

# Midterm stuffs

---

Misspelling detection and correction

# KBTG-CU-NECTEC NLP

## KBTG จับมือ NECTEC และจุฬาฯ เปิดตัว Thai NLP สร้างอีโคซิสเต็มพัฒนา NLP ภาษาไทย

By: sponsored on 16 December 2019 - 12:24

Tags: Kasikorn Bank NECTEC Artificial Intelligence Natural

Language Thailand Advertorial



KBTG หรือ KASIKORN Business-Technology Group ประกาศความร่วมมือกับ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC ในสังกัด สวทช.) และจุฬาลงกรณ์มหาวิทยาลัย เปิดตัวโครงการ Thai NLP ร่วมพัฒนาเทคโนโลยีประมวลผลภาษาธรรมชาติของภาษาไทย โดยเปิดเป็น Open API และตั้งเป้าสร้างแพลตฟอร์มนิเวศน์ไปจนถึงชุมชนของ NLP ภาษาไทยขึ้นมาเอง โดยมีคุณชัชติยา อินทริวชัย กรรมการผู้จัดการ ธนาคารกสิกรไทยมาเป็นประธานในการเปิดตัว NLP



# Leaderboard

#	Team Name	Notebook	Team Members	Score ⓘ	Entries	Last
1	sdfdf			0.92324	38	3h
2	schwanabc			0.92199	69	4h
3	Phuchit Chadasin			0.91737	74	1d
4	daidew			0.91711	33	9h
5	Kanokphat Jinanarong			0.91553	16	3h
6	Error			0.91013	16	5h
7	Knot Nuttanai			0.91005	56	3h
8	time131			0.90949	36	12h
9	Nonthanat			0.90847	60	3h
10	Pisit Wajanasara			0.90846	70	3h

#	△pub	Team Name	Notebook	Team Members	Score ⓘ	Entries
1	▲ 2	Phuchit Chadasin			0.91748	74
2	—	schwanabc			0.91561	69
3	▼ 2	sdfdf			0.91237	38
4	▲ 8	teeti watanatada			0.90730	37
5	▲ 1	Error			0.90691	16
6	▼ 1	Kanokphat Jinanarong			0.90656	16
7	▲ 1	time131			0.90635	36
8	▲ 2	Pisit Wajanasara			0.90555	70
9	▼ 5	daidew			0.90303	33
10	▼ 3	Knot Nuttanai			0.90240	56

# Interesting ideas

Good results need some char-based elements (or both)

Need sequence of words as input (even trigram can beat the TA baseline)

Tuning the threshold is important

Augment training set of sentences with misspellings by shuffling words

Exact architecture design does not really result in the difference in performance

# Post processing!

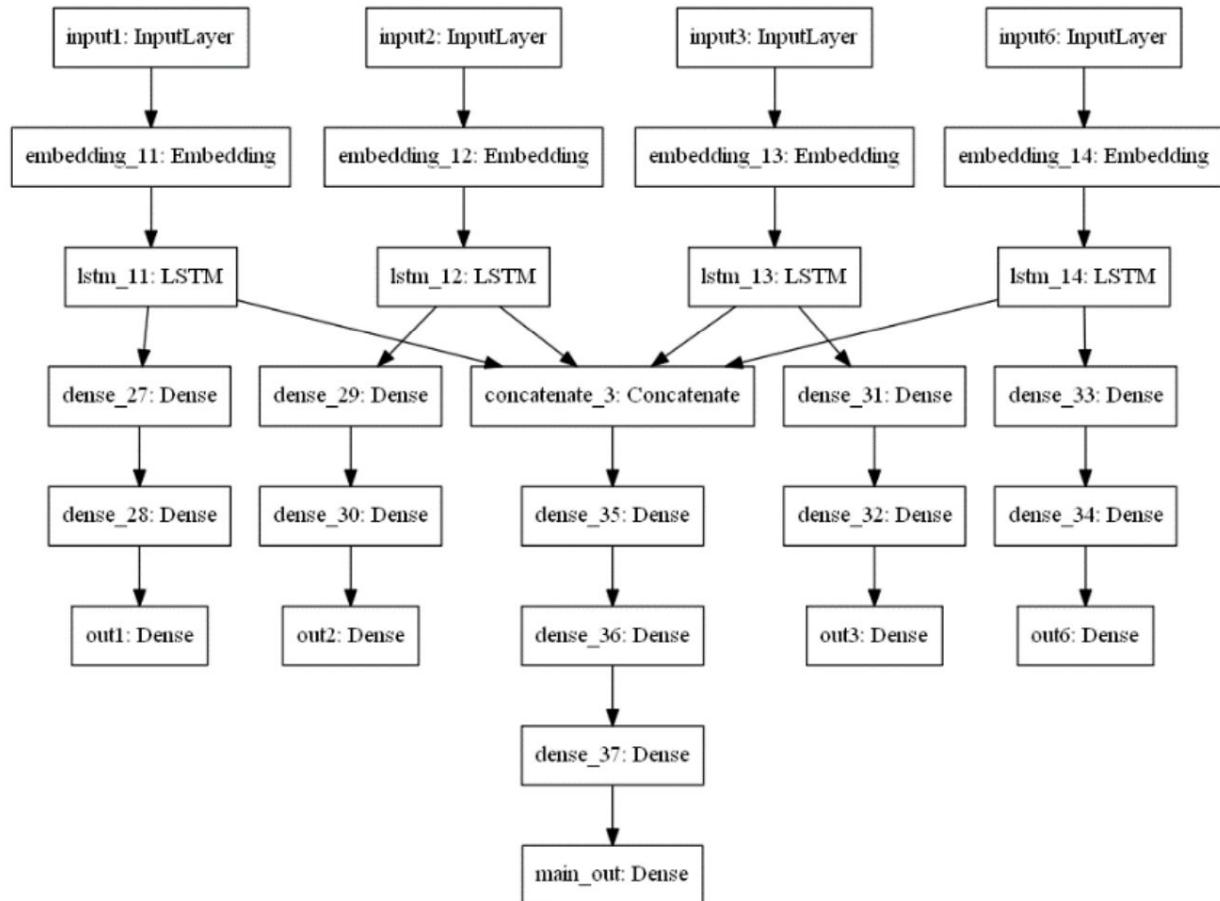
หลังจากใช้ ML model อย่างเดียวในการตรวจพบว่าได้ GINI อยู่ที่ 0.80 จึงได้ลองสร้าง validation set ขึ้นมา 15% ของ training set ทั้งหมด เพื่อดูว่าคำแบบไหนที่ไม่เดลยังตอบไม่ถูก จึงได้พบว่าส่วนใหญ่คำว่า คง และ ค่ะ นั้นโมเดลไม่สามารถ分辨ได้ว่าผิดหรือถูก จึงได้สร้าง rule-based เพื่อแก้ปัญหานี้ โดยจะมี list คำที่ เป็นไปได้ของคำก่อนหน้าสองคำของคำว่า คง ตัวอย่างเช่น ถ้าคำปัจจุบันคือ คง คำก่อนหน้าสองคำต้องมีคำใดคำหนึ่งอยู่ใน [“ยังไง”, “ใหม่”, “หรอ”] จึงจะถือว่าคำว่า คง ในที่นี่สะกดถูก และกลับกัน ถ้าคำปัจจุบันคือ ค่ะ แต่ ส่องคำข้างหน้าอยู่ใน list ก็จะถือว่า คง สะกดผิด ซึ่ง list ของคำเหล่านี้จริงๆที่ใช้มีหลายคำมาก ส่วนใหญ่เป็นคำที่ เป็นเชิงประโยคคำนาม หลังจากเพิ่มตรงนี้เข้าไปจะแนนกับเข้าใกล้ base line จากนั้นก็มีการเพิ่มบาง rule เข้าไป นิดหน่อยแต่ไม่ได้ช่วยมาก และ tune ไม่เดลเรื่อยๆแล้วลองส่งกับผ่าน baseline ได้คะแนนประมาณ 0.899

```
ka_list = ['นะ', 'นะ', 'ไหน', 'ป่าว', 'รีป่าว', 'หรือป่าว', 'หรือเปล่า', 'นัย', 'นัย', 'นัย', 'พี', 'คุณ', 'บัง', 'ยังไง', 'ไง',  
'จัย', 'ใจ', 'ใน', 'ไม่', 'หรอ',  
'ไหน', 'ไหน', 'ไหน', 'นะ', 'ป่าว', 'ละ', 'ล่ะ', 'ยังงัย', 'อะไร', 'อะรัย', 'เห่าไหร', 'เห่าไร', 'ไร', 'รัย',  
'หรือใน', 'อย่างไร', 'หรือ', 'หรอ', 'นิ', 'หนิ', 'บัง', 'ทำไม', 'ก']
```

# 3rd

## Multi-path char gram and multi-head Combine word and char embedding

**Input:** characters unigram [ '<s>', 'ໜ', 'ສ', 'ີ', 'ອ', '<s/>' ]  
characters bigram [ '<s>ໜ', 'ໜສ', 'ສີ', 'ີອ', 'ອ<s/>' ]  
characters trigram [ '<s>ໜສ', 'ໜສີ', 'ສີອ', 'ອ<s/>' ]  
current with windows size of 1 word [ '<s>', 'ໜສີ', 'ອ' ]  
4 heads with different inputs

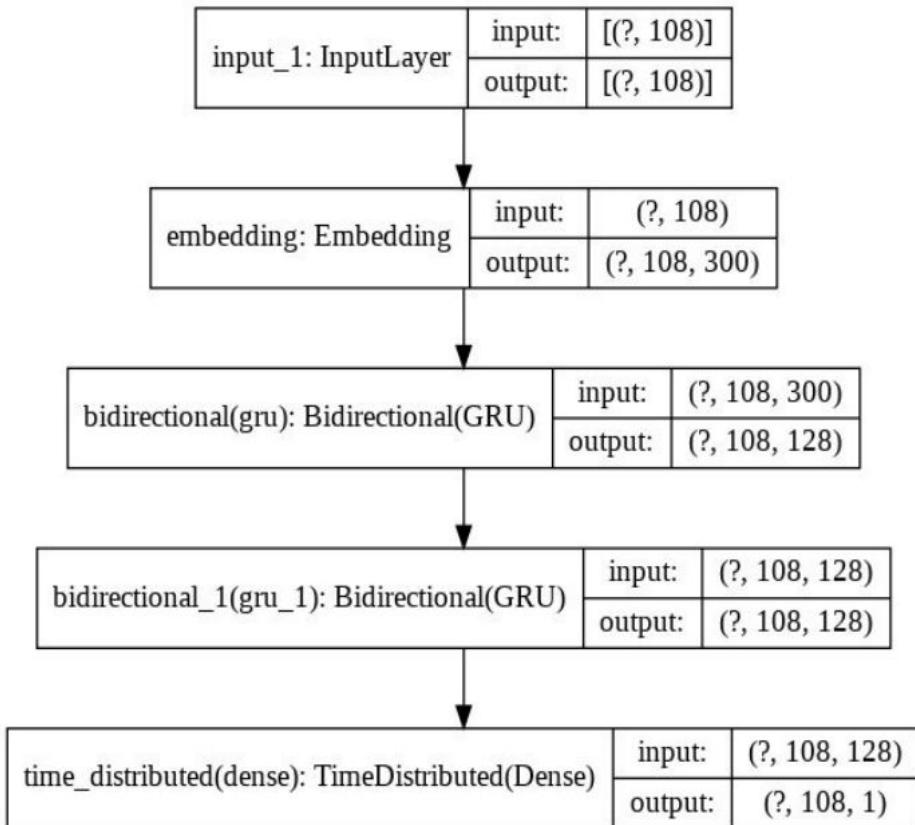


# 2nd

Word-based model  
Ensemble using “vote entropy” (69 submissions)

Method	Public scoreboard normalized Gini
Ensembles	<b>0.92199</b>
Fasttext(dim = 300)	0.91805
Thai2fit(dim = 300)	0.91310
random(dim = 64)	0.90742

Performance on the public scoreboard



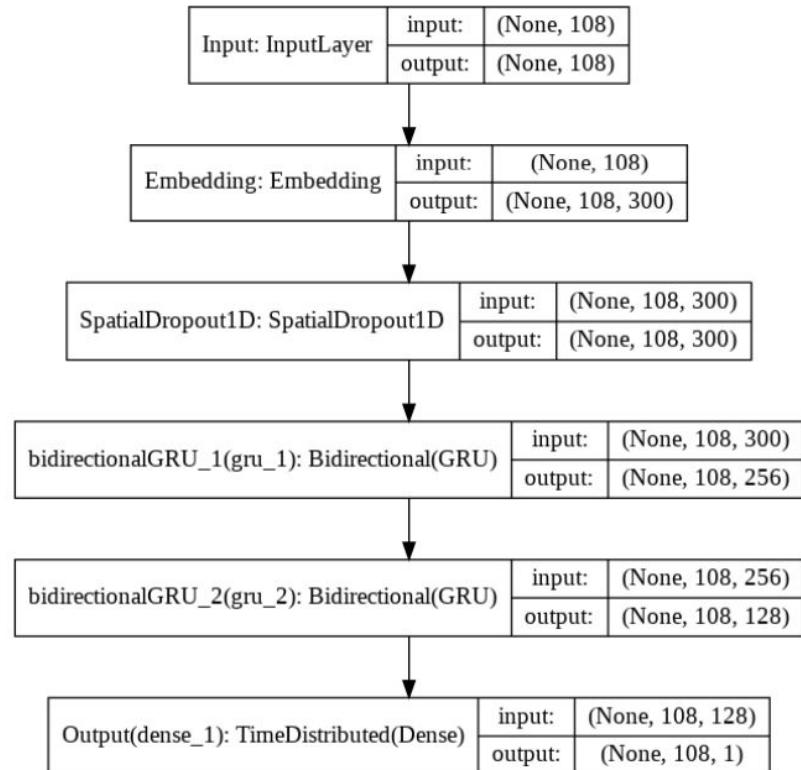
# 1st

Word-based model

Pythainlp pretrain word vector

I assume spatial dropout  
helps?

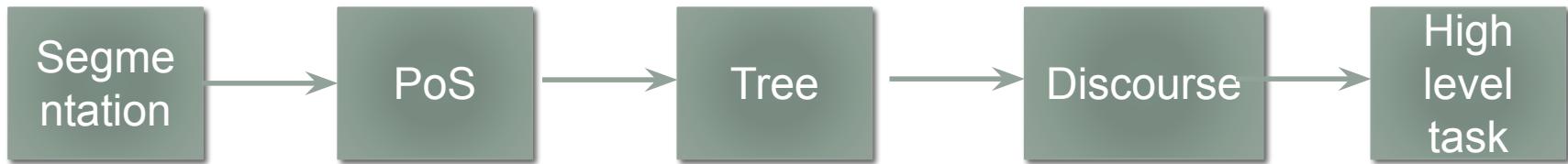
Just tune the threshold (74  
submissions)



# Error propagation and NLP systems

- Typical flow of NLP systems

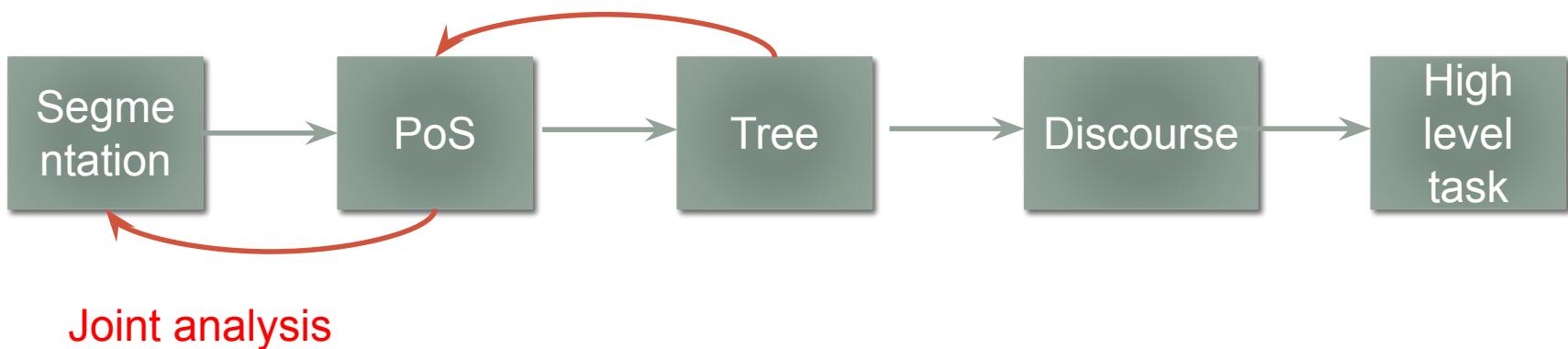
เข้าต้องมีหมอนกอด VS เข้าต้องมีหมอรกอด



Pipeline analysis

# Error propagation and NLP systems

- Use higher level info to help lower level
  - Iterative process



# Error propagation and NLP systems

- Do the task together



Joint analysis

# Error propagation and NLP systems

- One single task
  - Input closest to the input form – characters, words

High level task with low level structures as input

End-to-end analysis

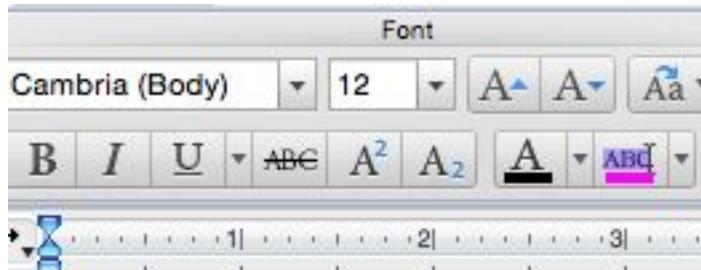
# Spell correction

natural lanuge processing

All Images Videos News Books More

About 3,180,000 results (0.52 seconds)

Showing results for natural ***language*** processing  
Search instead for natural lanuge processing



Rates of spelling errors ~1-20% depending on application

# Tasks

- Spell error detection
- Spell error correction
  - Suggestion list
  - Suggest a correction
  - Autocorrect



Confidence of the system

On my way to hell :)

school\*

damn autocorrect

# Type of spelling errors

- Non-word errors
  - Languaeg -> Language
- Real-word errors
  - Typographical errors
    - Three -> there
    - เข้าต้องมีหมอรกอด
  - Cognitive errors (homophones)
    - piece->peace
    - too->two
    - គែ -> គោ
- Dictionary is required.

# Corpora

- What kind of text?
  - Pantip, twitter, wikipedia, textbook, ...
- Extra: How to actually construct the training corpora?
  - Look at standard sentences and add misspellings
    - By keyboard distance, etc.

# Possible solutions

Detecting errors:

- Segmentation failure (word not in dictionary)

- Using prediction model

Correcting errors:

- Simple N-grams

- CBOW (neural based)

- Edit distance

End-to-end

- Seq2seq model (machine translation-based)

# A note on dictionary

- Require a large dictionary (millions) to handle real world internet usage
- Automatically generated from unigrams list
  - This list includes misspellings
  - Misspellings occur less frequently than correct words in similar context
- Or let users add new words when not accepting corrections
- Cute solution using NER to ignore NE words if OOV

# Assumptions

- This only handle one misspelled word per sentence?
  - How to handle multiple missing words in a row?
- Word level correction:
  - Cannot correct OOV words
- Character level correction:
  - Detecting misspellings in the character level might not be able to handle deletion errors

ฉ น ໄ ป เท ย ว

0 0 0 0 0 0 0 1 0 0

ฉ น ໄ ป เท ย ว

0 0 0 0 0 0 0 0 0

# Assumptions

- Discourse?
  - Only a couple of students actually talks about discourse

A: เดี๋ยวพรุ่งนี้เราต้องไปที่น่าน

B: จะไปที่**น่าน**ทำไมละ

นนท์ vs น่าน vs นั่น

# Spell correction in literature

1. Non-word
2. Real-word

# Non-word spelling error

- Detection:
  - Any word not in dictionary is an error
- Correction:
  1. Generate candidates
  2. Choose the one that has the best
    1. Smallest edit distance
    2. Best noisy channel probability

# Real word spelling error

- Detection:
    - ???
  - Correction:
    1. Generate candidates
      1. From similar pronunciations
      2. From similar spellings
    2. Choose the one that has the best
      1. Smallest edit distance
      2. Best noisy channel probability
- The criterions has to be **context sensitive!**  
ไปไหนดีค่ะ -> ไปไหนดีคะ

# Candidate generation

- Words with similar spelling
  - Edit distance
  - Typically at most 2 edit distance
  - Can be weighted based on the task, e.g. keyboard
  - Allow merging and splitting of words
    - Ladygaga -> Lady Gaga
    - |การะ|เกด| -> การะเกด
- Words with similar pronunciation
  - Example: Aspell, Soundex – a hash based on sounds
    - Jurafsky, Jarofsky, Jarovsky, and Jarovski all map to J612
  - Modern methods use Letter-2-sound (L2S) rules (also called Grapheme-to-Phoneme G2P)
- How to generate this in an efficient manner?

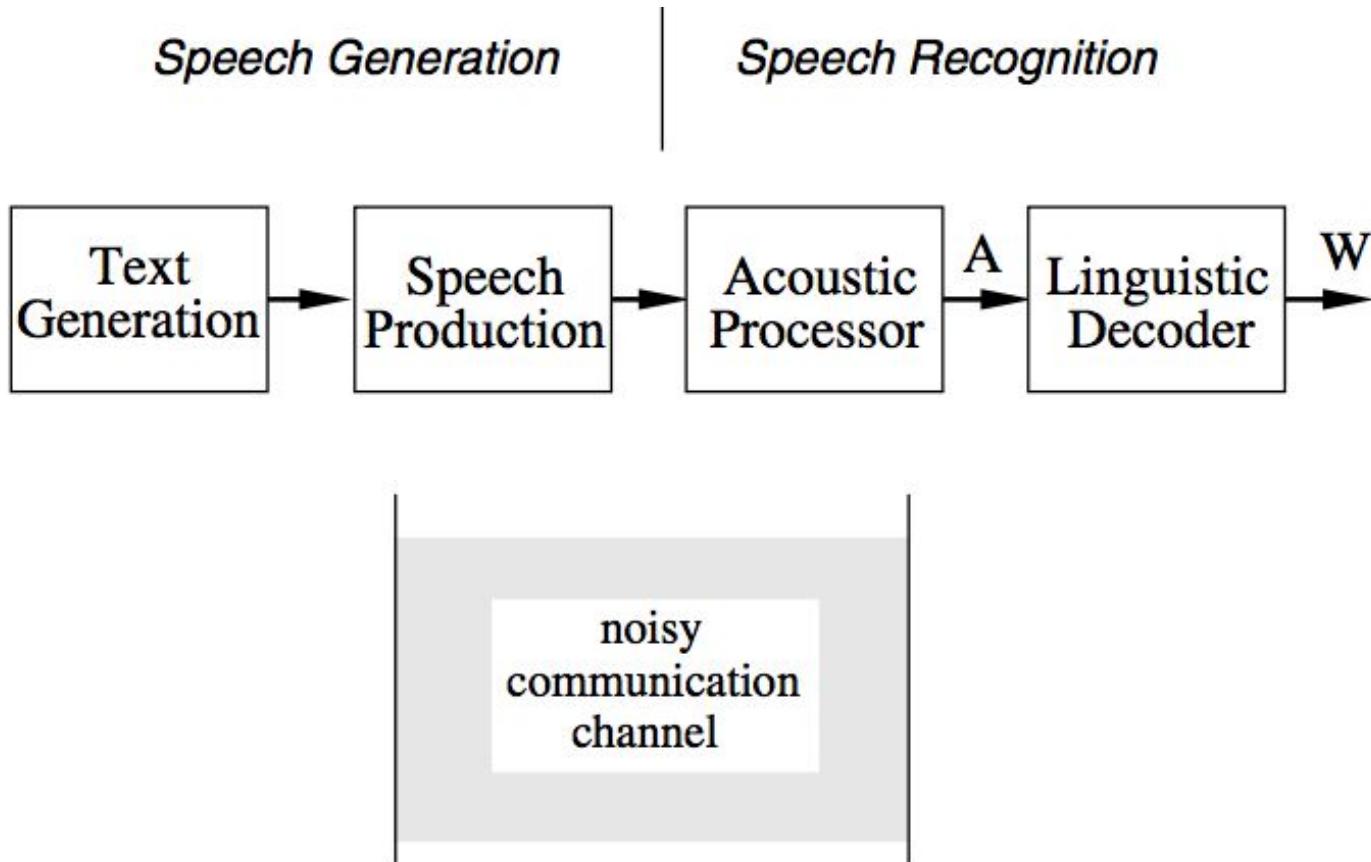
# Candidate generation

1. Run through dictionary, check edit distance (slow)
2. Generate all words with desired edit distance, then intersect with dictionary
3. Character n-gram inverted index. Find words which shared the most n-gram (fast)
  - Google -> goo oog ogl gle
  - Inverted index: goo -> google good goon good-bye
4. Compute with Finite State Transducer (FST) (fast)
5. Precomputed map of words
6. Mixture ( <https://github.com/wolfgarbe/SymSpell> ) (fast)

# The noisy channel model

- Used to rank the best candidates

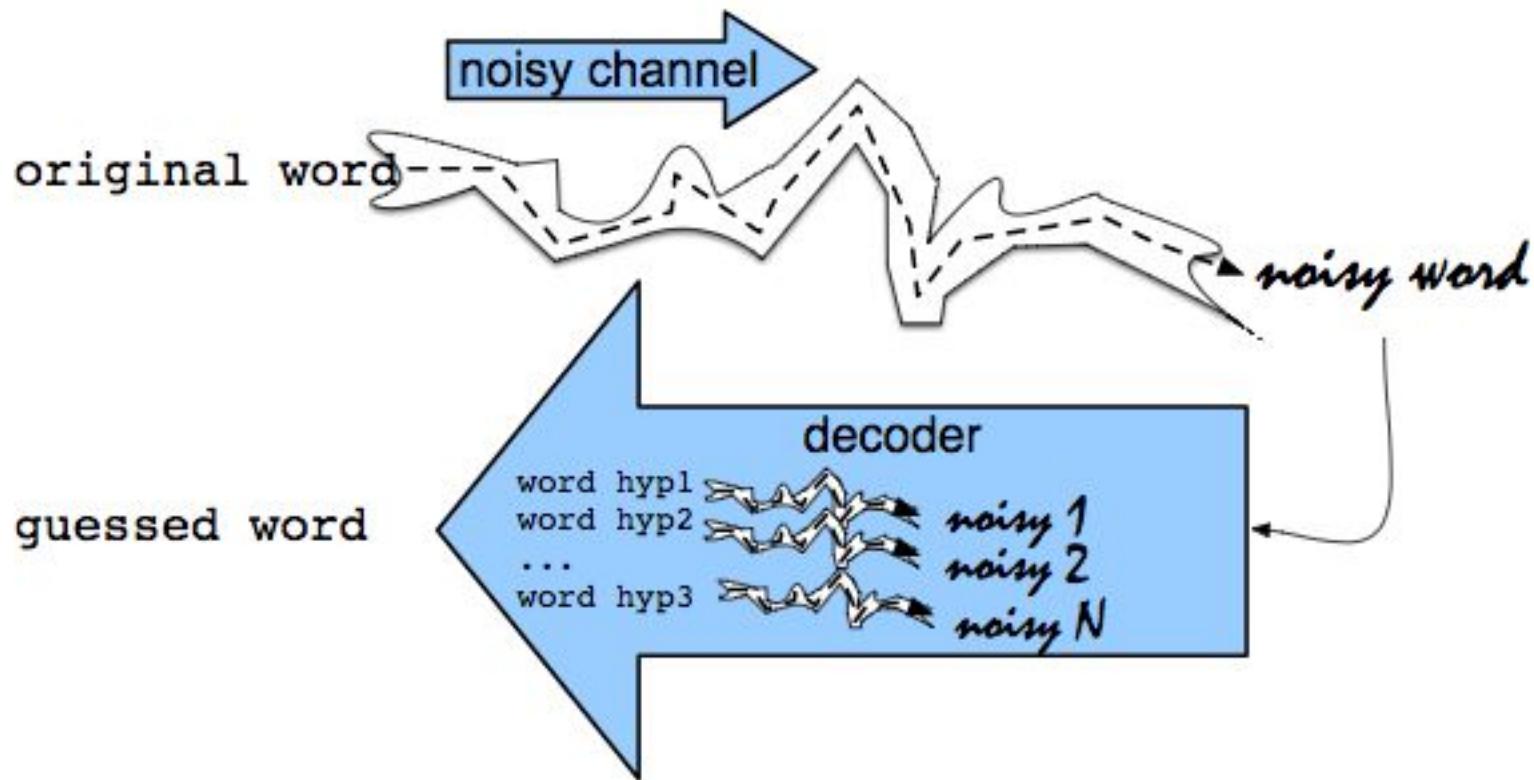
# Information theoretic formulation



$$W^* = \arg \max_W P(W | A)$$

$$P(W | A) = \frac{P(A | W)P(W)}{P(A)}$$

# The noisy channel intuition



# Bayes rule

- An observation  $x$  of the misspelled word
- Find the correct word  $w^*$  from vocab list  $V$

$$\begin{aligned} &= \operatorname{argmax}_{w \in V} P(w|x) \\ &= \operatorname{argmax}_{w \in V} \frac{P(x|w)P(w)}{P(x)} \quad \text{Language model} \\ &= \operatorname{argmax}_{w \in V} P(x|w)P(w) \quad \text{Channel/error model} \end{aligned}$$

# Noisy channel example

- actress
- Let's generate candidates

# Noisy channel example

Edit distance 1 (Damerau-Levenshtein distance)

Levenshtein edit distance – insertion, deletion, substitution

Damerau-Levenshtein edit distance - insertion, deletion, substitution, **transposition**

Error	Correction	Transformation			(Letter #)	Type
		Correct Letter	Error Letter	Position		
acress	actress	t	—	2		deletion
acress	cress	—	a	0		insertion
acress	caress	ca	ac	0		transposition
acress	access	c	r	2		substitution
acress	across	o	e	3		substitution
acress	acres	—	s	5		insertion
acress	acres	—	s	4		insertion

# Language model (unigram)

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

# Channel/error model

- Easiest to use a confusion matrix conditioned on the previous character.
- Based on counts

$$P(x|w) = \begin{cases} \frac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1}w_i]}, & \text{if deletion} \\ \frac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_iw_{i+1}]}, & \text{if transposition} \end{cases}$$

$w_i$  is the i-th character

# Learning the confusion matrix

1. Have a data of misspellings-corrections pair
2. Have a data of misspellings and use EM
  1. Initialize the confusion matrix to be equally likely
  2. E-step: run the spell checker and correct words
  3. M-step: re-estimate the confusion matrix based on corrections

Or use nearby keys  
(Each phone need to have different model)



# Channel/error model

Candidate Correction	Correct Letter	Error Letter	$x w$	$P(x w)$
actress	t	-	c ct	.000117
cress	-	a	a #	.00000144
caress	ca	ac	ac ca	.00000164
access	c	r	r c	.000000209
across	o	e	e o	.0000093
acres	-	s	es e	.0000321
acres	-	s	ss s	.0000342

# Noisy channel model

Candidate Correction	Correct Letter	Error Letter	x w	Error		
				P(x w)	P(w)	$10^9 * P(x w)P(w)$
actress	t	-	c ct	.000117	.0000231	2.7
cress	-	a	a #	.00000144	.000000544	0.00078
caress	ca	ac	ac ca	.00000164	.00000170	0.0028
access	c	r	r c	.000000209	.0000916	0.019
across	o	e	e o	.0000093	.000299	2.8
acres	-	s	es e	.0000321	.0000318	1.0
acres	-	s	ss s	.0000342	.0000318	1.0

Note1: It is typical to use more than unigrams for the LM (very important for real word errors)

Note2: typically added a weight parameter tuned on dev set

$$P(x|w)P(w)^\alpha$$

# Real word case for noisy channel

- Need to compare candidates against no correction
  - Error model needs a score for no correction
- Set to some parameter,  $c$ , representing how errorful our input is
- $P(w|x) = c$  for  $w = x$
- The rest of the probability  $1-c$  is weighted by the confusion matrix

# Classifier-based methods

- Have a pair of correction and features
  - whether/weather
  - Features:
    1. cloudy in +- 5 words
    2. Followed by “to VERB”
    3. Followed by “or not”
- End-to-end model using neural networks
  - Seq2Seq model (more on this in later lectures)

# Type of errors in Thai social

Table 1: Examples of different types of errors and their respective correction.

Type of Error	Occurrence (%)	Error	Correction
Misspelling	61.38	ทຸກคน คວ່າບຕຣ	ທຸກຄນ ຄວ່າບາຕຣ
Morphed	24.00	ครັບ ຕິ້ລຳຄົກຄຄ	ครັບ ນ່າຮັກ
Abbreviation	15.00	ມຄ ພນ	ມ.ຄ. ພຽງນີ້
Spoonerism	0.14	ພັບກນ	ພບກັບ
Slangs	0.08	ຕື່ເນີຍ ອ້ອຍ	No correction ທອດສະພານ
Other	5.63	ໂຮບິນສັນ	No correction

# Error detector

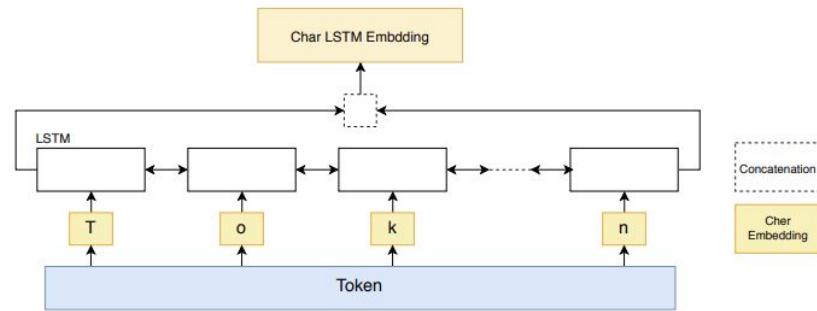


Figure 3: Character LSTM embedding layer encoding a word token.

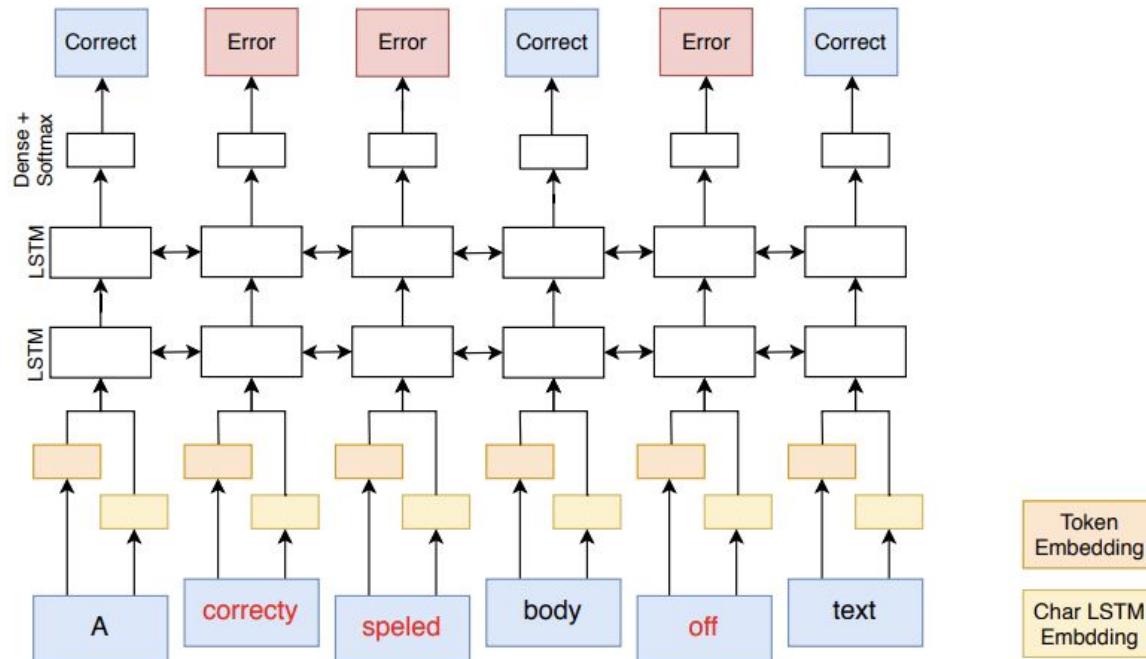


Figure 2: Error Detector operating on a sequence.

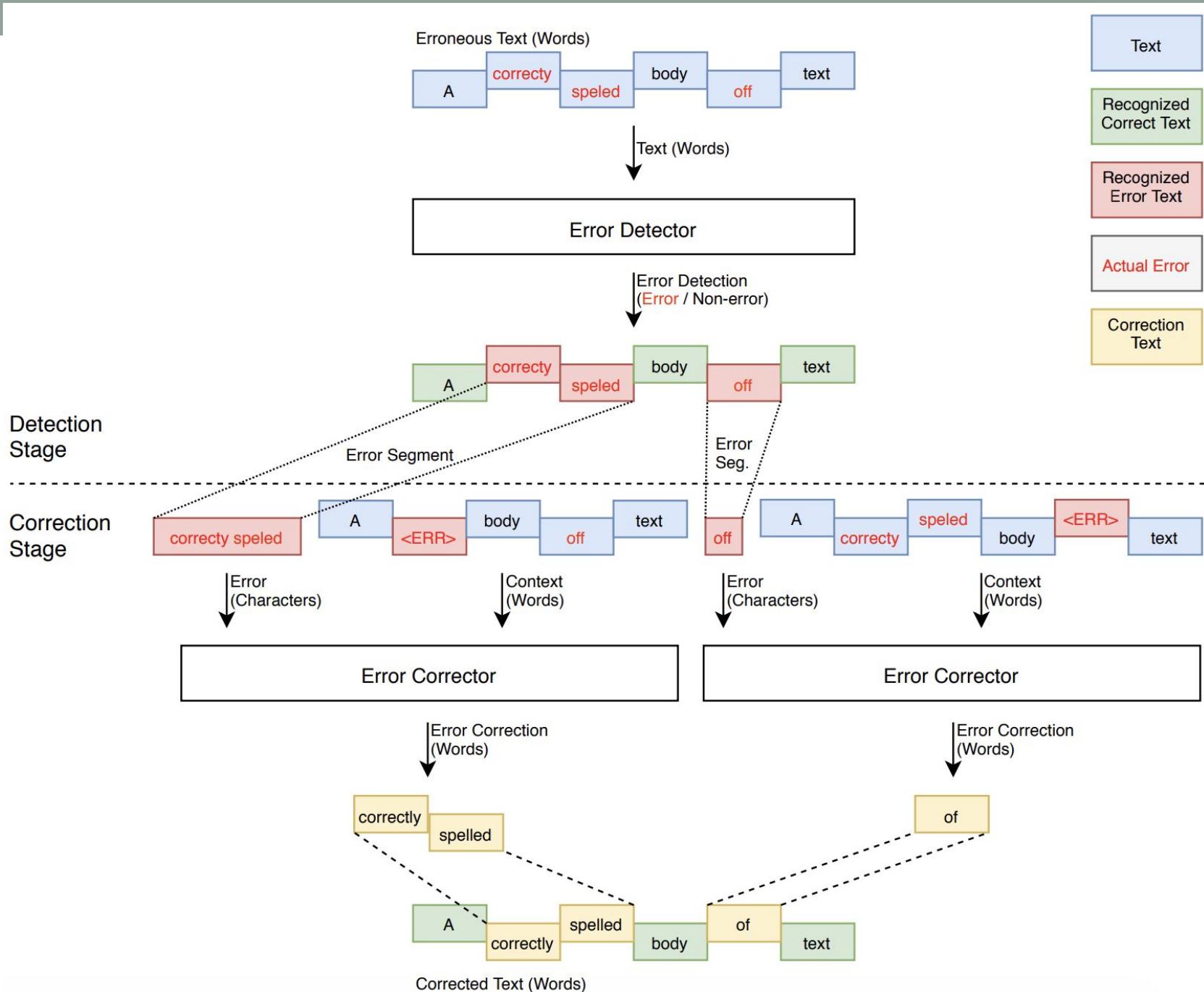


Table 3: Evaluation of end-to-end error correction on Thai UGWC test-set on various systems

Model	Type	GLEU	WER (%)	$\Delta$ WER (%)
Do nothing (source text)	-	0.8845	3.77	0.00
Ideal correction (Oracle)	-	1.0000	0.00	-100.00
<b>Off the shelf</b>				
Hunspell	2-Stage	0.8267	8.11	+115.12
PyThaiNLP	2-Stage	0.8612	5.58	+48.01
<b>Trained</b>				
Hunspell	2-Stage	0.8598	5.57	+47.75
Bi-GRU (180 token limit)	E2E	0.4035	50.82	+1,247.92
Bi-GRU (20 token limit)	E2E	0.7404	15.97	+323.57
Bi-GRU (50 token limit)	E2E	0.7462	17.51	+364.38
Copy-Aug Transformer*	E2E	0.9374	2.58	-31.56
Copy-Aug Transformer**	E2E	0.9409	2.51	-33.42
Ours	2-Stage	0.9453	2.24	-40.66
Ours*	2-Stage	0.9361	2.83	-25.03
Ours**	2-Stage	<b>0.9502</b>	<b>2.07</b>	<b>-45.21</b>
Ours** (with Oracle Detection)	2-Stage	0.9774	1.08	-71.39

\* Model is only trained on noise injected dataset.

\*\* Model is pre-trained on noise injected dataset before fine-tuned on the regular training-set.

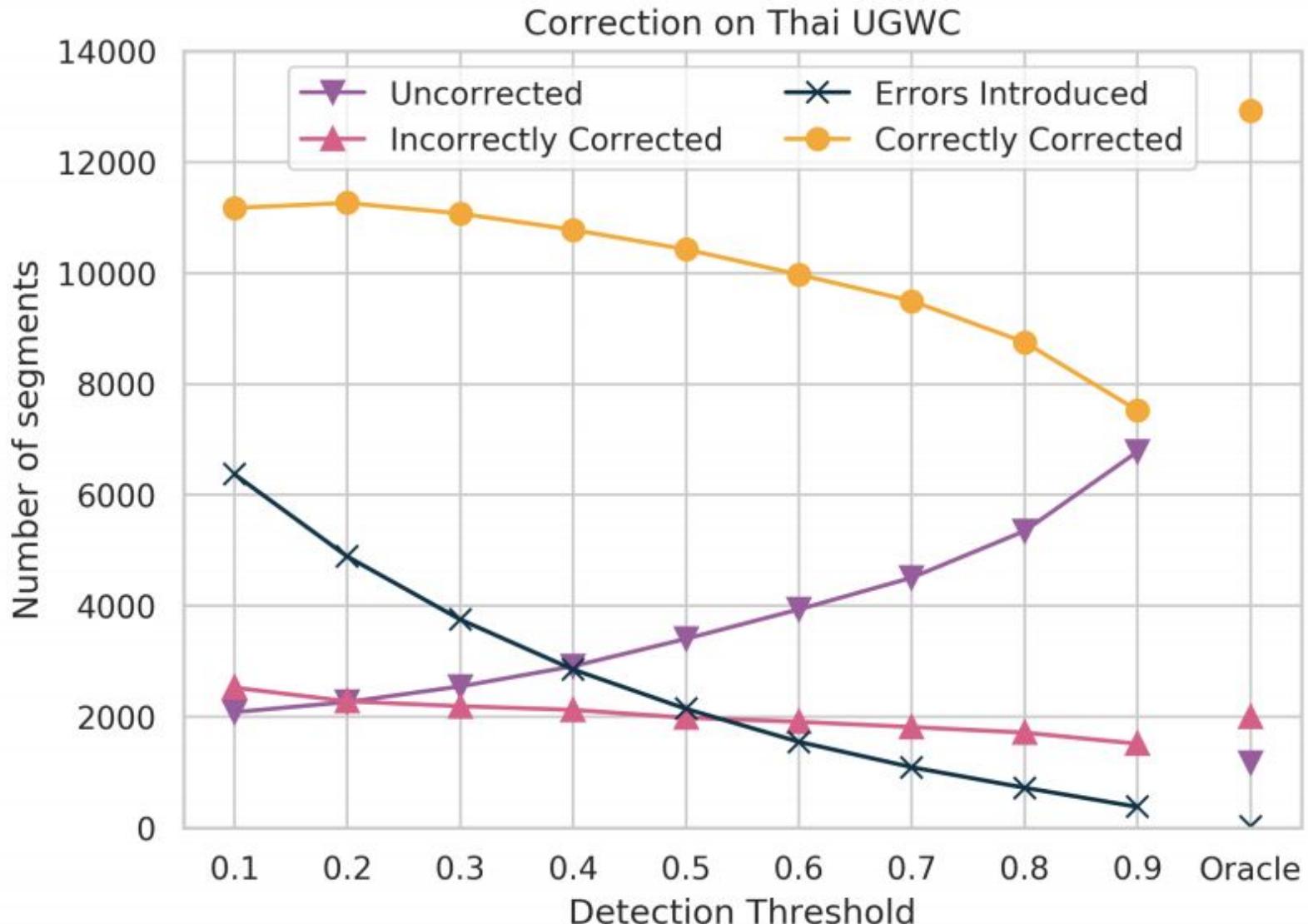


Figure 14: Types of corrections produced from our method on the Thai UGWC test-set

# PARSING

---

PCFG, Recursive Neural Network

Document classification, sentiment analysis, QA,  
conversation agents, summarization, translation

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

Words

Morphology

Characters

Word embeddings

CommunicationEvent(e)

Agent(e, Alice)

Recipient(e, Bob)

SpeakerContext(s)

TemporalBefore(e, s)

Parsing

NP

PoS

Noun

Tagging

VerbPast

VP

PP

Prep

Noun

Punct

Alice talked to Bob.

Language modeling

↑ Tokenization

talk -ed

[VerbPast]

Alice talked to Bob.

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

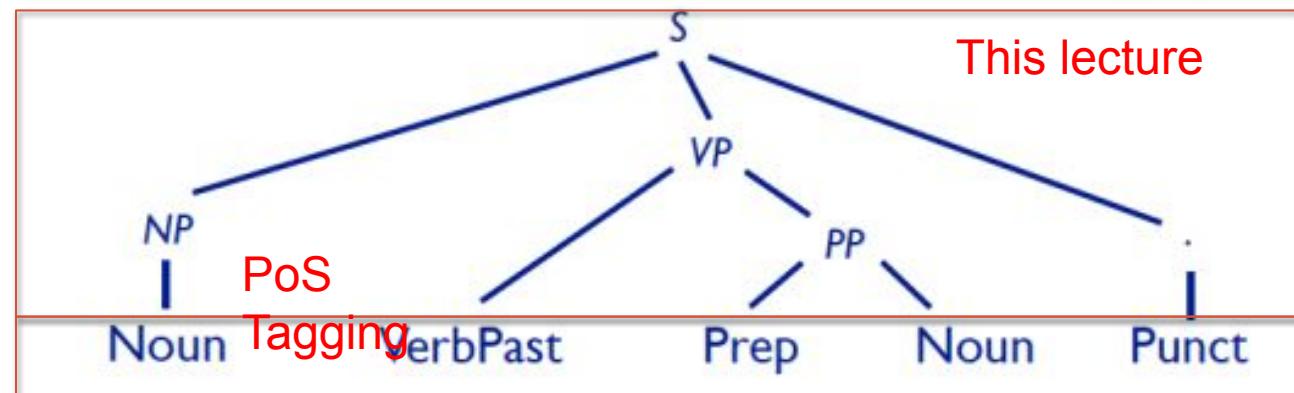
Words

Morphology

Characters

Word embeddings

CommunicationEvent(e)  
Agent(e, Alice)  
Recipient(e, Bob)  
SpeakerContext(s)  
TemporalBefore(e, s)



Alice talked to Bob.

↑ Tokenization

talk -ed

[VerbPast]

Alice talked to Bob.

Use Syntax to create sentence level meanings!

Discourse

Semantics

Syntax: Constituents

Syntax: Part of Speech

Words

Morphology

Characters

Word embeddings

CommunicationEvent(e)  
Agent(e, Alice)  
Recipient(e, Bob)

SpeakerContext(s)  
TemporalBefore(e, s)

Parsing



Alice talked to Bob.

Language modeling

talk -ed

[VerbPast]

Alice talked to Bob.

Tokenization

# Semantic embeddings of several words

- Compositionality
- We know how to create a dense vector representation for a word
  - What about larger linguistic units? (e.g. phrase, sentence )
- We can combine smaller units into a larger unit

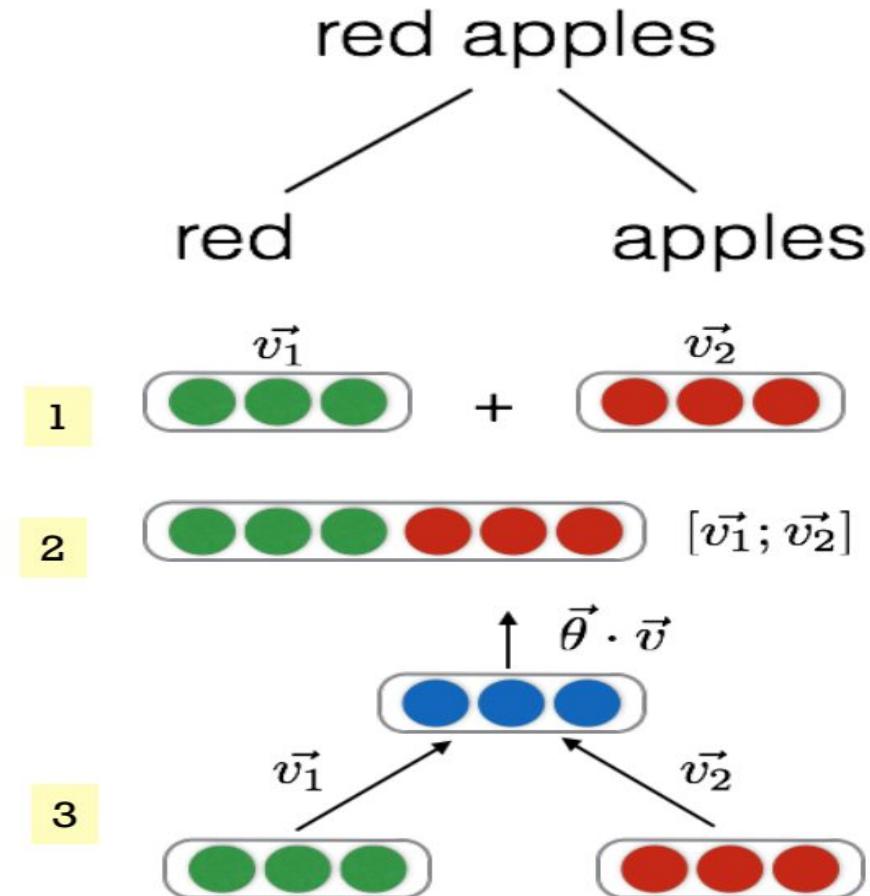
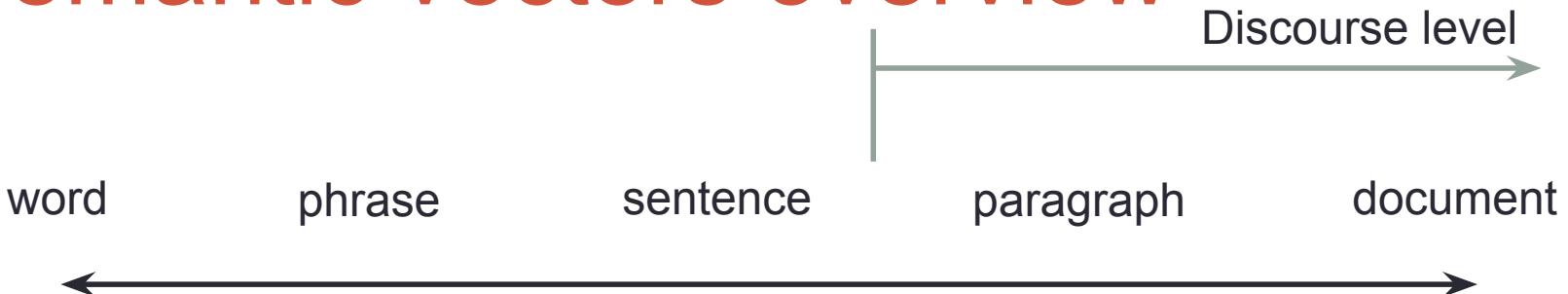
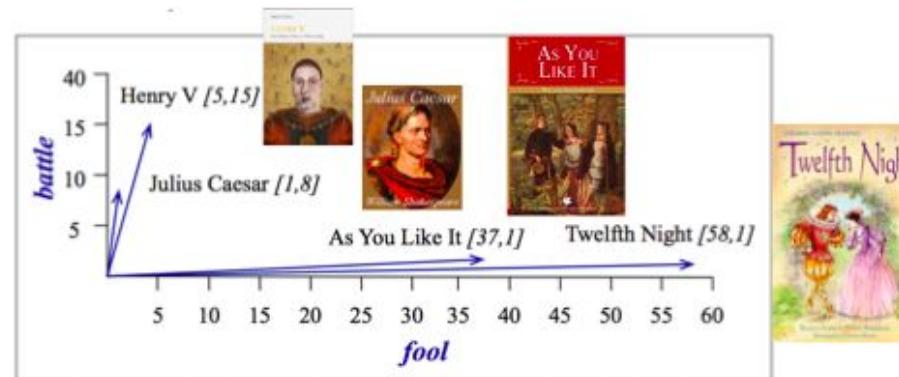


Image ref: Prof. Regina Barzilay , NLP@MIT

# Semantic vectors overview



- Word vectors
  - Co-occurrence
  - PPMI
  - TFIDF
  - Word2vec
    - CBoW
    - Skip-gram



**Figure 15.3** A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

- Doc vectors
    - Term-document
      - Bag of words model
- Recurrent networks (HW3 ToDo#6)

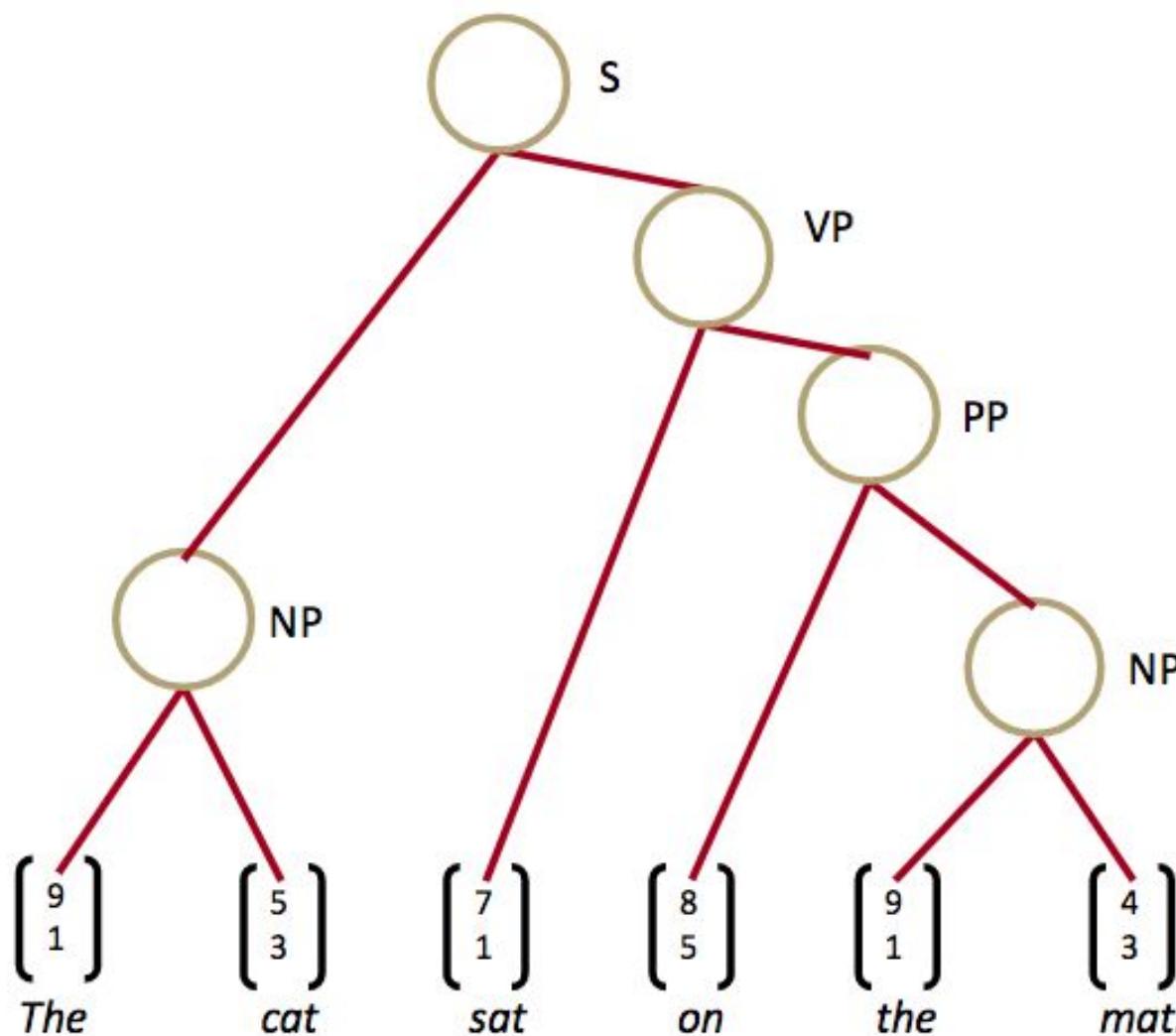
# Semantic vectors overview



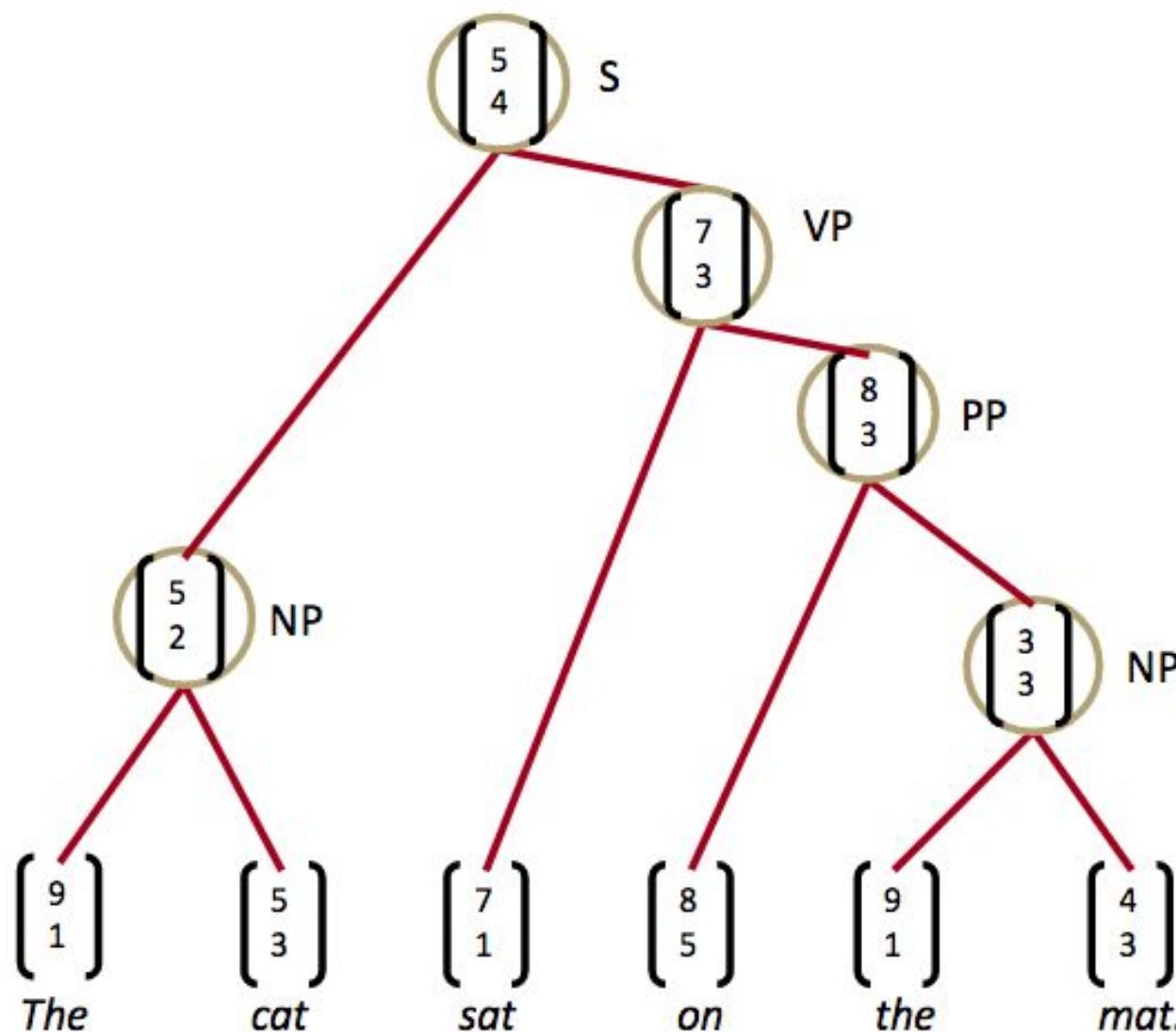
How do we represent things in this level? Without ignoring word order (bag of words)

- Word vectors
  - Co-occurrence
  - PPMI
  - TFIDF
  - Word2vec
    - CBoW
    - Skip-gram
- Doc vectors
  - Term-document
    - Bag of words model
  - Doc2Vec
  - LDA2Vec
- Recurrent networks

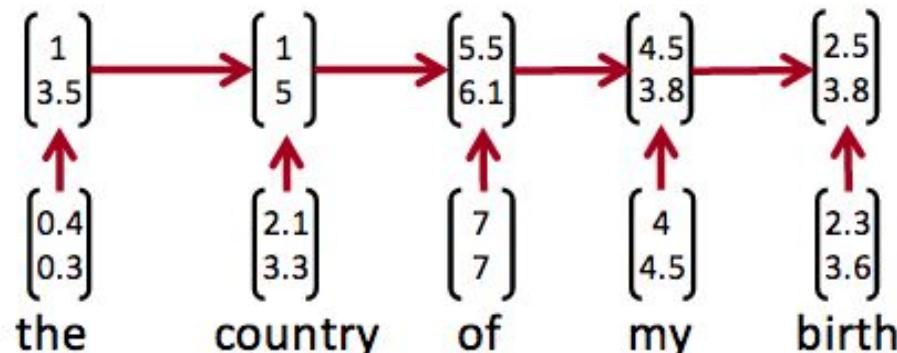
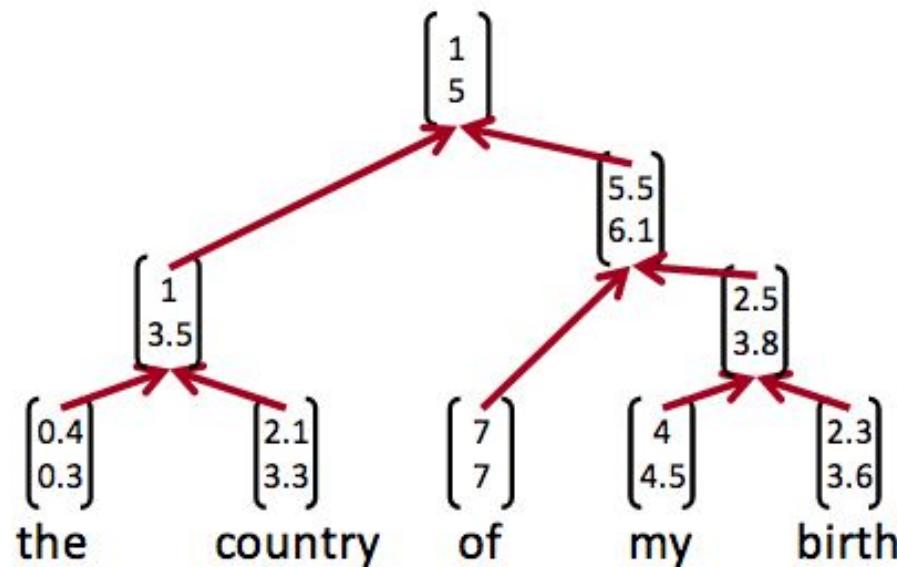
# Parsing and representation big picture



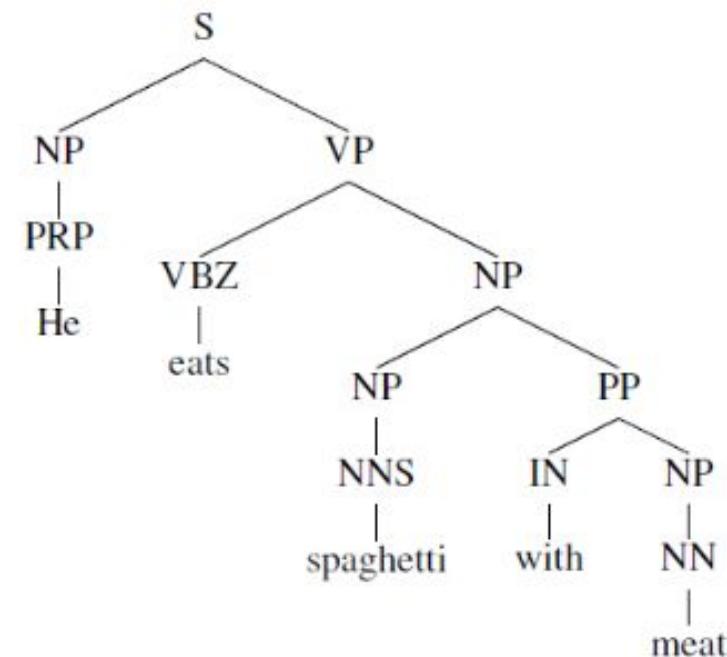
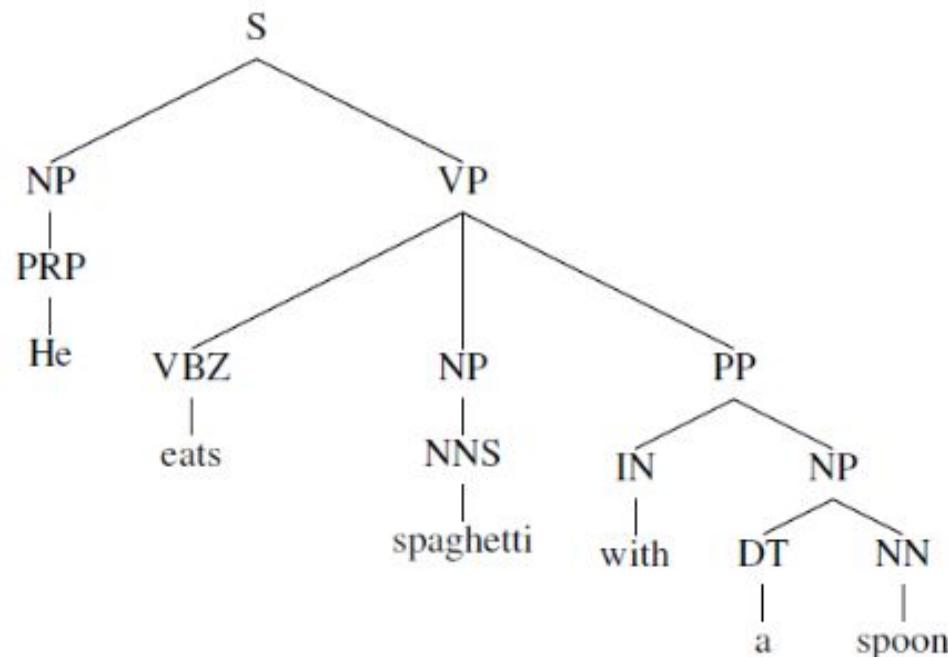
# Parsing and representation big picture



# Recursive vs Recurrent representation



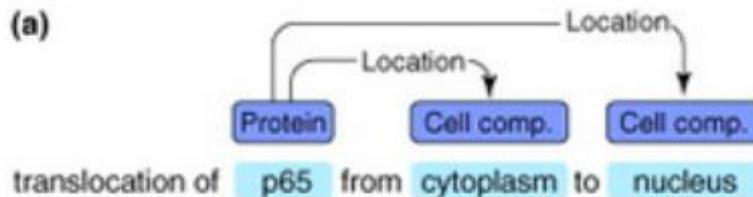
# Cases where recursive might be better?



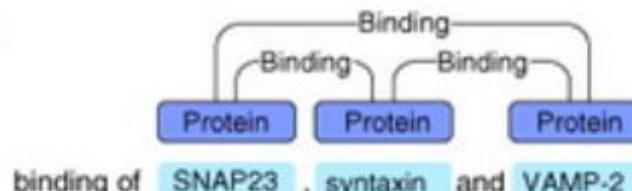
But recurrent structures might be able to learn this too...

# Application of parse trees: info extraction

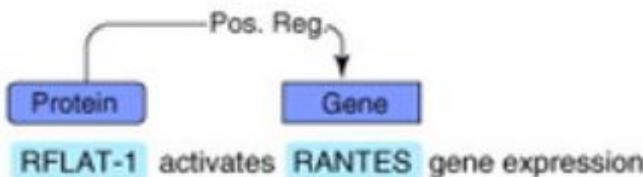
(a)



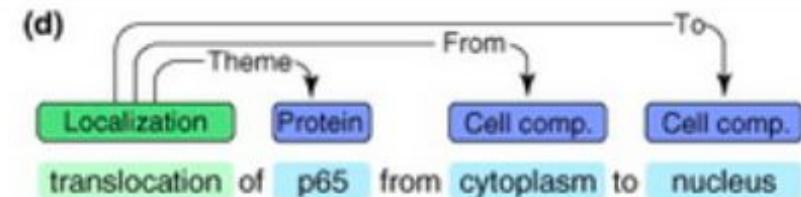
(b)



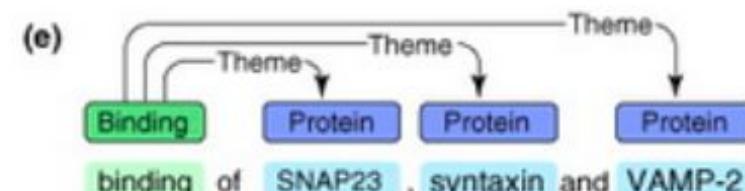
(c)



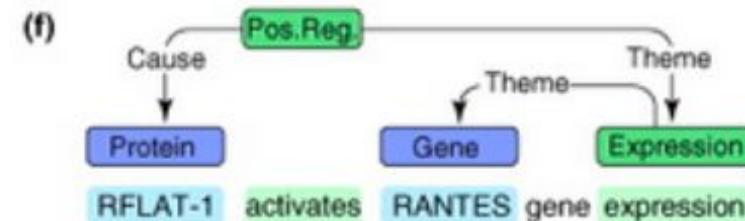
(d)



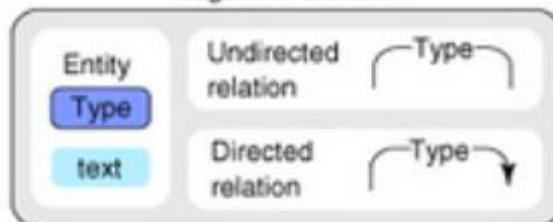
(e)



(f)



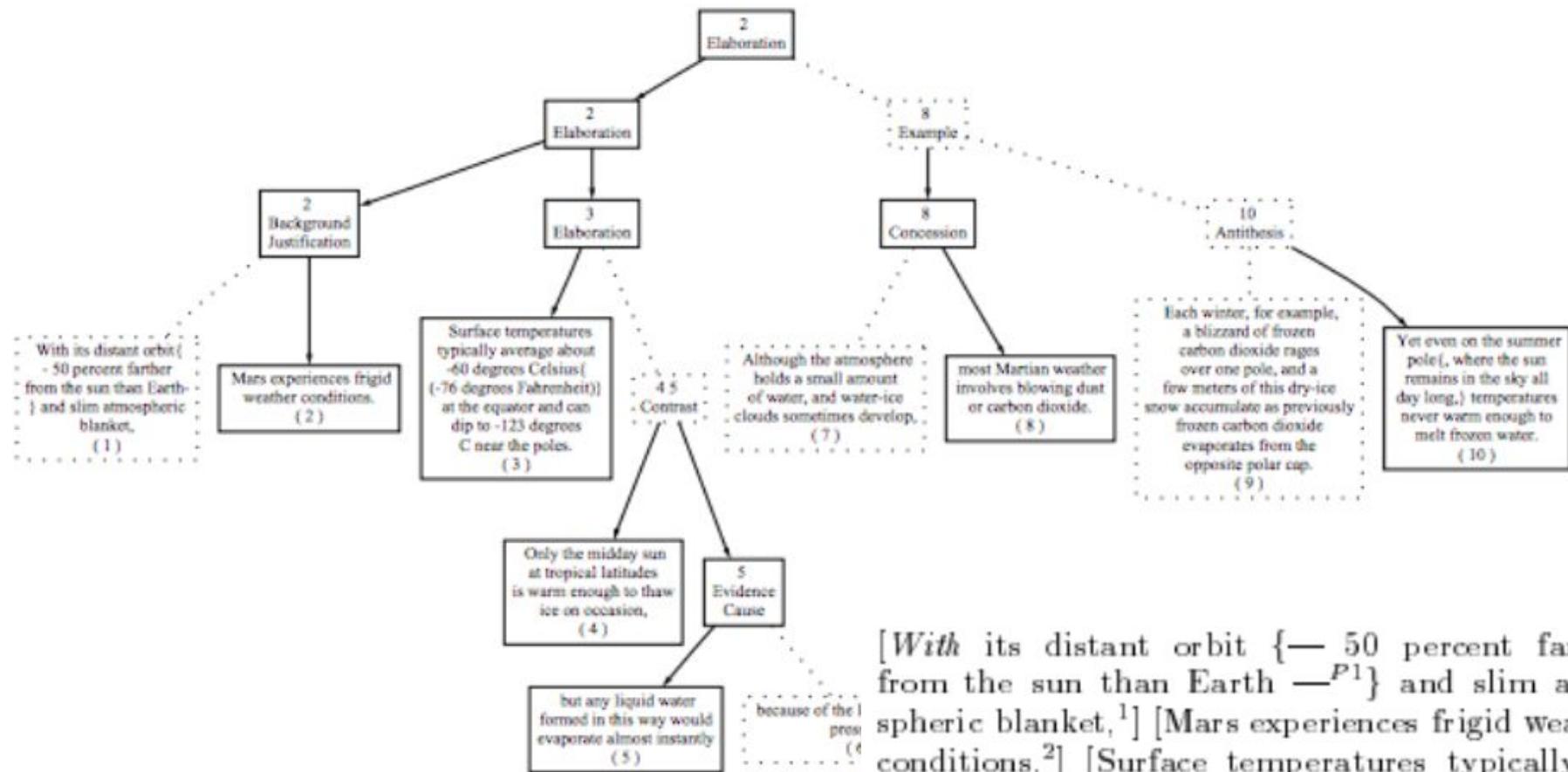
Legend : relations



Legend : events



# A side note: discourse parsing



[With its distant orbit {— 50 percent farther from the sun than Earth —<sup>P1</sup>} and slim atmospheric blanket,<sup>1</sup>] [Mars experiences frigid weather conditions.<sup>2</sup>] [Surface temperatures typically average about -60 degrees Celsius (-76 degrees Fahrenheit) at the equator and can dip to -123 degrees C near the poles.<sup>3</sup>] [Only the midday sun at tropical latitudes is warm enough to thaw ice

# Overview

- Types of grammars
  - Context Free Grammar
  - Probabilistic Context Free Grammar
    - CYK parser
  - Dependency Grammar
    - Transition-based parsing
    - Recursive neural networks

# Constituents

- Groups of words behaving as a single units

- Ex: Noun phrase

Harry the Horse

The reason he comes into the house

They

A high-class spot such as Mindy's

- Checking for constituents

- See if they can appear in similar syntactic environments

They sit...

The reason he comes into the house is...

# Context-Free Grammar (CFG)

- A grammar specifies what kind of parse tree can be generated.
- CFG or Phrase-Structure Grammars assumes the grammar is context-free
  - Most forms of natural language are context-free
    - Thai and English are typically CFG languages
    - Used in many programming languages
- CFG is based on constituents structures

# A context-free grammar example

A CFG is defined by  
 $G = (N, S, \Sigma, R)$

- $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$  Nonterminals
- $S = S$  Starting symbol
- $\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$

$S$	$\Rightarrow$	$NP$	$VP$
$VP$	$\Rightarrow$	$Vi$	
$VP$	$\Rightarrow$	$Vt$	$NP$
$VP$	$\Rightarrow$	$VP$	$PP$
$NP$	$\Rightarrow$	$DT$	$NN$
$NP$	$\Rightarrow$	$NP$	$PP$
$PP$	$\Rightarrow$	$IN$	$NP$

$Vi$	$\Rightarrow$	sleeps
$Vt$	$\Rightarrow$	saw
$NN$	$\Rightarrow$	man
$NN$	$\Rightarrow$	woman
$NN$	$\Rightarrow$	telescope
$DT$	$\Rightarrow$	the
$IN$	$\Rightarrow$	with
$IN$	$\Rightarrow$	in

Terminals

Production rules  
 $X \rightarrow Y_1 \dots Y_n$   
 $Y_i$  can be terminal or nonterminal  
The rule only relies on  $X$  (no context)  
(vs context-sensitive)

Note:  $S$ =sentence,  $VP$ =verb phrase,  $NP$ =noun phrase,  $PP$ =prepositional phrase,  $DT$ =determiner,  $Vi$ =intransitive verb,  $Vt$ =transitive verb,  $NN$ =noun,  $IN$ =preposition

# Generation using left-most derivation strategy

- Recursive strategy

```
left_most_derivation(A):
```

```
    if terminal(A):  
        return A
```

Return if terminal

```
    rule = choose(rules(A))  
    rhs = right_hand_side(rule)  
    return concatenate([left_most_derivation(A') for A' in rhs])
```

$R =$	$S \Rightarrow NP VP$
	$VP \Rightarrow Vi$
	$VP \Rightarrow Vt NP$
	$VP \Rightarrow VP PP$
	$NP \Rightarrow DT NN$
	$NP \Rightarrow NP PP$
	$PP \Rightarrow IN NP$

$Vi \Rightarrow sleeps$
$Vt \Rightarrow saw$
$NN \Rightarrow man$
$NN \Rightarrow woman$
$NN \Rightarrow telescope$
$DT \Rightarrow the$
$IN \Rightarrow with$
$IN \Rightarrow in$

- To generate we call `left_most_derivation(S)` (top down)
- A string belongs to the language of a CFG if there exist a sequence of left-most derivation that can generate the string

$$L = \{s \in \Sigma^* | s = \text{left\_most\_derivation}(S)\}$$

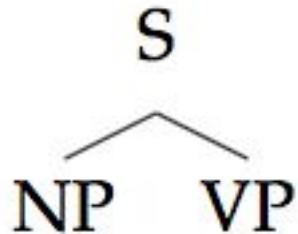
# Example

- The woman saw the man with the telescope

$$R = \begin{array}{l|ll} S & \Rightarrow & NP \quad VP \\ \hline VP & \Rightarrow & Vi \\ VP & \Rightarrow & Vt \quad NP \\ VP & \Rightarrow & VP \quad PP \\ \hline NP & \Rightarrow & DT \quad NN \\ NP & \Rightarrow & NP \quad PP \\ \hline PP & \Rightarrow & IN \quad NP \end{array} \quad \begin{array}{l|l} Vi & \Rightarrow \text{sleeps} \\ Vt & \Rightarrow \text{saw} \\ \hline NN & \Rightarrow \text{man} \\ NN & \Rightarrow \text{woman} \\ NN & \Rightarrow \text{telescope} \\ \hline DT & \Rightarrow \text{the} \\ IN & \Rightarrow \text{with} \\ IN & \Rightarrow \text{in} \end{array}$$

# Example

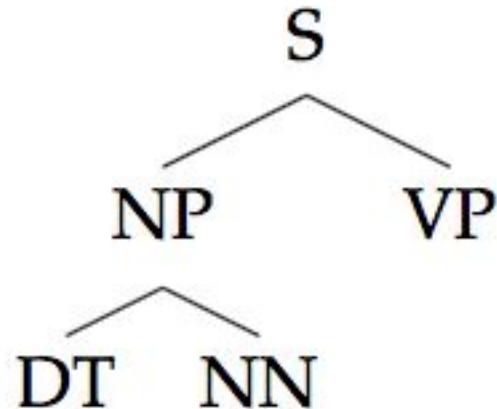
- The woman saw the man with the telescope


$$R = \begin{array}{l|ll} \text{S} & \Rightarrow & \text{NP} \quad \text{VP} \\ \hline \text{VP} & \Rightarrow & \text{Vi} \\ \text{VP} & \Rightarrow & \text{Vt} \quad \text{NP} \\ \text{VP} & \Rightarrow & \text{VP} \quad \text{PP} \\ \hline \text{NP} & \Rightarrow & \text{DT} \quad \text{NN} \\ \text{NP} & \Rightarrow & \text{NP} \quad \text{PP} \\ \hline \text{PP} & \Rightarrow & \text{IN} \quad \text{NP} \end{array}$$

Vi	$\Rightarrow$	sleeps
Vt	$\Rightarrow$	saw
NN	$\Rightarrow$	man
NN	$\Rightarrow$	woman
NN	$\Rightarrow$	telescope
DT	$\Rightarrow$	the
IN	$\Rightarrow$	with
IN	$\Rightarrow$	in

# Example

- The woman saw the man with the telescope

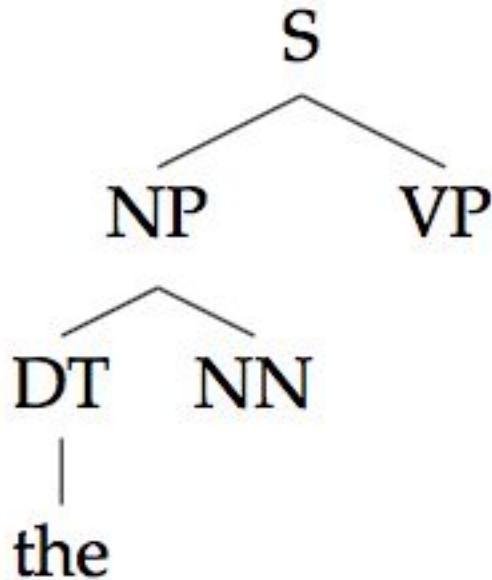


$R =$	S	$\Rightarrow$	NP	VP
	VP	$\Rightarrow$	Vi	
	VP	$\Rightarrow$	Vt	NP
	VP	$\Rightarrow$	VP	PP
	NP	$\Rightarrow$	DT	NN
	NP	$\Rightarrow$	NP	PP
	PP	$\Rightarrow$	IN	NP

Vi	⇒	sleeps
Vt	⇒	saw
NN	⇒	man
NN	⇒	woman
NN	⇒	telescope
DT	⇒	the
IN	⇒	with
IN	⇒	in

# Example

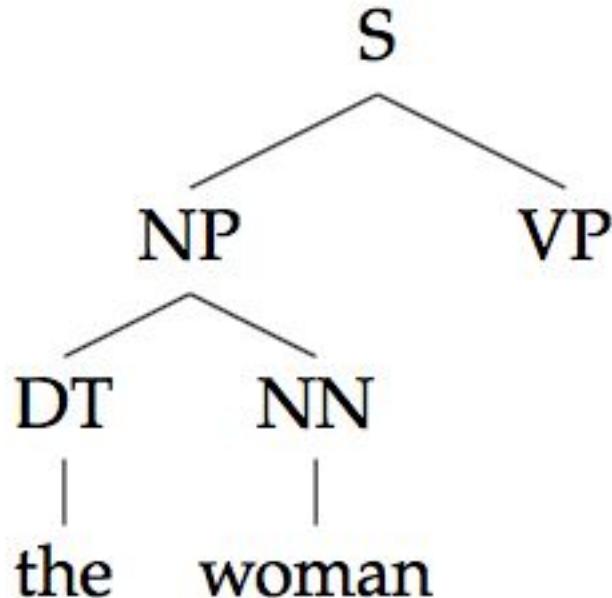
- The woman saw the man with the telescope



$R =$	$S \Rightarrow NP \quad VP$	$Vi \Rightarrow sleeps$
	$VP \Rightarrow Vi$	$Vt \Rightarrow saw$
	$VP \Rightarrow Vt \quad NP$	$NN \Rightarrow man$
	$VP \Rightarrow VP \quad PP$	$NN \Rightarrow woman$
	$NP \Rightarrow DT \quad NN$	$NN \Rightarrow telescope$
	$NP \Rightarrow NP \quad PP$	$DT \Rightarrow the$
	$PP \Rightarrow IN \quad NP$	$IN \Rightarrow with$
		$IN \Rightarrow in$

# Example

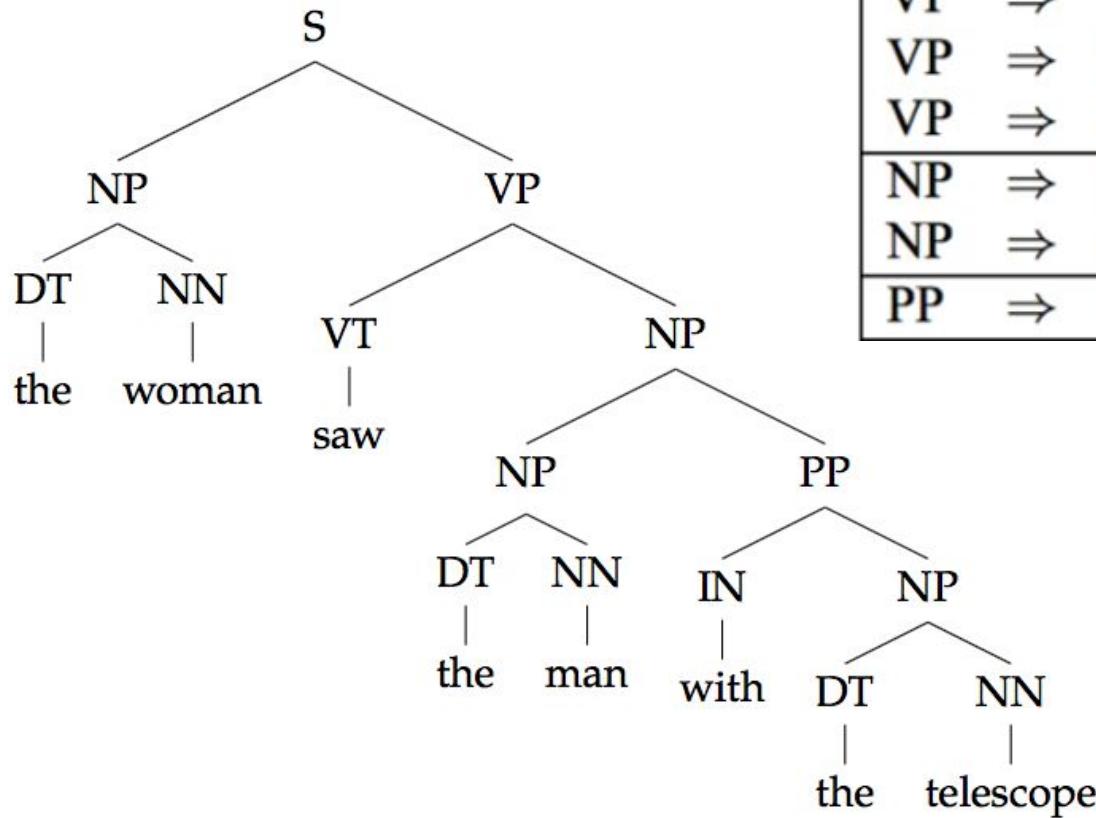
- The woman saw the man with the telescope



$R =$	$S \Rightarrow NP \quad VP$	$Vi \Rightarrow \text{sleeps}$
	$VP \Rightarrow Vi$	$Vt \Rightarrow \text{saw}$
	$VP \Rightarrow Vt \quad NP$	$NN \Rightarrow \text{man}$
	$VP \Rightarrow VP \quad PP$	$NN \Rightarrow \text{woman}$
	$NP \Rightarrow DT \quad NN$	$NN \Rightarrow \text{telescope}$
	$NP \Rightarrow NP \quad PP$	$DT \Rightarrow \text{the}$
	$PP \Rightarrow IN \quad NP$	$IN \Rightarrow \text{with}$
		$IN \Rightarrow \text{in}$

# Example

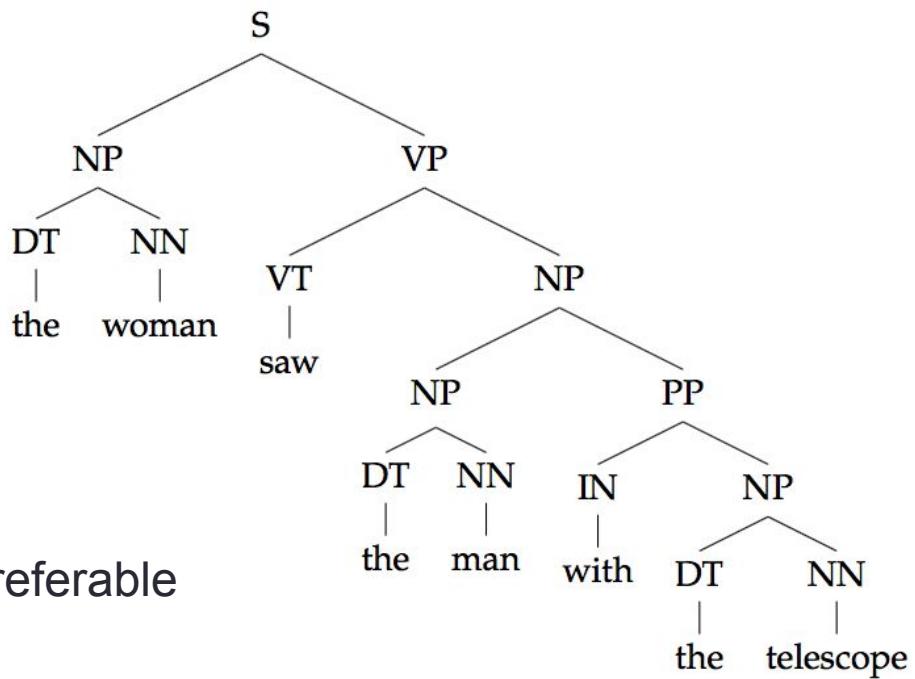
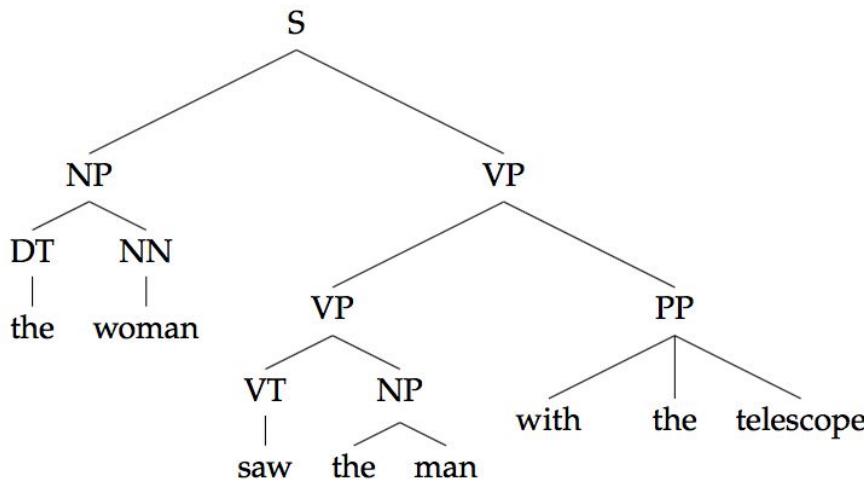
- The woman saw the man with the telescope



$R =$	$S \Rightarrow NP \quad VP$	$Vi \Rightarrow sleeps$
	$VP \Rightarrow Vi$	$Vt \Rightarrow saw$
	$VP \Rightarrow Vt \quad NP$	$NN \Rightarrow man$
	$VP \Rightarrow VP \quad PP$	$NN \Rightarrow woman$
	$NP \Rightarrow DT \quad NN$	$NN \Rightarrow telescope$
	$NP \Rightarrow NP \quad PP$	$DT \Rightarrow the$
	$PP \Rightarrow IN \quad NP$	$IN \Rightarrow with$
		$IN \Rightarrow in$

# Ambiguities

- There can be multiple derivations for the same string
- These sentences are ambiguous as each parse represents a different meaning



Need a way to say which one is more preferable  
Add probabilities!

# Probabilistic Context-Free Grammar

- Production rules now have **probabilities**

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

The probability of a (sentence, parse tree) pair is

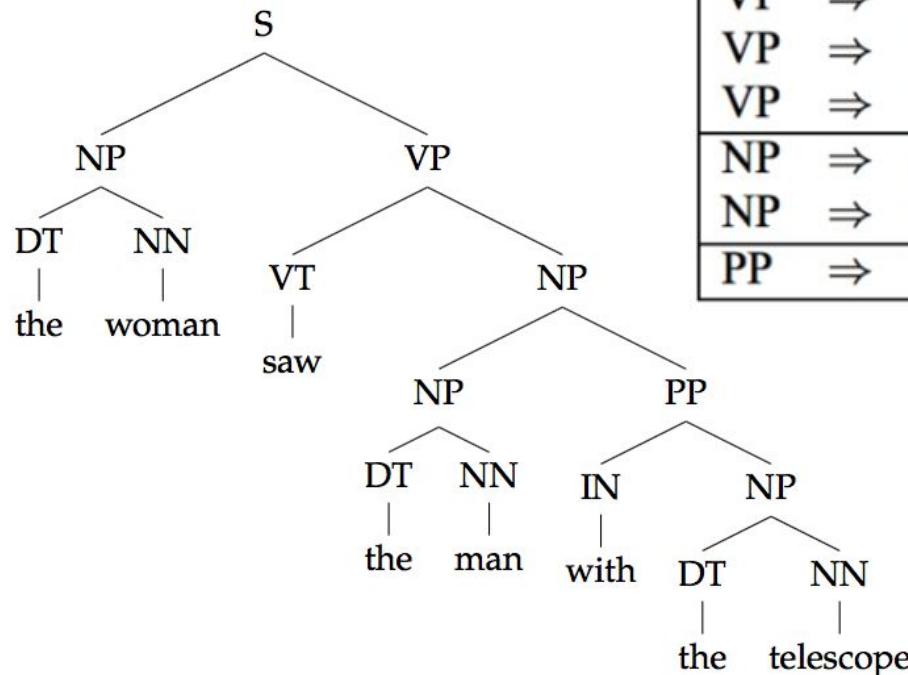
$$p(S, T) = \prod_{i=1}^m p(\alpha_i \rightarrow \beta_i | \alpha_i)$$

m is the number of transitions

$$P(NN \rightarrow \text{man} | NN) = 0.7$$

# P(S, T)

- $P(S, T) = [1.0 * 0.3 * 1.0 * 0.2] * [0.4 * 1.0 * 0.7 * (0.3 * 1.0 * 0.7) * (1.0 * 0.5 * 0.3 * 1.0 * 0.1)]$



S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

# Estimating transition probabilities

- Counts from training set!

$$P(S \rightarrow NP\ VP|S) = \frac{count(S \rightarrow NP\ VP)}{count(S)}$$

# PCFG related questions

- What is the most likely parse? (**parsing task**)
  - $\operatorname{argmax}_T P(T, S)$
- What is the probability of the sentence,  $P(S)$ ? (**Language modeling task**)
  - $P(S) = \sum_T P(T, S)$

How?

Dynamic programming  
(CYK algorithm –  
Cocke–Younger–Kasami algorithm)

# Chomsky Normal Form

- CYK can be used if the CFG is in Chomsky Normal Form
- A CFG is in Chomsky Normal Form if each rule either converts to **two nonterminals** or a **single terminal**.

$$X \rightarrow Y_1 Y_2$$

Uppercase letters mean nonterminal  
Lowercase letters mean terminal

$$X \rightarrow y$$

- Any CFG can be converted to CNF

- For example NP  $\rightarrow$  DT, ADJ, NN
  - Turns into two rules
  - NP  $\rightarrow$  DT, ADJP
  - ADJP  $\rightarrow$  ADJ, NN

•  $N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$  Nonterminals

•  $S = S$  Starting symbol

•  $\Sigma = \{\text{sleeps, saw, man, woman, telescope, the, with, in}\}$  Terminals

$R =$	$S \Rightarrow NP \quad VP$	$Vi \Rightarrow \text{sleeps}$
	$VP \Rightarrow Vi$	$Vt \Rightarrow \text{saw}$
	$VP \Rightarrow Vt \quad NP$	$NN \Rightarrow \text{man}$
	$VP \Rightarrow VP \quad PP$	$NN \Rightarrow \text{woman}$
	$NP \Rightarrow DT \quad NN$	$NN \Rightarrow \text{telescope}$
	$NP \Rightarrow NP \quad PP$	$DT \Rightarrow \text{the}$
	$PP \Rightarrow IN \quad NP$	$IN \Rightarrow \text{with}$
		$IN \Rightarrow \text{in}$

Production rules

$X \rightarrow Y_1 \dots Y_n$

$Y_i$  can be terminal or nonterminal

The rule only relies on

$X$  (no context)

(vs context-sensitive)

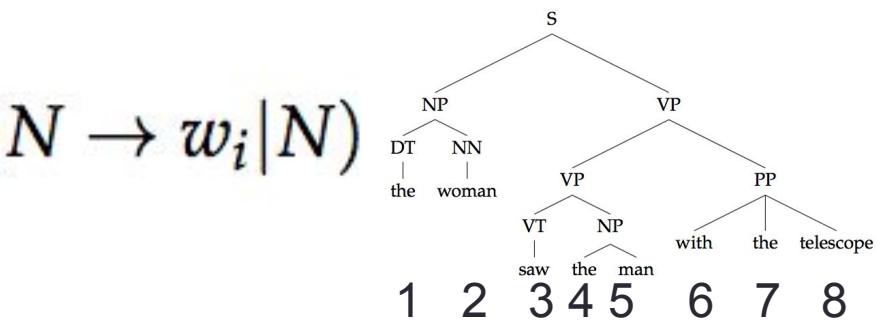
# CYK algorithm for parsing

3 dimensional

- $\pi(i, j, N)$  probability that words i to j can be generated by nonterminal N

- Base case:  $\pi(i, i, N) = P(N \rightarrow w_i | N)$

- Inductive case:



$$\pi(i, j, N) = \max_{k, P, Q} Pr(N \rightarrow P \mid Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

where  $k \in \{i, \dots, j - 1\}$ ,  $P \in \mathcal{N}$ , and  $Q \in \mathcal{N}$ .

Saves the rule that gave max probability to backtrack

# CYK algorithm for language modeling

- $\pi(i, j, N)$  probability that words i to j can be generated by nonterminal N

- Base case:  $\pi(i, i, N) = P(N \rightarrow w_i | N)$

- Inductive case:  
$$\pi(i, j, N) = \max_{k, P, Q} Pr(N \rightarrow P \ Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$
$$\text{argmax}_T P(T, S) \quad \text{vs} \quad P(S) = \sum_T P(T, S)$$

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P \ Q | N) \cdot \pi(i, \underline{k}, P) \cdot \pi(\underline{k + 1, j}, Q)$$

where  $k \in \{i, \dots, j - 1\}$ ,  $P \in \mathcal{N}$ , and  $Q \in \mathcal{N}$ .

# Language modeling example CYK with PCFG

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$\pi(i, j, N)$

A

a			
b			
c			

B

a			
b			
c			

# Base case

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, i, N) = P(N \rightarrow w_i | N)$$

A

a	0.2		
b		0	
c			0

B

a	0		
b		0.1	
c			0.2

# 1<sup>st</sup> step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{\Lambda\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

B

a	0.2		
b		0	
c			0

Only consider  $i, j$  where  $i < j$

a	0		
b		0.1	
c			0.2

# 1<sup>st</sup> step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

a	0.2	0.016	
b		0	
c		0	

$$\begin{aligned}
 \pi[1, 2, A] &= P(A \rightarrow AA) \cdot P(A \rightarrow a | A) \cdot P(A \rightarrow b | A) \\
 &\quad + P(A \rightarrow AB) \cdot P(A \rightarrow a | A) \cdot P(B \rightarrow b | B) \\
 &\quad + P(A \rightarrow BA) \cdot P(B \rightarrow a | B) \cdot P(A \rightarrow b | A) \\
 &\quad + P(A \rightarrow BB) \cdot P(B \rightarrow a | B) \cdot P(B \rightarrow b | B) \\
 &= 0 + P(A \rightarrow AB) \cdot \pi[1, 1, A] \cdot \pi[2, 2, B] + 0 + 0 \\
 &= 0.8 \cdot 0.2 \cdot 0.1 = 0.016
 \end{aligned}$$

# 1<sup>st</sup> step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

$$\begin{aligned}\pi(1, 2, B) &= P(B \rightarrow BB)P(B \rightarrow a | B)P(B \rightarrow b | B) \\ &= P(B \rightarrow BB)\pi(1, 1, B)\pi(2, 2, B) \\ &= 0.7 * 0 * 0.1 = 0\end{aligned}$$

B

a	0	0	
b		0.1	
c			0.2

## 2<sup>nd</sup> step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

B

a	0.2	0.016	
b		0	0
c		0	

a	0	0	
b		0.1	0.014
c			0.2

$$\pi(2,3,A) = P(A \rightarrow A B) \pi(2,2,A) \pi(3,3,B) = 0.8 * 0 * 0.2 = 0$$

## 2<sup>nd</sup> step

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

B

a	0.2	0.016	
b		0	0
c			0

a	0	0	
b		0.1	0.014
c			0.2

$$\pi(2,3,B) = P(B \rightarrow B | B) \pi(2,2,B) \pi(3,3,B) + 0 = 0.7 * 0.1 * 0.2 = 0.014$$

# Finish

- $N = \{A, B\}$
- $\Sigma = \{a, b, c\}$
- $S = \{A\}$

Find  $P(\text{"abc"})$

$A \rightarrow AB$	0.8
$A \rightarrow a$	0.2
$B \rightarrow BB$	0.7
$B \rightarrow b$	0.1
$B \rightarrow c$	0.2

$$\pi(i, j, N) = \sum_{k, P, Q} P(N \rightarrow P | Q | N) \cdot \pi(i, k, P) \cdot \pi(k + 1, j, Q)$$

A

B

a	0.2	0.016	0.0048
b		0	0
c		0	0

a	0	0	0
b		0.1	0.014
c			0.2

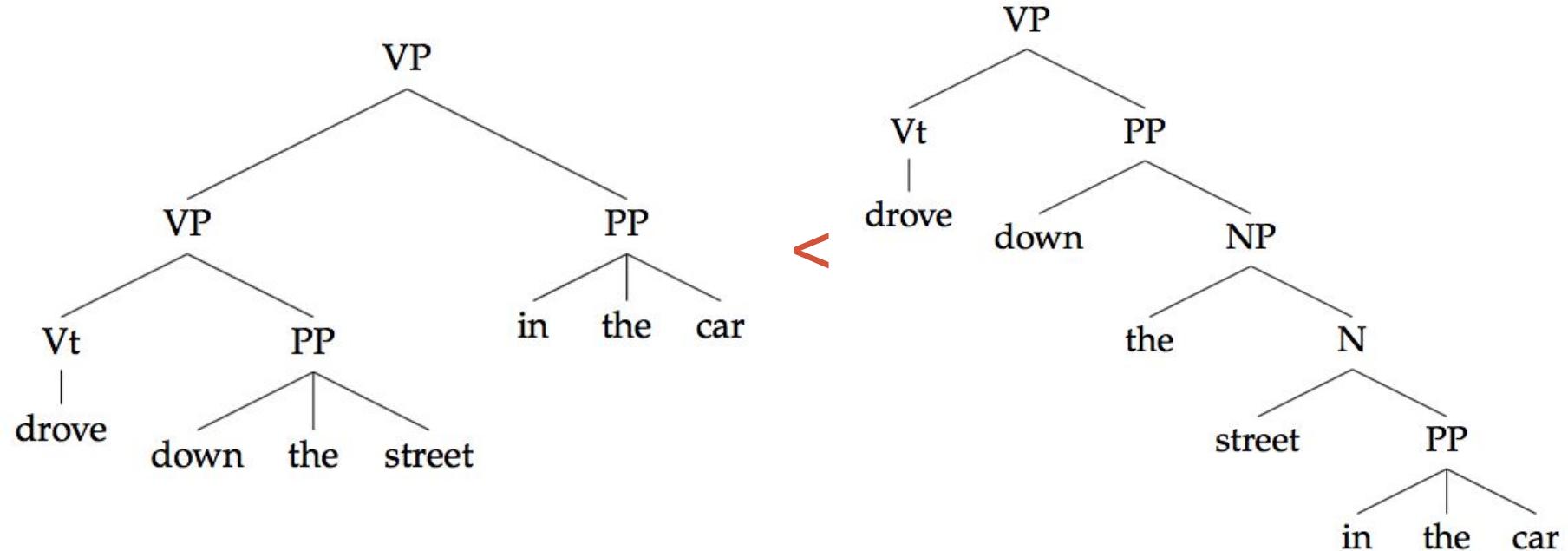
$$\begin{aligned} \pi(1, 3, A) &= P(A \rightarrow AB) \pi(1, 2, A) \pi(3, 3, B) + P(A \rightarrow a) \pi(1, 1, A) \pi(2, 3, B) \\ &= 0.8 * 0.016 * 0.2 + 0.8 * 0.2 * 0.014 = 0.0048 \end{aligned}$$

# PCFG weakness

- Lack of sensitivity to lexical info (does not consider semantics)
- Lack of sensitivity to structural frequency

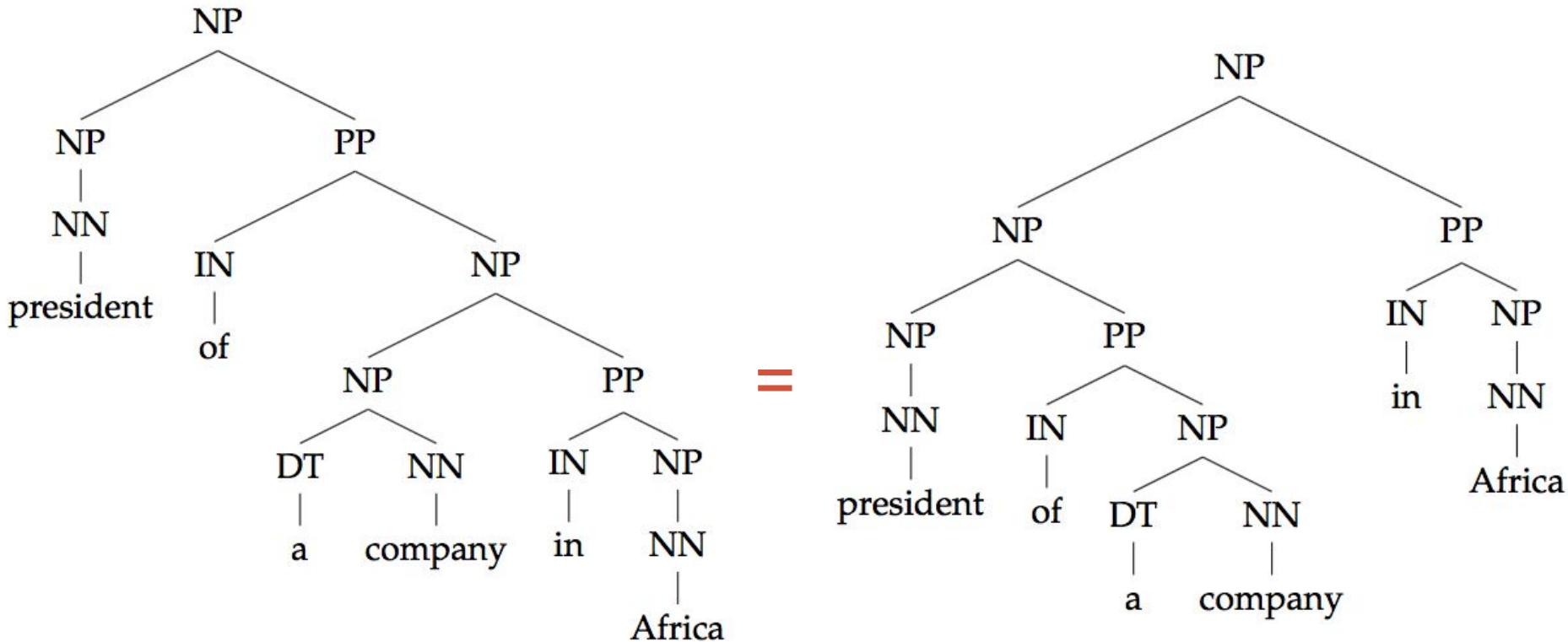
# Lack of lexical info

The probabilities only see terminals and expansions



The street in the car

# Lack of sensitivity to structural frequency



Both trees have same expansion and so same probability

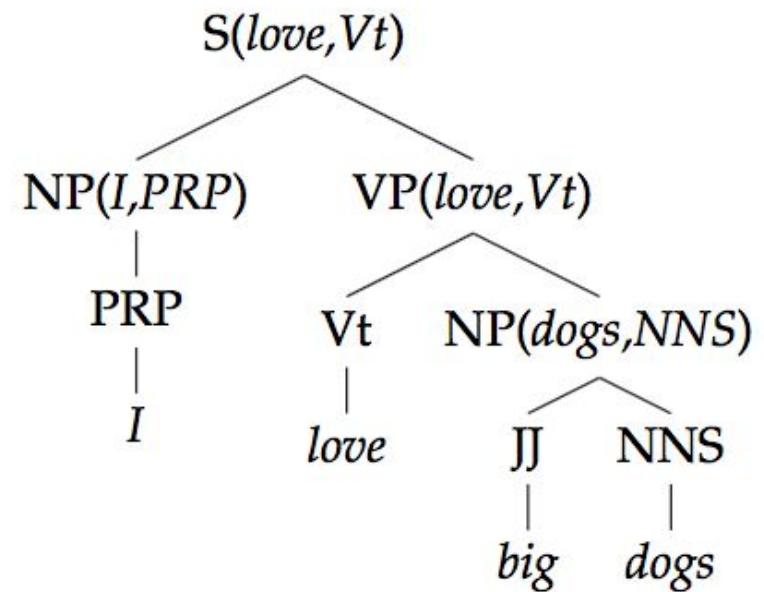
Turns out, left hand side structure appears more frequently in the training set

# Methods to improve PCFG

- Add lexical information to the tree
  - Label each node with associate words
    - Lexicalized trees
    - Latent variable grammars
- Look at bigger portions of the tree at a time

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

Vi	⇒	sleeps	1.0
Vt	⇒	saw	1.0
NN	⇒	man	0.7
NN	⇒	woman	0.2
NN	⇒	telescope	0.1
DT	⇒	the	1.0
IN	⇒	with	0.5
IN	⇒	in	0.5



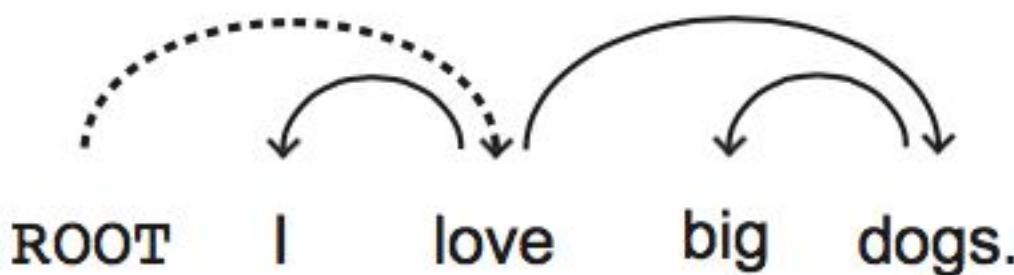
Lexicalized tree

# Dependency grammar

- CFG is based on constituency relation
- In dependency grammar the structure is composed of lexical items (words) linked by edges to form a tree
- Assumptions
  - Each words in a sentence is related or modifies another word
  - All words have a direct or indirect relation to the main verb

# Example

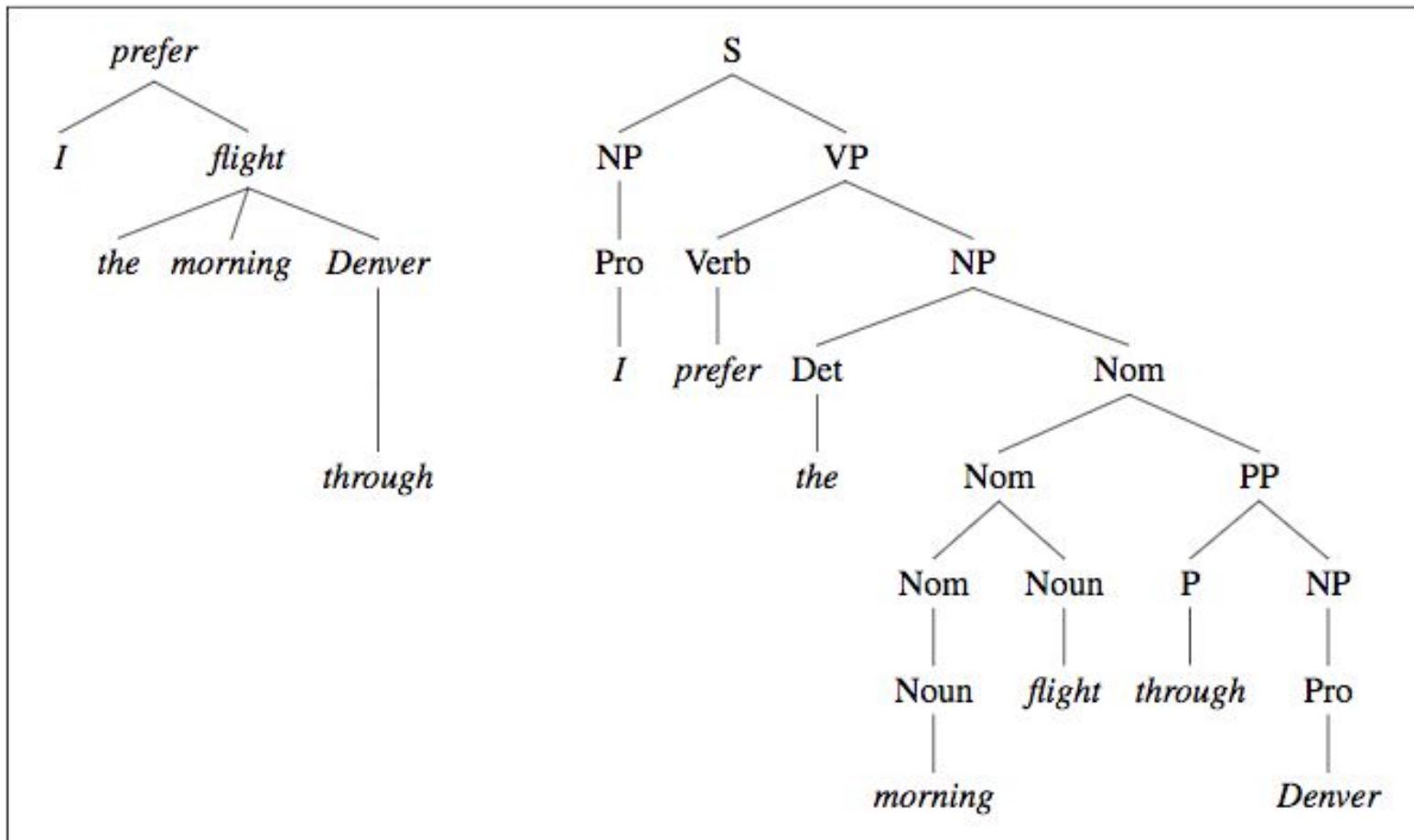
- Add ROOT node as the root of the tree
- The main verb always point to ROOT



A -> B means  
A governs B or  
B depends on A  
A is the head of B

- Each arc can have a category for the relationship.
- Each word can have a PoS label

# Constituency structures vs dependency structures



**Figure 14.1** A dependency-style parse alongside the corresponding constituent-based analysis for *I prefer the morning flight through Denver.*

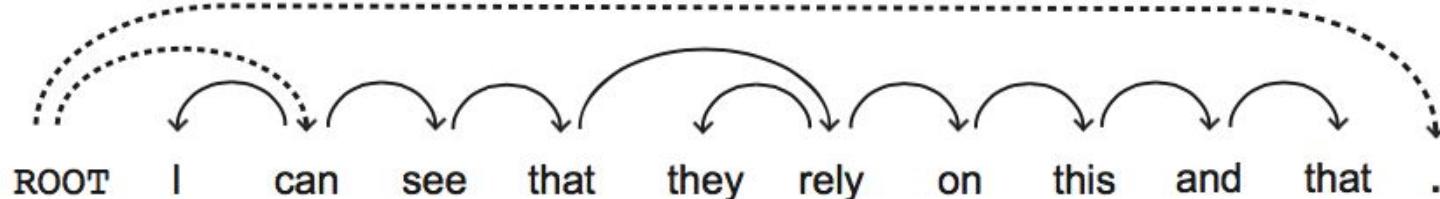
# Constituency structures vs dependency structures

- Constituency structures use more nodes to represent sentences at different levels.
- Constituency structures explicitly label non-terminal nodes (NP vs VP)
- Constituency structures encode more info than dependency structures
- You can convert constituency structures to dependency structures
  - Dependency parsers trained on this is usually better than dependency parsers trained on original dependency structures

# Criteria for heads (basics)

- Head, H. Dependent D
- D modifies H
  - Big (D) dogs (H), willow (D) tree (H)
- H can often replace D
  - I love big (D) dogs (H) -> I love dogs
- H is obligatory while D sometimes is optional
- H determines whether D is obligatory
  - Sarah sneezed (H) vs George kicks (H) the chair (D)
- More criterias! Mostly depends on corpus

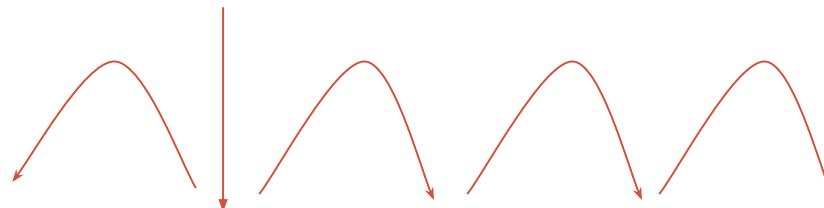
Ex



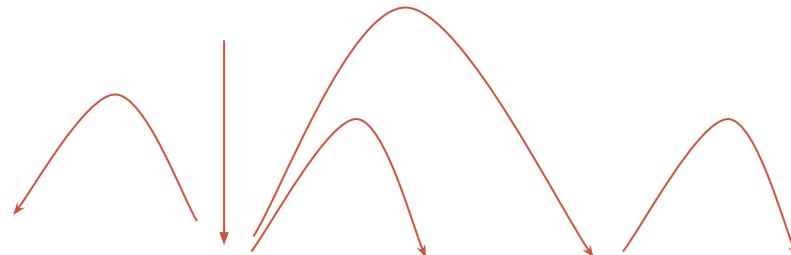
# Dependency and meaning

- Scientists study whales from space
- Scientists study whales from space

# Dependency and meaning



- Scientists study whales from space



- Scientists study whales from space

# Sample Thai sentences

- วัน หนึ่ง เกิด เหตุ ที่ ไม่ คาดคิด ขึ้น มา
- การ ที่ เรา จะ รู้ ถึง สิ่ง เหล่านั้น ได้ ก็ ต้อง อาศัย ปัจจัย หลาย ๆ อย่าง ประกอบ เข้า ด้วยกัน

Note these criteria depends on the corpus convention

# Sample Thai sentences

- 
- วัน หนึ่ง เกิด เหตุ ที่ ไม่ คาดคิด ขึ้น มา

Note these criteria depends on the corpus convention

# Sample Thai sentences

• การ ที่ เรายจะ รู้สึก สิ่ง เหล่านั้น ได้ ก็ ต้อง อาศัย

ปัจจัย หลาย ๆ อย่าง ประกอบ เข้า ด้วยกัน

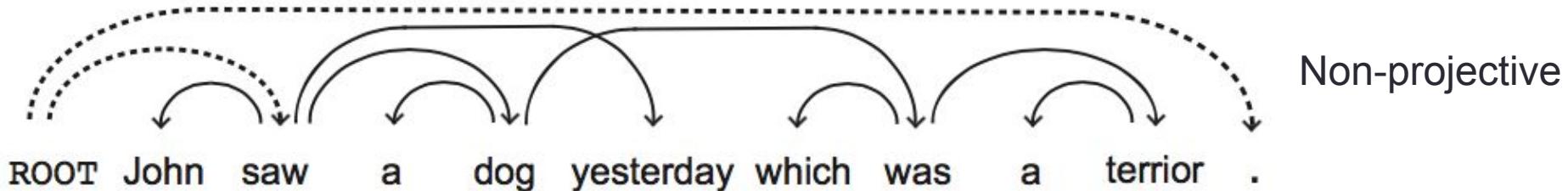
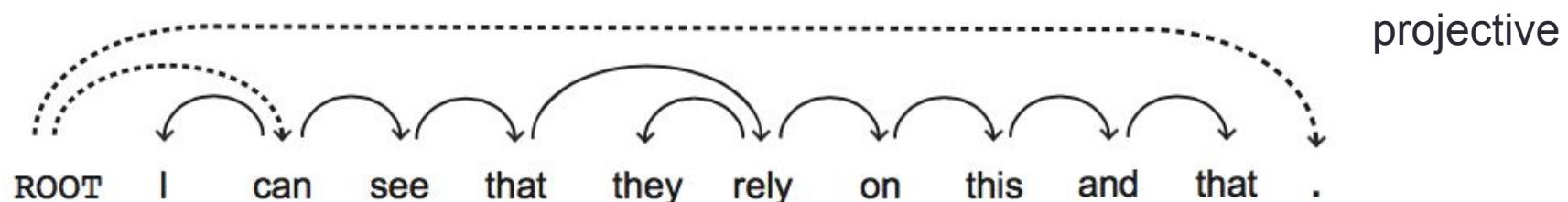
Note these criteria depends on the corpus convention

# Dependency graph requirements

- Syntactic structure is complete (connectedness, spanning)
- Hierarchical (acyclic)
- Every word has a single head

# Projectivity

- A dependency graph is projective if the arcs do not cross



English and Thai are mostly projective.

Some languages are more non-projective than others,  
for example German, Dutch, Czech.

When picking a parser algo, check whether it assumes projectivity

# Transition-based parsing (Nivre 2007)

- Use a stack and buffer data structure and sequentially add edges
- Characteristics
  - Greedy algo. Only goes left to right. No backtracking
    - Fast  $O(n)$
  - Requires projectivity
  - The algo is closely related to how humans parse sentences (left to right one word at a time instead of looking at the sentence as a whole)

# Arc-standard Transition-based parsing

- A stack  $\sigma$ , written with top of the stack to the right
    - Starts with the ROOT symbol
  - A buffer  $\beta$ , written with top to the left
    - Starts with the input sentence
  - A set of dependency arcs  $A$ 
    - Starts of empty
  - A set of actions
- $$\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$$
1. Shift       $\sigma, w_i | \beta, A \rightarrow \sigma | w_i, \beta, A$     Move buffer to stack
  2. Left-Arc<sub>r</sub>     $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_j, \beta, A \cup \{r(w_j, w_i)\}$
  3. Right-Arc<sub>r</sub>     $\sigma | w_i | w_j, \beta, A \rightarrow \sigma | w_i, \beta, A \cup \{r(w_i, w_j)\}$
- Finishes when  $\beta$  becomes empty

# I ate fish

Start



Shift



Shift



**Start:**  $\sigma = [\text{ROOT}], \beta = w_1, \dots, w_n, A = \emptyset$   
1. Shift  $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$   
2. Left-Arc<sub>r</sub>  $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$   
3. Right-Arc<sub>r</sub>  $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$   
**Finish:**  $\beta = \emptyset$

# I ate fish

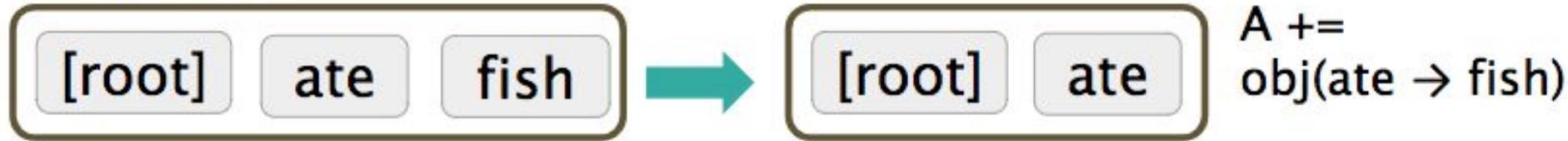
Left Arc



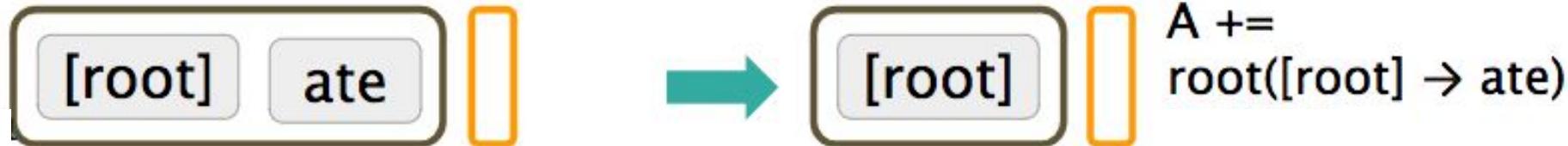
Shift



Right Arc



Right Arc



# Discriminative parsing

- How to choose an action?
  - Shift, left-arc, right-arc
- Each action is predicted by a discriminative classifier (SVM, logistic regression, Neural networks) over legal moves
  - Features: top two word from stack, POS, children info; first word in buffer, POS, children info; etc.
- Greedy and no beamsearch
  - But you can include beamsearch (modern parsers do)

# Discriminative parsing with neural networks

Shift, left, right (in actual, left/right + type of dependency)  
So  $2N+1$ , where  $N$  = dependency types

## Softmax layer:

$$p = \text{softmax}(W_2 h)$$

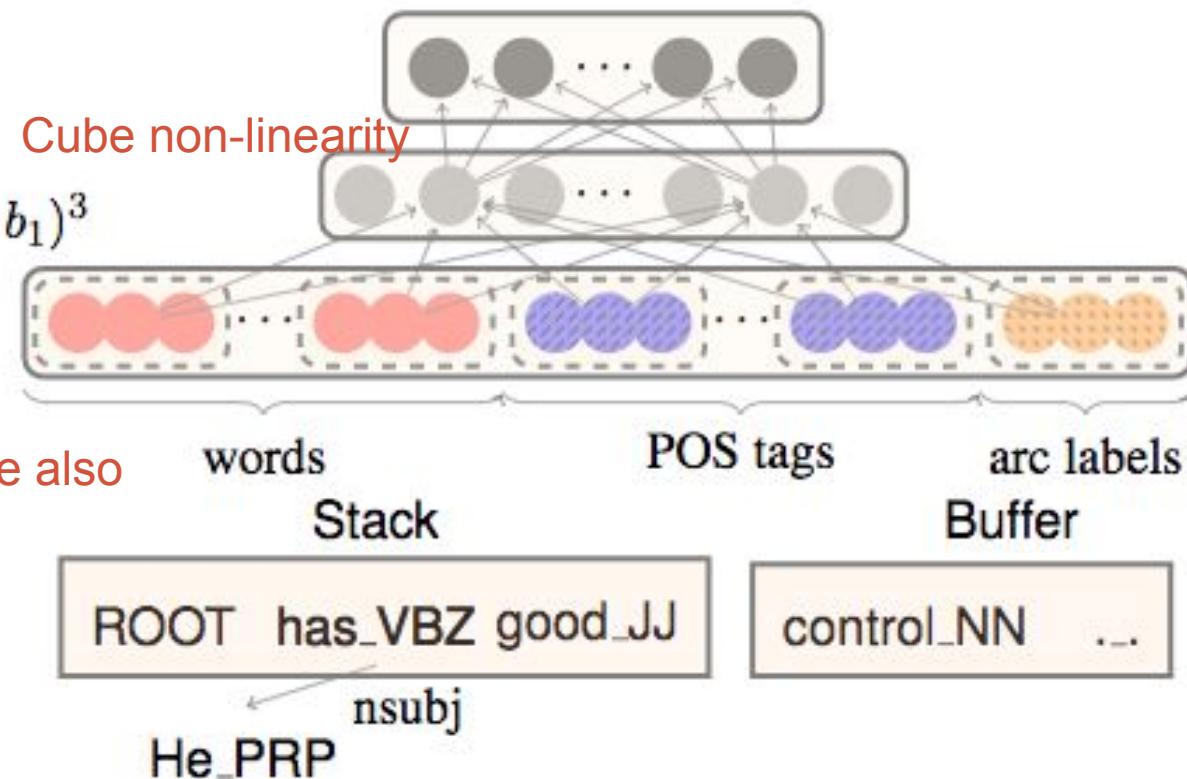
## Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

## Input layer: $[x^w, x^t, x^l]$

POS and arc labels are also embeddings

## Configuration



# Improvements

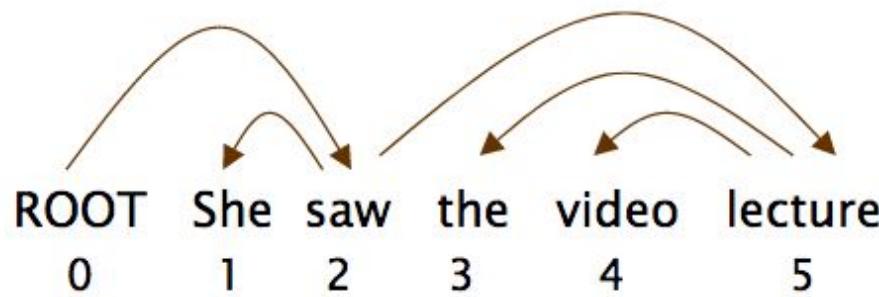
- Bigger networks
- Beam search
- Global inference over the sequence (CRF style)
- Lead to SyntaxNet and Parsey McParseFace model

<https://github.com/tensorflow/models/tree/master/research/syntaxnet>

<https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>

# Parsing evaluation

- Labeled parsing accuracy (LAS)
- Unlabeled parsing accuracy (UAS)



$$\text{Acc} = \frac{\# \text{ correct deps}}{\# \text{ of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

## Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

## Parsed

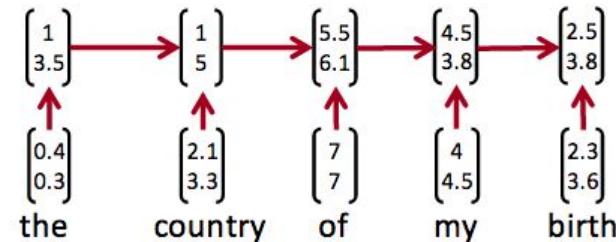
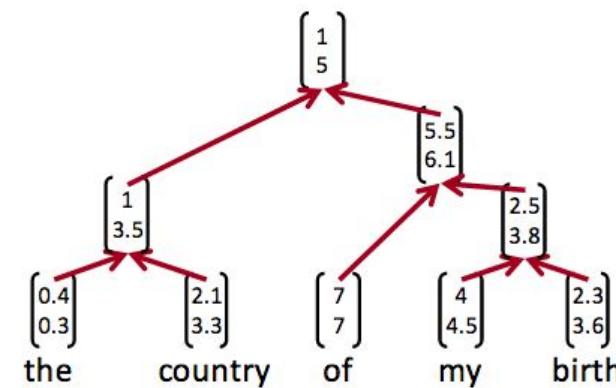
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

# UAS of Parsey McParseFace

Model	News	Web	Questions
Martins et al. (2013)	93.10	88.23	94.21
Zhang and McDonald (2014)	93.32	88.65	93.37
Weiss et al. (2015)	93.91	89.29	94.17
Andor et al. (2016)*	94.44	90.17	95.40
Parsey McParseface	94.15	89.08	94.77

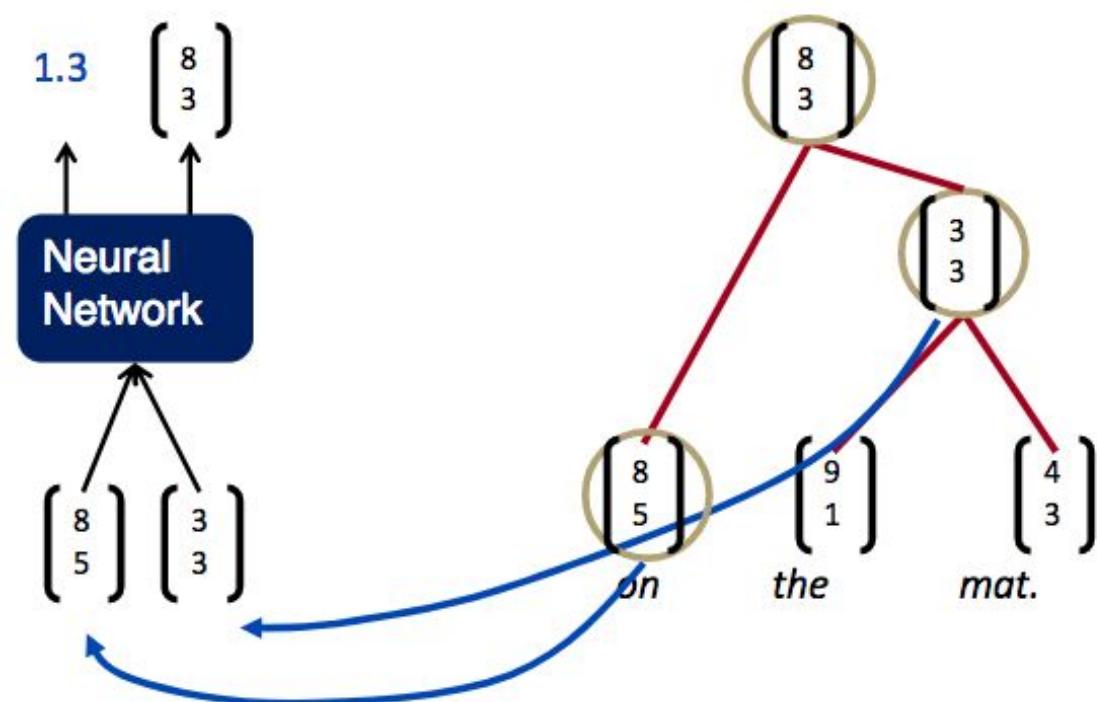
# Recursive neural networks

- Not really used in parsing anymore but interesting concept
- Recursive vs Recurrent

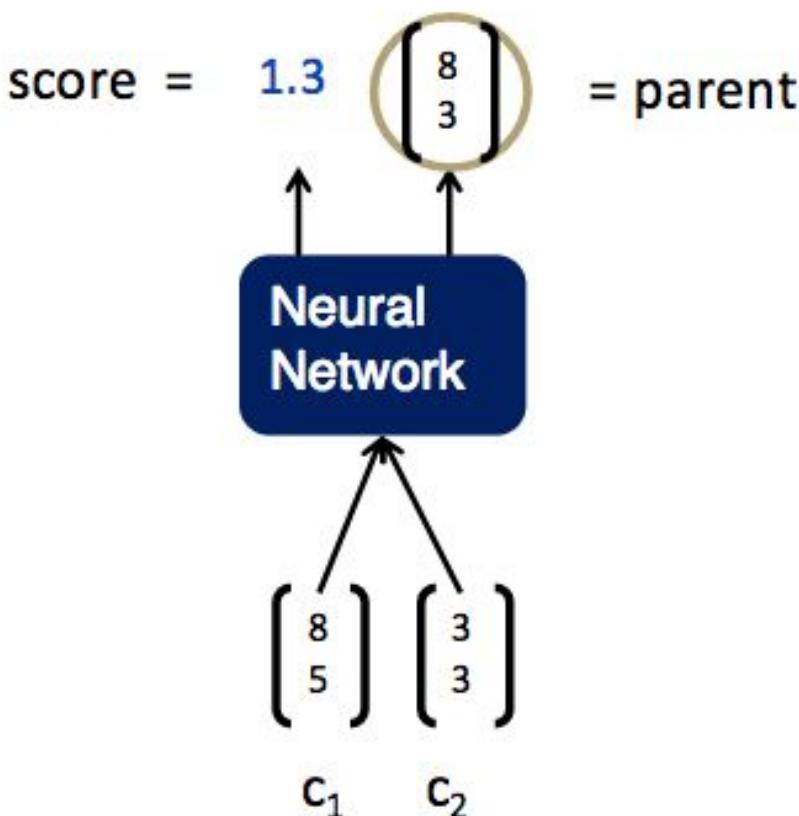


# Recursive neural networks

- Concept: try different connections, see if which one gives the highest score (graph-based dependency parsers)
- Inputs: two candidate children representations
- Output:
  - Semantic representation of the parent
  - Score of new node

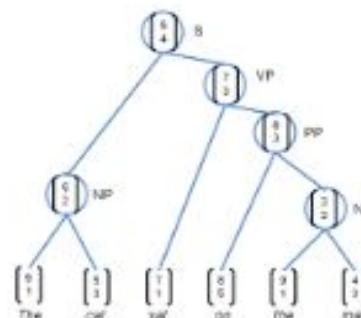


# Shared recursive structure



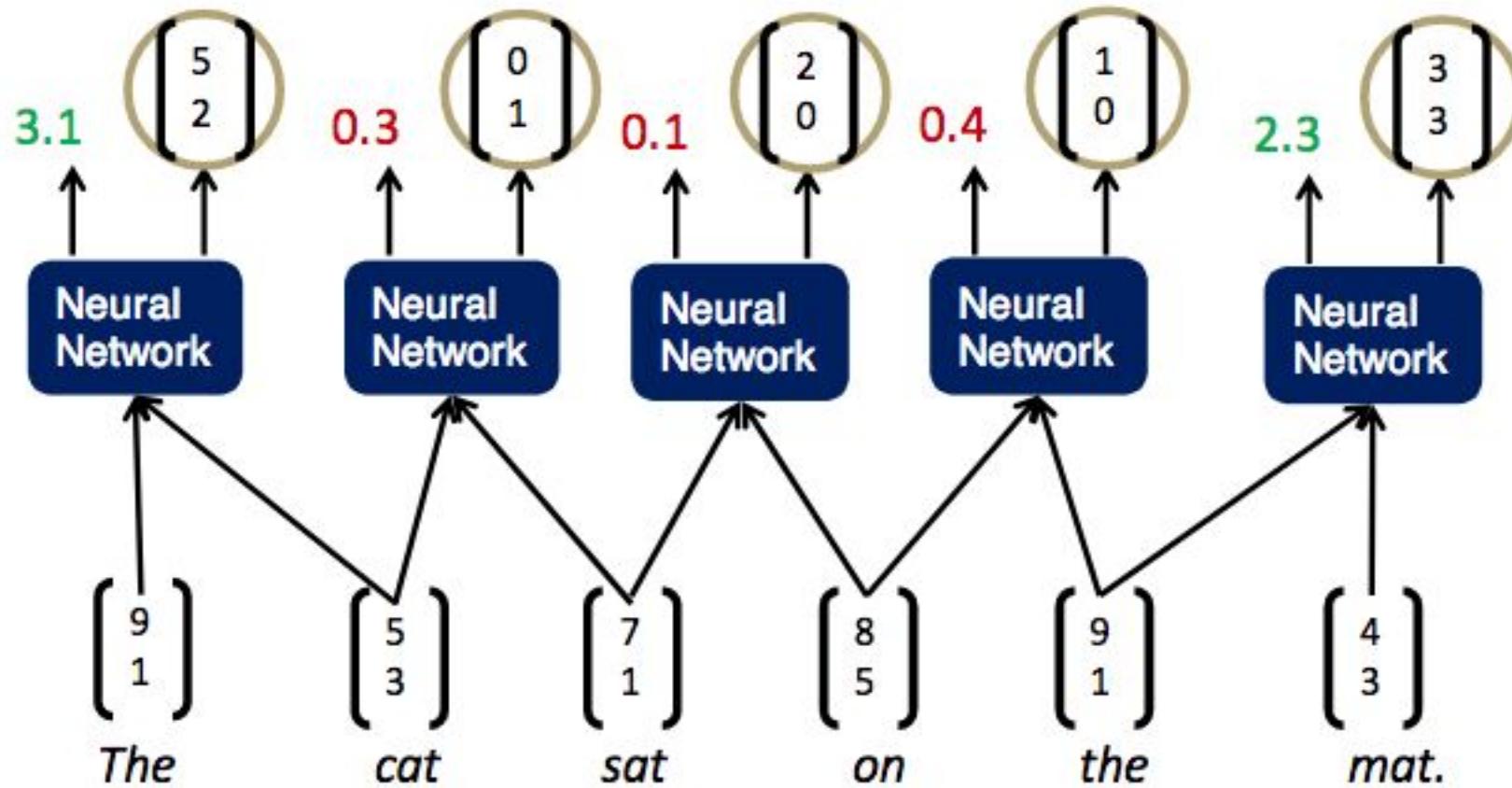
$$\left. \begin{array}{l} \text{score} = U^T p \\ p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right), \end{array} \right\}$$

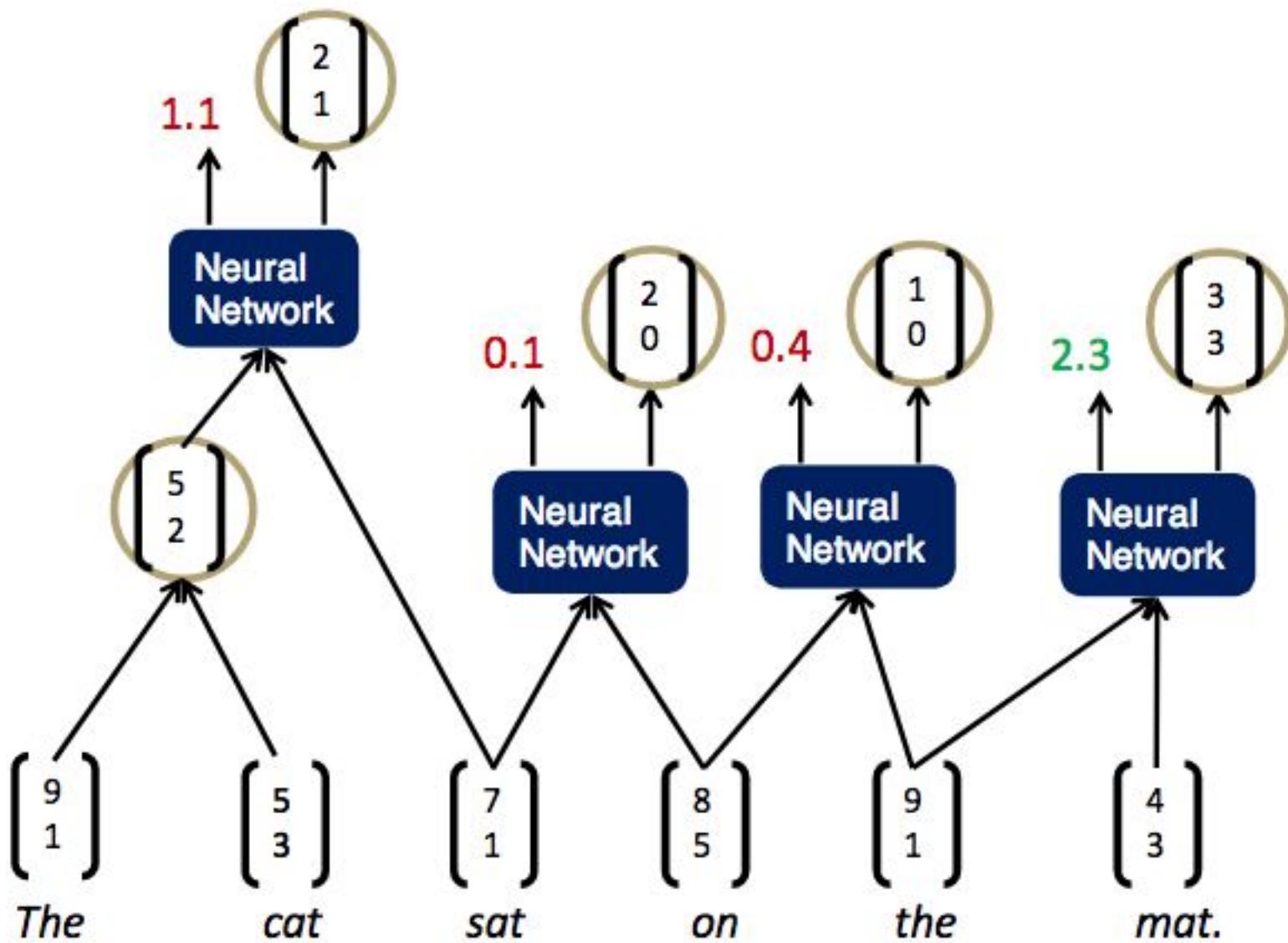
Same  $W$  parameters at all nodes of the tree

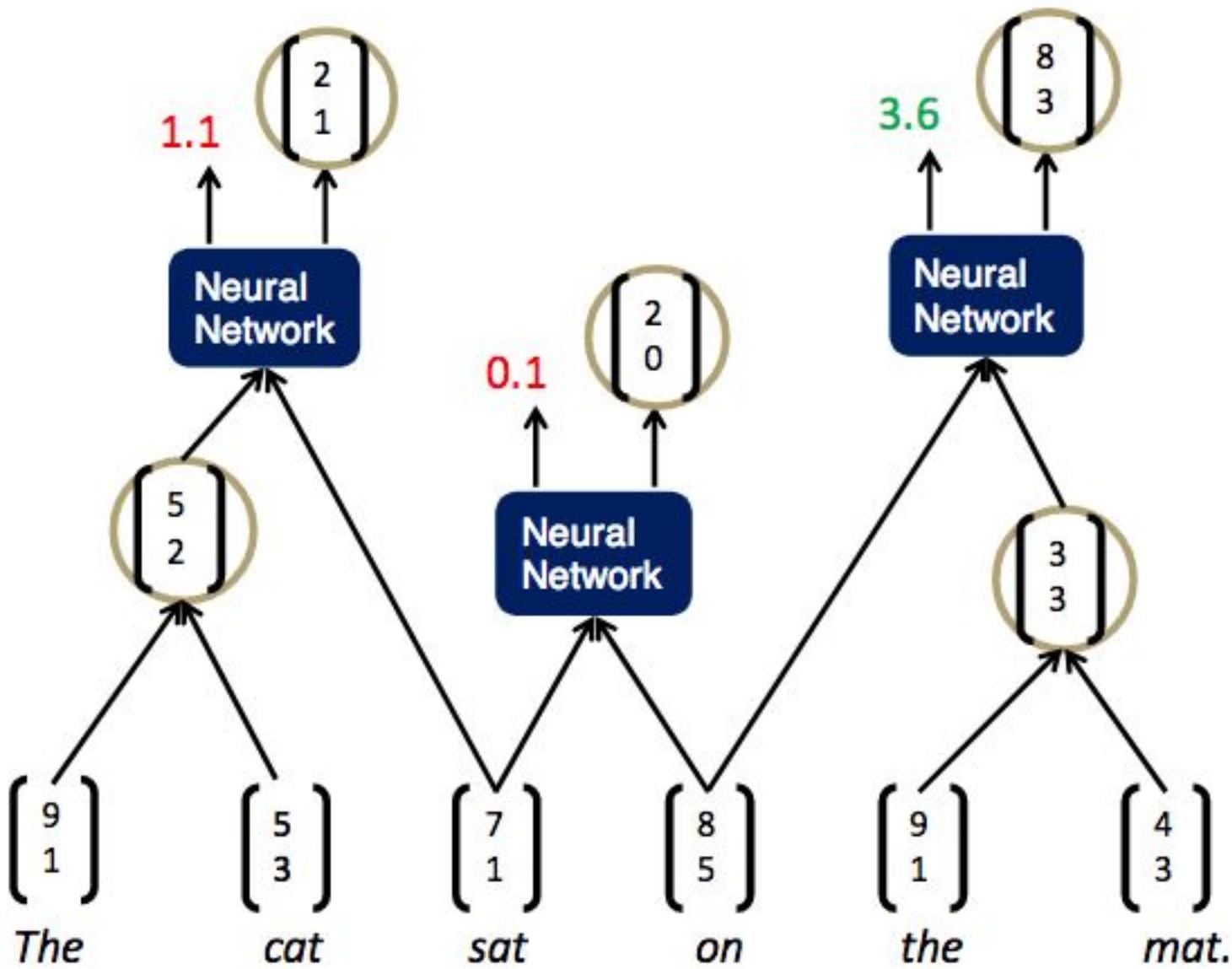


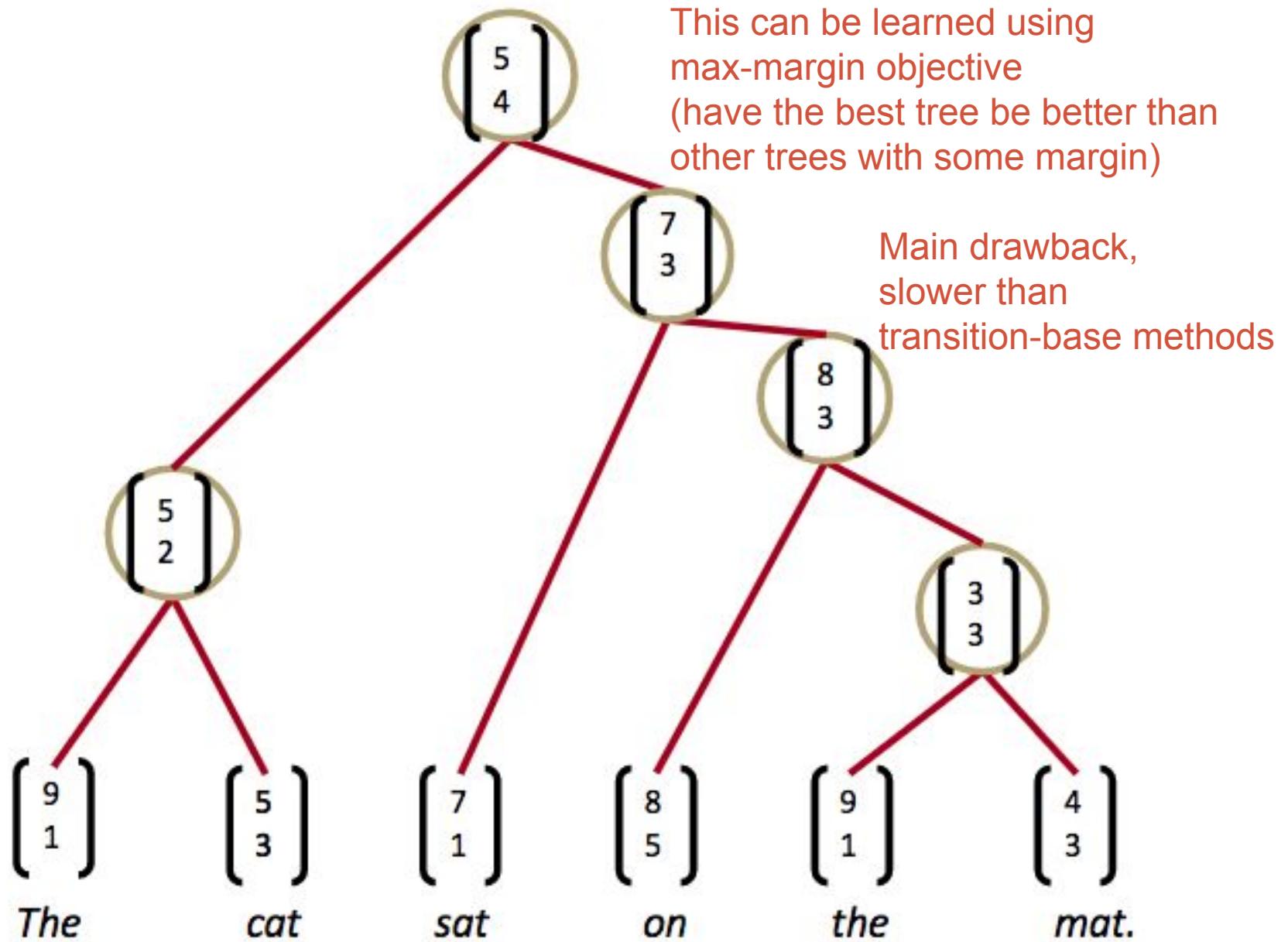
# Parsing with recursive networks

Follow the highest scoring pair

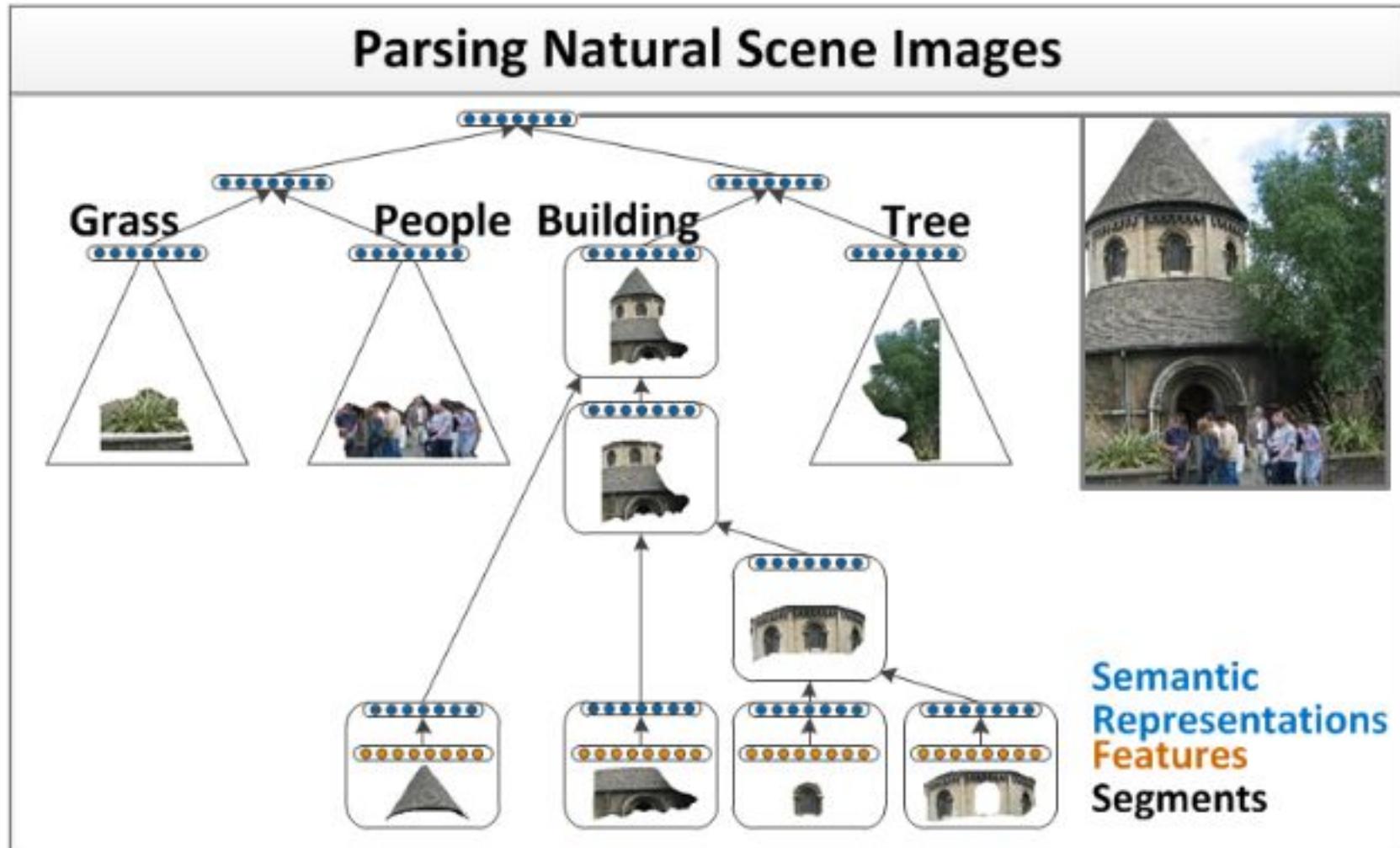








# Recursive neural networks for scene parsing

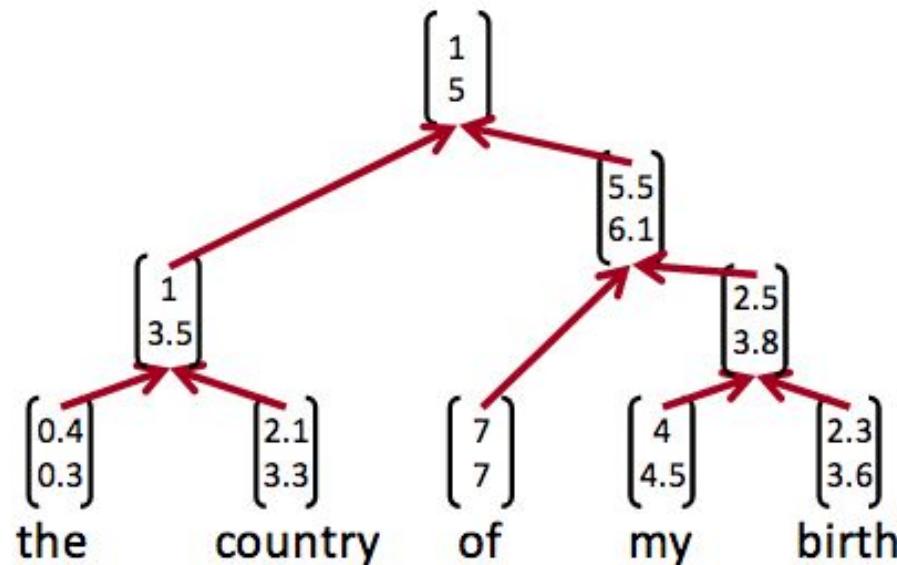


# Resources for parsing

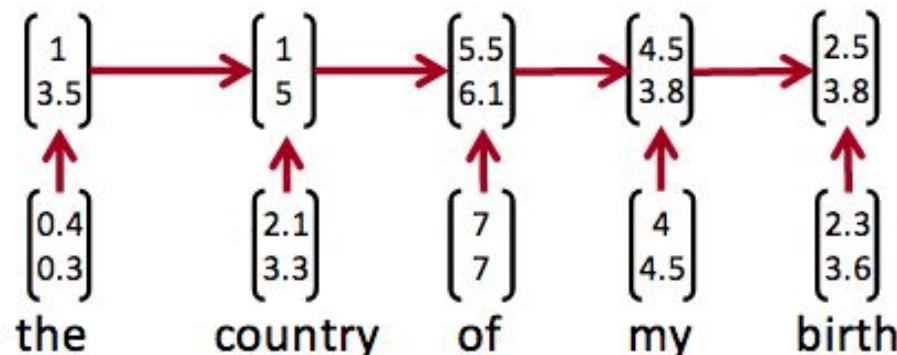
- Orchid & some corpus in development by NECTEC
- <http://universaldependencies.org/>

▶	Russian	3	1,226K	📘📘📘W	IE, Slavic
▶	Sanskrit	1	1K	📘	IE, Indic
▶	Serbian	1	86K	📘W	IE, Slavic
▶	Slovak	1	106K	📘📘📘	IE, Slavic
▶	Slovenian	2	170K	📘📘📘i	IE, Slavic
▶	Spanish	3	1,004K	📘📘📘W	IE, Romance
▶	Swedish	3	195K	📘📘📘i	IE, Germanic
▶	Swedish Sign Language	1	1K	💬	Sign Language
▶	Tamil	1	9K	📘	Dravidian, Southern
▶	Telugu	1	6K	✍	Dravidian, South Central
▶	Thai	1	23K	📘W	Tai-Kadai
▶	Turkish	2	74K	📘📘W	Turkic, Southwestern
▶	Ukrainian	1	100K	📘📘📘W	IE, Slavic
▶	Upper Sorbian	1	10K	📘i	IE, Slavic
▶	Urdu	1	138K	📘	IE, Indic
▶	Uyghur	1	15K	📘	Turkic, Southeastern
▶	Vietnamese	1	43K	📘	Austro-Asiatic

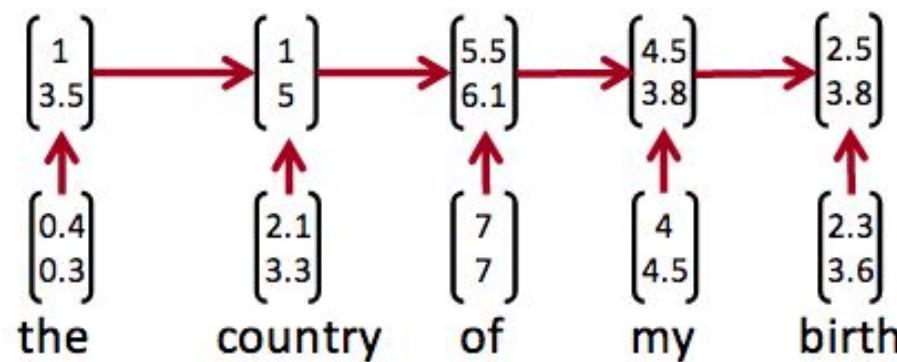
# Recursive vs Recurrent representation



resurgence of recursive idea in newer BERTs



# Towards unsupervised dependency parsing



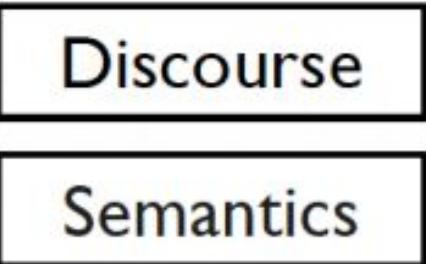
# Towards unsupervised dependency parsing



What is this piece?  
Next lecture! (attention  
modeling)

the                    country                    of                    my                    birth

Typical flow



Syntax: Constituents

Syntax: Part of Speech

Words

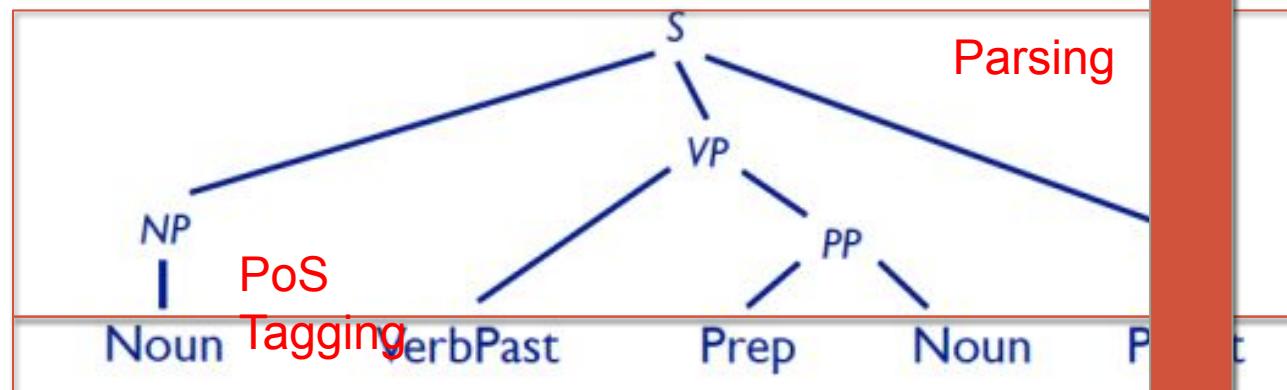
Morphology

Characters

Word embeddings

CommunicationEvent(e)  
Agent(e, Alice)  
Recipient(e, Bob)

SpeakerContext(s)  
TemporalBefore(e, s)



Alice talked to Bob.

alice talked to bob.

talk -ed [VerbPast]

# An top down end-to-end approach

- Input: text
- Output: some task, sentiment analysis score
- Automatically gets parse tree (without any treebank corpus)

Example of generated parse tree



Three men drink at a reflective bar



Workers at Basking Robbins are filling orders



Three men are socializing during happy hour



Workers filling orders at Basking Robbins

<https://arxiv.org/pdf/1705.09207.pdf>

Can also use recursive neural networks to learn unsupervised parse trees

Example <https://arxiv.org/pdf/1707.02786.pdf>

# Summary

- Parsing
- Types of grammars
  - Context Free Grammar
  - Probabilistic Context Free Grammar (Constituency parsers)
    - CYK parser
  - Dependency Grammar (Dependency parsers)
    - Transition-based parsing
    - Recursive Neural networks