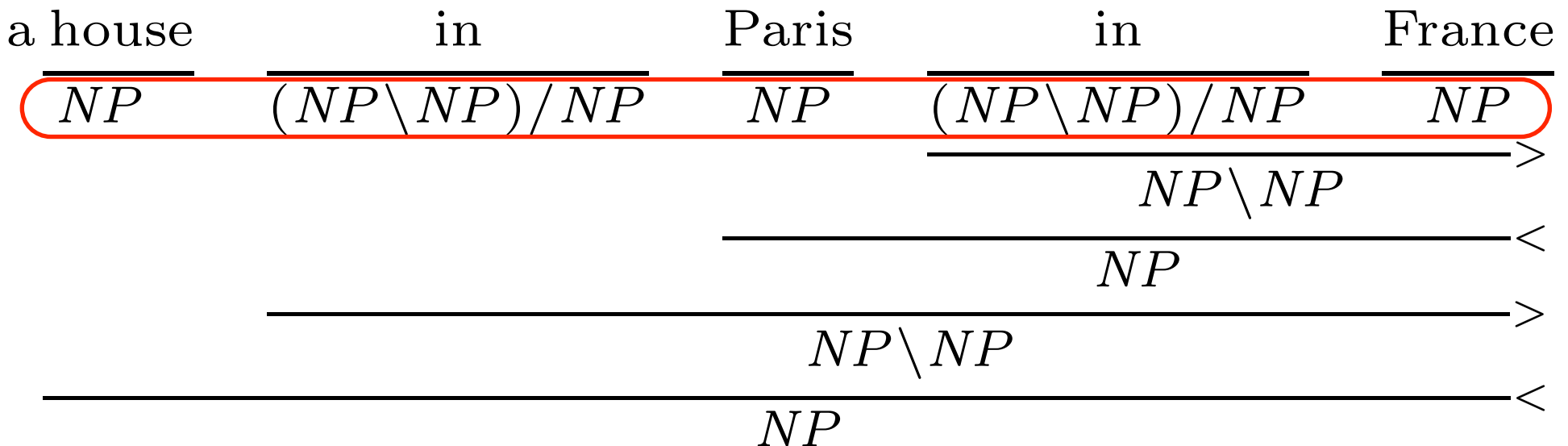
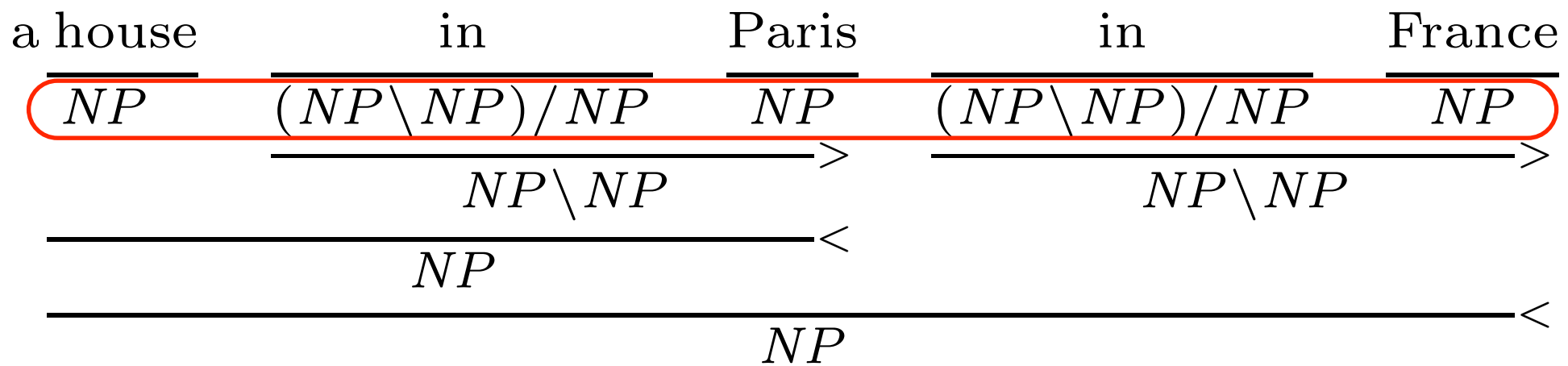
The sequence for outside score may not realize a single tree

But can be a **good approximation of the upper bound** of the true best score



Limitation

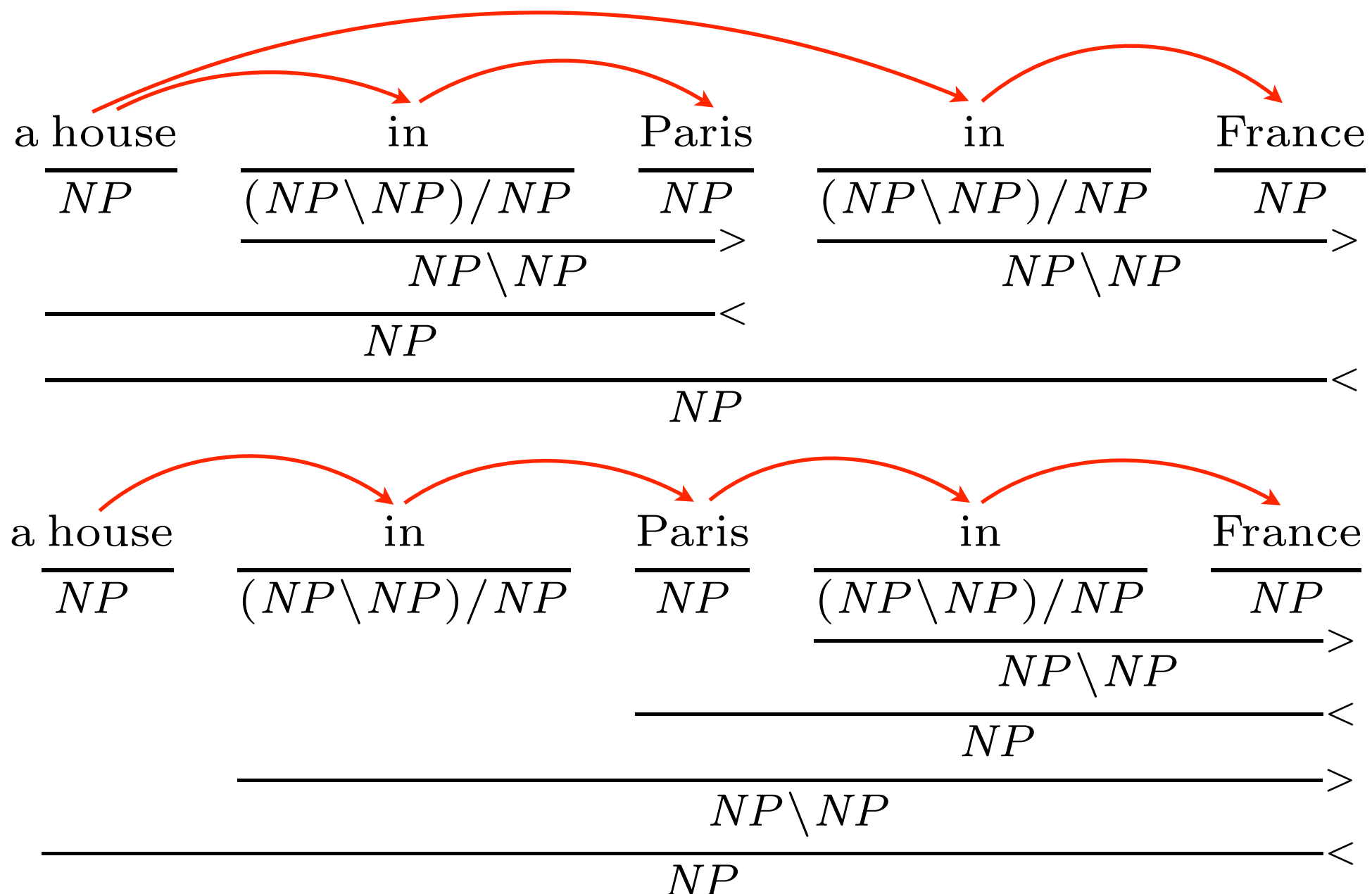
- ▶ For some supertag sequence, the derivation on that cannot be uniquely determined
 - Lewis et al. select the one using some heuristic rule



A* CCG + dependency parsing

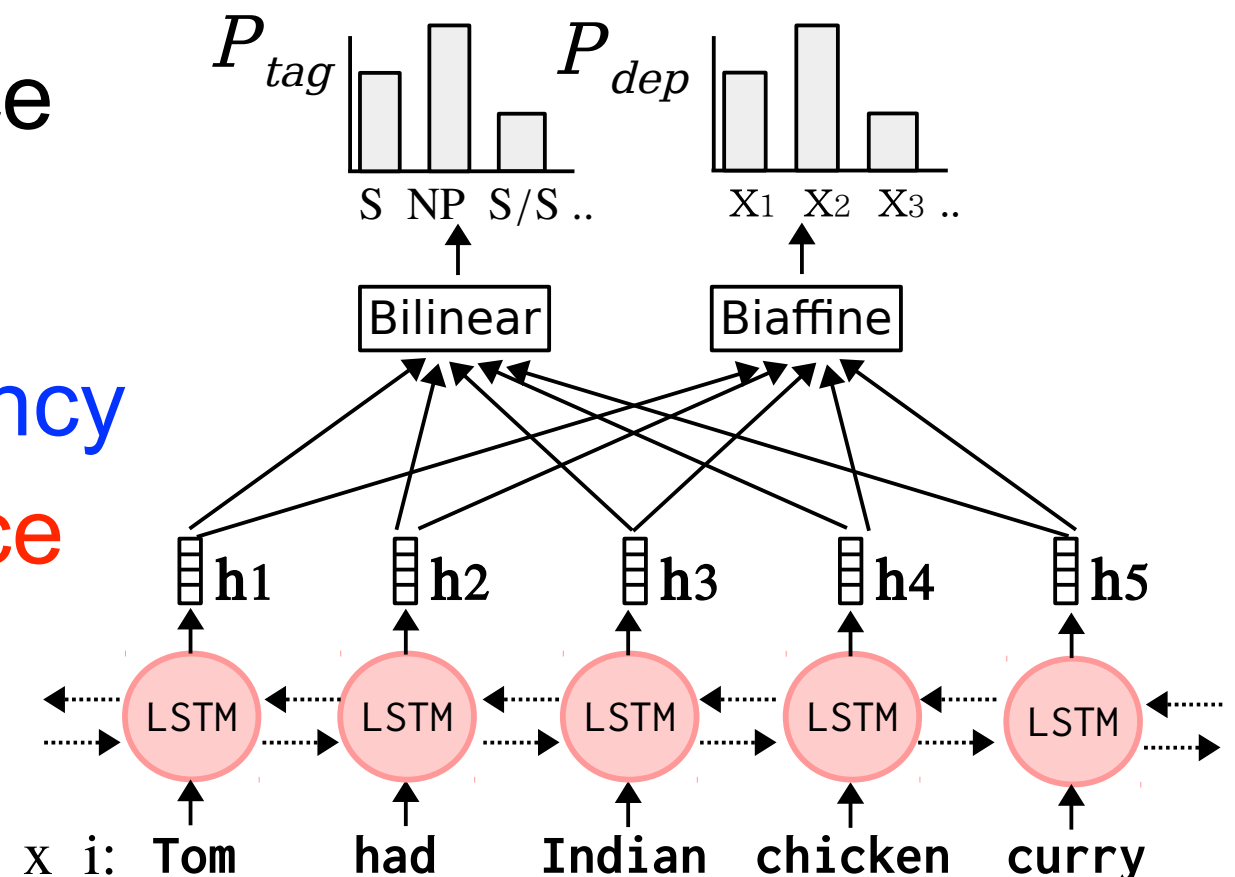
Yoshikawa, Noji, and Matsumoto (2017 ACL)

- **Problem:** supertags cannot resolve all ambiguities
- **Propose:** joint model of supertag and **word-to-word dependencies**



A* search can still be applied

- ▶ We use a simpler dependency model for keeping A* search applicable
- ▶ We use **bidirectional LSTMs** to model both CCG supertags and dependencies (multi-task learning)
- Bi-LSTMs are very powerful neural networks for sequence prediction tasks
- **Point:** we cast the **dependency prediction task** as a **sequence prediction task** to enable A* search



depccg

- ▶ <https://github.com/masashi-y/depccg>
- ▶ Current state-of-the-art for English and Japanese CCG parsing

Outline

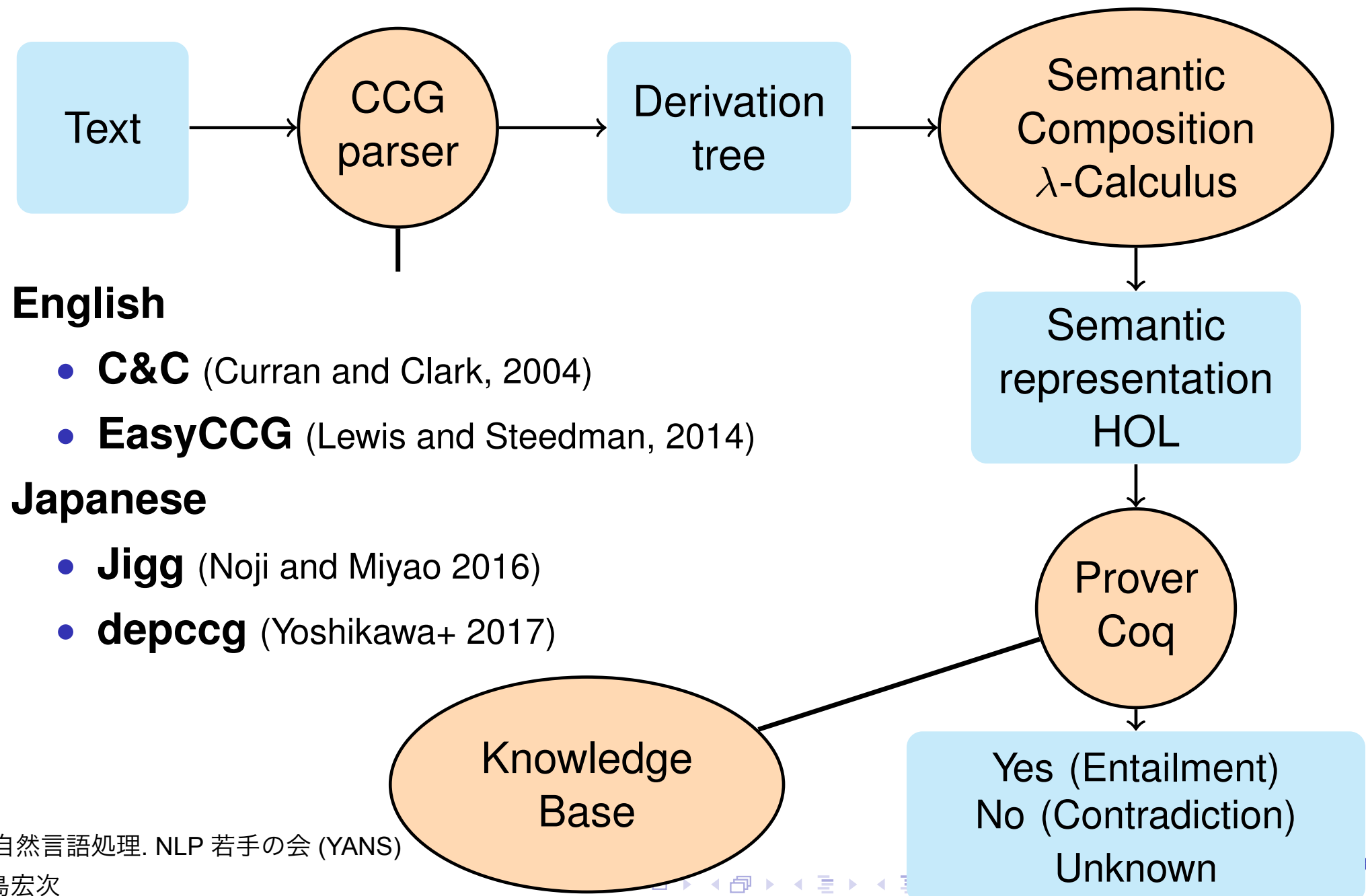
- ▶ Efficient parsing techniques for CCG
- ▶ Bottom-up semantic parsing with CCG
 - application to textual entailment
 - issues in question answering
- ▶ Top-down semantic parsing with CCG for question answering

QA is moved to the next lecture (Thursday)

ccg2lambda

► System for RTE using CCG parsers and logics

- Developed mainly by researchers in NII, AIST, Ochanomizu univ.
- Mineshima et al. (2015, 2016), Martínez-Gómez et al. (2016)



Recognizing textual entailment (RTE)

► **Task:** judge whether the input sentences (**premise**; 前提) semantically **entail** another sentence (**hypothesis**; 仮定)

- Given **P**, **H** is true or false?

P Smoking in restaurants is prohibited by law in most cities in Japan

H Some cities does not allow smoking in public spaces

⇒ **true (entail)**, because **most cities does not mean all cities**

- *The best way of testing an NLP system's semantic capacity* (Cooper, et al., 1994)
- Applications: Summarization, QA on articles, etc.

Challenges

P Smoking in restaurants is prohibited by law in most cities in Japan

H Some cities does not allow smoking in public spaces

► Relationships between content words:

- prohibited → not allowed, restaurants → public spaces

► Logical relations arose with function words:

- most, some, not, etc.
- Implicit logical relations: e.g.,
 - extensional complement: *saw* NP VP → NP VP(ed)
 - intentional complement: *believe* NP VP ↗ NP VP(ed)

► We need to handle both (lexical and logical) aspects

Entailment as logical proof

P Smoking in restaurants is prohibited by law in most cities in Japan

H Some cities does not allow smoking in public spaces

↓ CCG helps

P $\exists x. (smoking(x) \wedge most(\lambda y. city(y), \lambda y. prohibited(x) \wedge in(x, y)))$

H $\exists x. (smoking(x) \wedge \exists y (city(y) \wedge \neg allowed(x) \wedge in(x, y)))$

► Approach:

- try to prove the statement: **P** \rightarrow **H**
- We can use a prover system, which receives a logical form (P \rightarrow H) and returns the following values:
 - yes (entail), no (contradict), and unknown
- ccg2lambda uses Coq as a automatic theorem prover

CCG and logical forms

- ▶ One attractive property of CCG is its transparency between syntax and semantics (among syntactic theories)
- ▶ By assigning a logical form to each word, the sentence logical form can be obtained automatically
 - some: $\lambda F \lambda G. \exists x. (Fx \wedge Gx)$
 - woman: $\lambda x. woman(x)$

$$\begin{array}{c}
 \begin{array}{c} \text{Some} \\ NP/N \\ \lambda F \lambda G. \exists x. (Fx \wedge Gx) \end{array} \quad \begin{array}{c} \text{woman} \\ N \\ \lambda x. woman(x) \end{array} \\
 \hline
 NP \\
 \lambda G. \exists x (woman(x) \wedge G(x))
 \end{array}
 \begin{array}{c}
 \begin{array}{c} \text{ordered} \\ (S \backslash NP) / NP \\ \lambda Q_1 \lambda Q_2. Q_2(\lambda x. Q_1(\lambda y. order(x, y))) \end{array} \quad \begin{array}{c} \text{tea} \\ N \\ \lambda y. tea(y) \end{array} \\
 \hline
 S \backslash NP \\
 \lambda Q_2. Q_2(\lambda x. \exists y. (tea(y) \wedge order(x, y)))
 \end{array}
 \begin{array}{c}
 \text{lex} \\
 NP \\
 \lambda F. \exists y. (tea(y) \wedge F(y))
 \end{array}
 \end{array}
 \begin{array}{c}
 > & & > \\
 \hline
 S \\
 \exists x. (woman(x) \wedge \exists y. (tea(y) \wedge order(x, y)))
 \end{array}
 <$$

Pipeline

- ▶ CCG parser does not output logical forms
- ▶ We first parse the CCG with a CCG parser, and then assign a logical form on each node
 - We only have to assign a logical form to each word
 - Logical forms on the internal nodes are calculated based on the definition of each rule
 - e.g, $X/Y: f \quad Y/Z: g \quad \rightarrow \quad X/Z: \lambda x.f(g \ x) \quad (>B)$

Templates for assigning logical forms

1. For **closed words**: lexical entries directly assigned to surface form: 100 entries (English) and 113 entries (Japanese)

Example

- **category**: NP/N
- **semantics**: $\lambda F \lambda G \lambda H. \forall x (Fx \wedge Gx \rightarrow H)$
- **surf**: every

2. For **open words**: semantic templates for syntactic categories: 129 entries (English) and 37 entries (Japanese)

Example

- **category**: N
- **semantics**: $\lambda x. E(x)$

“E” is a position in which a particular lexical item appears.

Adding lexical knowledge

P $\exists x.(smoking(x) \wedge most(\lambda y.city(y), \lambda y.prohibited(x) \wedge in(x, y)))$

H $\exists x.(smoking(x) \wedge \exists y(city(y) \wedge \neg allowed(x) \wedge in(x, y)))$

- ▶ Logical relations (e.g., most, every) can well be handled by a theorem prover
- ▶ But a prover does not have any world knowledge (e.g., prohibited \rightarrow \neg allowed)
- ▶ ccg2lambda supports adding such lexical knowledge in a proof step:
 - Martínez-Gómez et al., On-demand injection of lexical knowledge for recognizing textual entailment. In EACL 2017.

Adding lexical knowledge

T: men are sawing logs.

$$\exists x.(\text{man}(x) \wedge \exists y.(\text{log}(y) \wedge \text{saw}(x, y)))$$

H: men are cutting wood.

$$\exists x.(\text{man}(x) \wedge \exists y.(\text{wood}(y) \wedge \text{cut}(x, y)))$$

Method: to inject lexical knowledge into the proof.

- Word relations can be found in ontologies (e.g. WordNet, etc.)

$$\forall x \forall y. \text{saw}(x, y) \rightarrow \text{cut}(x, y)$$

$$\forall x. \text{log}(x) \rightarrow \text{wood}(x)$$

SICK dataset

- ▶ The standard dataset for evaluating RTE system
 - ccg2lambda is an unsupervised system (does not use training data)
 - Size: 4,500/500/4,927 for training, dev. and testing.
 - Label distribution: .29/.15/.56 for yes/no/unk.
 - About 212,000 running words.
 - Average premise and conclusion length: 10.6.

Examples:

Problem ID	T-H pairs	Entailment
1412	T: <i>Men are sawing logs.</i> H: <i>Men are cutting wood.</i>	Yes
4114	T: <i>There is no man eating food.</i> H: <i>A man is eating a pizza.</i>	No
718	T: <i>A few men in a competition are running outside.</i> H: <i>A few men are running competitions outside.</i>	Unknown

SICK results

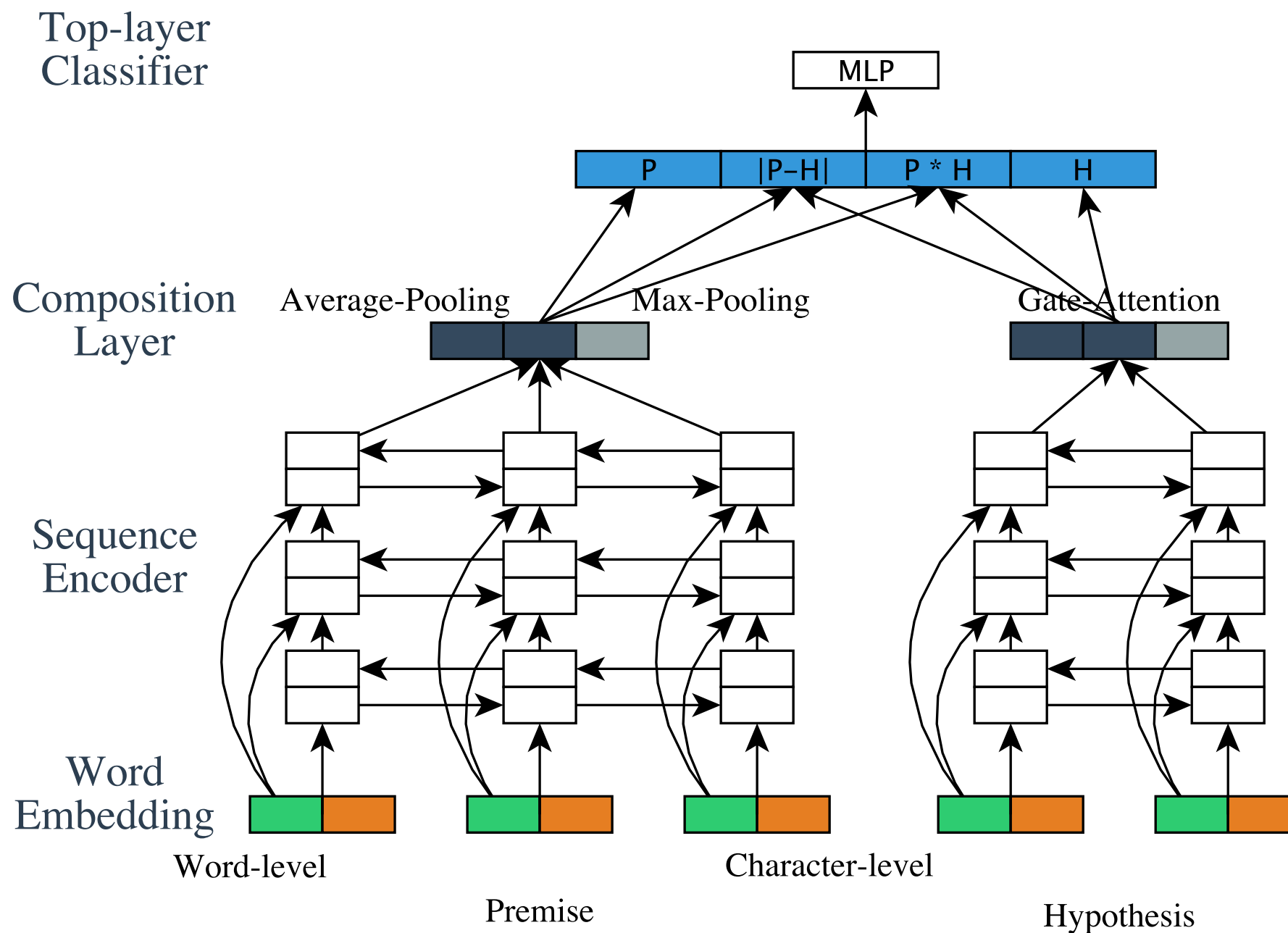
System		Prec.	Rec.	Acc.
Baseline (majority)		—	—	56.69
MLN (Beltagy+ ACL14)		—	—	73.40
Nutcracker	logic-based	—	—	74.30
Nutcracker-WN		—	—	77.50
Nutcracker-WN-PPDB		—	—	78.60
MLN-WN-PPDB		—	—	80.40
LangPro Hybrid-800 (Abzianidze, EMNLP2015)		97.95	58.11	81.35
The Meaning Factory (Bjerva+ SemEval14)		93.63	60.64	81.60
ccg2lambda, No axioms		98.90	46.48	76.65
ccg2lambda, Naïve		92.99	59.70	80.98
ccg2lambda, Abduction (WN, VerbOcean)		97.04	63.64	83.13
SemantiKLUE	supervised machine learning	85.40	69.63	82.32
UNAL-NLP		81.99	76.80	83.05
ECNU		84.37	74.37	83.64
Illinois-LH		81.56	81.87	84.57
MLN-eclassif (Beltagy+ CL2016)		—	—	85.10
Yin-Schutze (EACL2017)		—	—	87.10

Bottleneck in knowledge

- ▶ Logic-based system (including ccg2lambda) shows high-precision and low-recall
 - High-precision: If the system answers “yes” or “no”, often that is correct
 - Low-recall: The system answers “unknown” too aggressively
 - Low-recall is essentially due to the lack of word knowledge
 - WordNet does not cover all relationships between the words
 - Relationships between more than one word (paraphrase, etc.)
 - P: A woman is sewing with a machine
 - H: A woman is using a machine made for sewing
 - Required knowledge:
“sewing with a machine” → “using a machine made for sewing”

Challenges

Alternative approach: deep learning



- We train a neural-network model where an input sentence is encoded in a vector, and some classifier is applied

Deep learning can learn logics?

- ▶ DL methods use a very large corpus (>500K training pairs) to learn a classifier
- ▶ In other words, **it only requires a training data** (and machine resource = GPU)
- ▶ This indicates we may sidestep the complexity of most linguistic phenomena, including qualification (e.g., every, most), with machine learning?
 - like visual recognition on ImageNet?

FraCaS dataset

fracas-067

Premise 1 All residents of the North American continent can travel freely within Europe.
Premise 2 Every Canadian resident is a resident of the North American continent.
Hypothesis All Canadian residents can travel freely within Europe.
Answer Yes

fracas-074

Premise 1 Most Europeans can travel freely within Europe.
Hypothesis Most Europeans who are resident outside Europe can travel freely within Europe.
Answer Unknown

There are multi-premise problems

quantifier section

- ▶ A dataset developed by a theoretical linguist
- ▶ Divided into sections by the relevant linguistic phenomena
 - Generalized quantifier, Plurals, Comparatives, etc.
 - A collection of **logically difficult problems**

Deep learning on FraCaS

Section	#	Ours	Nut	Bow16
Quantifiers	74	.77	.53	.64*
Plurals	33	.67	.52	.54*
Adjectives	22	.68	.32	.47*
Comparatives	31	.48	.45	.56*
Verbs	8	.62	.62	.62*
Attitudes	13	.77	.46	.67*
Total	181	.69	.50	

- ▶ Ours: ccg2lambda
- ▶ Nut: Previous logic-based system (Nutcracker)
- ▶ Bow16: A deep learning model learned with a huge amount of training data (called SNLI)
- ▶ Note (*): Single premise problems only (Bow16 cannot handle multi-premises)

Discussion

- ▶ Currently deep learning models do not work well on FraCaS
- ▶ It is a very open problem whether a strong machine learning technique, and big data, are enough to be able to understand the language
- ▶ Advantage of bottom-up approach (ccg2lambda):
 - High interpretability of the results:
 - we can see the reason why the system fails (e.g., missing knowledge)
 - Extensibility: e.g., we can add more lexical knowledge, if we have
- ▶ **Can we integrate these two very different approaches?**