CS 533: Natural Language Processing

Constituency Parsing

Karl Stratos

Rutgers University

Project Logistics

1. Proposal (due 3/24)
2. Milestone (due 4/15)
3. Presentation (tentatively 4/29)
4. Final report (due 5/4)

Possible Project Types (More Details in Template)

- Extend and apply recent machine learning methods to previously unconsidered NLP tasks
  - Search last/this year's AISTATS/ICLR/ICML/NeurIPS/UAI publications
- Extend a recent machine learning method in NLP
  - Search last/this year's ACL/CoNLL/EMNLP/NAACL
- Reimplement and replicate an existing technically challenging NLP paper from scratch
  - No available public code!
- There may be a limited number of predefined projects
  - No promise
  - Priority given to those with higher assignment grades

Course Status

Covered (all in the context of neural networks)

- Language models and conditional language models
- Pretrained representations from language modeling, evaluation tasks (GLUE, SuperGLUE)
- Tagging, parsing (today)

Will cover

- Latent-variable models (EM, VAEs)
- Information extraction (entity linking)

before the proposal due date

You will have enough background to read state-of-the-art research papers for your proposal.

HMM Loose Ends

- Recall HMM

$$p(x_1 \dots x_n,\ y_1 \dots y_n) = \prod_{i=1}^{n+1} t(y_i|y_{i-1}) \times \prod_{i=1}^{n} o(x_i|y_i)$$

- The **forward algorithm** computes in $O(|\mathcal{Y}|^2\, n)$

$$\alpha(i,y) = \sum_{y_1 \dots y_i \in \mathcal{Y}^i:\ y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

- The **backward algorithm** computes in $O(|\mathcal{Y}|^2\, n)$

$$\beta(i,y) = \sum_{y_i \dots y_n \in \mathcal{Y}^{n-i+1}:\ y_i = y} p(x_{i+1} \dots x_n, y_{i+1} \dots y_n | y_i)$$

Backward Algorithm

Base case. For $y \in \mathcal{Y}$,

$$\beta(n, y) = t(\texttt{STOP}|y)$$

Main. For $i = n - 1 \ldots 1$, for $y \in \mathcal{Y}$,

$$\beta(i, y) = \sum_{y' \in \mathcal{Y}} t(y'|y) \times o(x_{i+1}|y') \times \beta(i + 1, y')$$

Marginals

- Once forward and backward probabilities are computed, useful marginal probabilities can be computed.
  - Debugging tip: $p(x_1 \dots x_n) = \sum_y \alpha(i, y)\beta(i, y)$ for any i
- Marginal decoding: for $i = 1 \dots n$ predict $y \in \mathcal{Y}$ with highest

$$p(x_1 \dots x_n, y_i = y) = \alpha(i, y) \times \beta(i, y)$$

- Tag pair conditional marginals (will be useful later)

$$p(y_i = y, y_{i+1} = y' | x_1 \dots x_n) \\ = \frac{\alpha(i, y) \times t(y'|y) \times o(x_{i+1}|y') \times \beta(i+1, y')}{p(x_1 \dots x_n)}$$

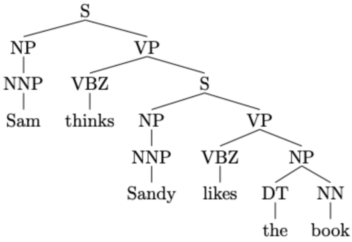
Agenda

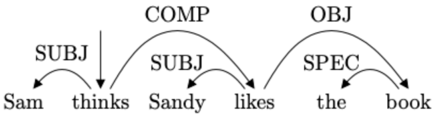
1. Parsing
2. PCFG: A model for constituency parsing
3. Transition-based dependency parsing

(Some slides adapted from Chris Manning, Mike Collins)

Constituency Parsing vs Dependency Parsing

Two formalisms for linguistic structure





Constituency Structure

Nested constituents/phrases

- Start by labeing words with POS tags

  (D the) (N dog) (V saw) (D the) (N cat)

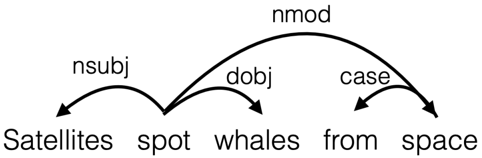
- Recursively combine constituents according to rules

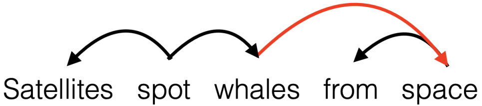
  (NP (D the) (N dog)) (V saw) (D the) (N cat)
  (NP (D the) (N dog)) (V saw) (NP (D the) (N cat))
  (NP (D the) (N dog)) (VP (V saw) (NP (D the) (N cat)))
  (S (NP (D the) (N dog)) (VP (V saw) (NP (D the) (N cat))))

  Used rules: NP $\rightarrow$ D N, VP $\rightarrow$ V NP, S $\rightarrow$ NP VP

Dependency Structure

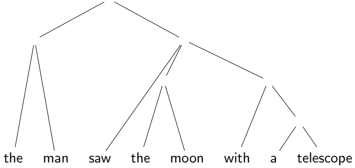
Labeled pairwise relations between words



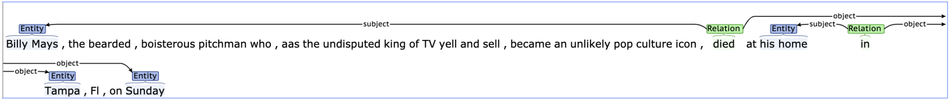


Case for Parsing

- Most compelling example of latent structure in language



- Hypothesis: parsing is intimately related to intelligence
- Also some useful applications, e.g., relation extraction



Context-Free Grammar (CFG)

A tuple $G = (N, \Sigma, R, S)$

- N: non-terminal symbols (constituents)
- Σ: terminal symbols (words)
- R: rules of form $X \to Y_1 \dots Y_m$ where $X \in N, Y_i \in N \cup \Sigma$
- $S \in N$: start symbol

Example CFG

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

$R =$

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

Grammar

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

Lexicon

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

Left-Most Derivation

Given a CFG $G = (N, \Sigma, R, S)$, a left-most derivation is a sequence of strings $s_1 \ldots s_n$ where

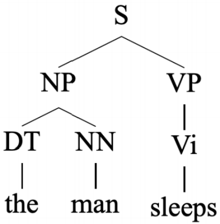
- $s_1 = S$
- For $i = 2 \ldots n$,

$$s_i = \text{ExpandLeftMostNonterminal}(s_{i-1})$$

- $s_n \in \Sigma^*$ (aka "yield" of the derivation)

Example

- $s_1 = \text{S}$
- $s_2 = \text{NP VP}$
- $s_3 = \text{DT NN VP}$
- $s_4 = \text{the NN VP}$
- $s_5 = \text{the man VP}$
- $s_6 = \text{the man Vi}$
- $s_7 = \text{the man sleeps}$



$R =$

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

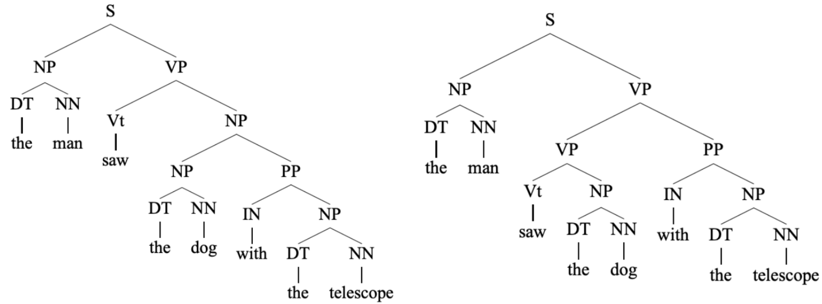
Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

<u>A derivation can be represented as a parse tree!</u>

- A string $s \in \Sigma^*$ is in the language defined by the CFG if there is at least one derivation whose yield is s
- The set of possible derivations may be finite or infinite

Ambiguity

Some string can have multiple valid derivations (i.e., parse trees).



Number of binary trees over $n+1$ nodes (n-th Catalyn number)

$$C_n = \frac{1}{n+1}\binom{2n}{n} > 6.5 \text{ billion for } n = 20$$

Rule-Based to Statistical

- Rule-based: manually construct some CFG that recognizes as many English strings as possible
  - Never enough, no way to choose the right parse
- Statistical: annotate sentences with parse trees (aka. treebank) and learn a statistical model to disambiguate

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

The Penn Treebank Project (Marcus et al, 1993)

Treebanks

- Standard setup: WSJ portion of Penn Treebank
  - 40,000 trees for training
  - 1,700 trees for validation
  - 2,400 trees for evaluation
- Building a treebank vs building a grammar?
  - Broad coverage, more natural annotation
  - Contains distributional information of English
  - Can be used to evaluate parsers

Probabilistic Context-Free Grammar (PCFG)

A tuple $G = (N, \Sigma, R, S, q)$

- N: non-terminal symbols (constituents)
- Σ: terminal symbols (words)
- R: rules of form $X \rightarrow Y_1 \ldots Y_m$ where $X \in N, Y_i \in N \cup \Sigma$
- $S \in N$: start symbol
- q: rule probability $q(\alpha \rightarrow \beta) \geq 0$ for every rule $\alpha \rightarrow \beta \in R$ such that $\sum_\beta q(X \rightarrow \beta) = 1$ for any $X \in N$

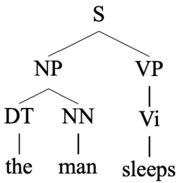
S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

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

Probability of a Tree Under PCFG

For any derivation (parse tree) containing rules:
$\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_l \rightarrow \beta_l$, the probability of the parse is:

$$\prod_{i=1}^{l} q(\alpha_i \rightarrow \beta_i)$$



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

$$P(t) = q(\mathrm{S} \rightarrow \mathrm{NP\ VP}) \times q(\mathrm{NP} \rightarrow \mathrm{DT\ NN}) \times q(\mathrm{DT} \rightarrow \mathrm{the})$$
$$\times\, q(\mathrm{NN} \rightarrow \mathrm{man}) \times q(\mathrm{VP} \rightarrow \mathrm{Vi}) \times q(\mathrm{Vi} \rightarrow \mathrm{sleeps})$$

$$= 1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0 = 0.084$$

Estimating a PCFG from a Treebank

Given trees $t^{(1)} \dots t^{(N)}$ in the training data

- N: all non-terminal symbols (constituents) seen in the data
- Σ: all terminal symbols (words) seen in the data
- R: all rules seen in the data
- $S \in N$: special start symbol (if the data does not already have it, add it to every tree)
- q: MLE estimate

$$q(\alpha \to \beta) = \frac{\textbf{count}(\alpha \to \beta)}{\sum_{\beta} \textbf{count}(\alpha \to \beta)}$$

If we see VP $\to$ Vt NP 10 times and VP 1000 times, than $q(\text{VP} \to \text{Vt NP}) = 0.01$

Aside: Improper PCFG

$$A \rightarrow A\ A \qquad \text{with probability } \gamma$$
$$A \rightarrow a \qquad \text{with probability } 1-\gamma$$

Lemma. Define $S_h = \sum_{t:\ \mathrm{height}(t) \leq h} p(t)$. Let $S^* = \lim_{h \rightarrow \infty} S_h$. Then $S^* < 1$ if $\gamma > 0.5$.

- Total probability of parses is less than one!

Fortunately, an MLE from a finite treebank is never improper (aka. "tight") (Chi and Geman, 2015)
https://www.aclweb.org/anthology/J98-2005.pdf

Marginalization and Inference

GEN$(x_1 \ldots x_n)$ denotes the set of all valid derivations for $x_1 \ldots x_n$ under the considered PCFG.

1. What is the probability of $x_1 \ldots x_n$ under a PCFG?

$$\sum_{t \in \textbf{GEN}(x_1 \ldots x_n)} p(t)$$

2. What is the most likely tree of $x_1 \ldots x_n$ under a PCFG?

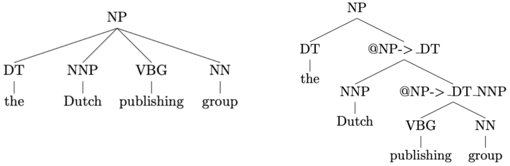
$$\arg\max_{t \in \textbf{GEN}(x_1 \ldots x_n)} p(t)$$

Chomsky Normal Form (CNF)

From here on, always assume that a PCFG $G = (N, \Sigma, R, S, q)$ is in CNF: meaning every $\alpha \rightarrow \beta \in R$ is either

1. Binary non-terminal production: $X \rightarrow Y\ Z$ where $X, Y, Z \in N$
2. Unary terminal production: $X \rightarrow x$ where $X \in N$, $x \in \Sigma$

Not a big deal: can convert PCFG to equivalent CNF (and back)



Inside Algorithm: Bottom-Up Marginalization

- For $1 \leq i \leq j \leq n$, for all $X \in N$,

$$\alpha(i, j, X) = \sum_{t \in \textbf{GEN}(x_i \ldots x_j):\ \mathrm{root}(t) = X} p(t)$$

  We will see that computing each $\alpha(i, j, X)$ takes $O(n\,|R|)$ time using **dynamic programming**.

- Return

$$\alpha(1, n, S) = \sum_{t \in \textbf{GEN}(x_1 \ldots x_n)} p(t)$$

- Total runtime?

Base Case ($i = j$)

For $i = 1 \dots n$, for $X \in N$,

$$\alpha(i, i, X) = q(X \to x_i)$$

Main Body ($j = i + l$)

For $l = 1 \dots n - 1$, for $i = 1 \dots n - l$ (set $j = i + l$), for $X \in N$,

$$\alpha(i, j, X) = \sum_{\substack{i \leq k < j \\ X \to Y\ Z \in R}} q(X \to Y\ Z) \times \alpha(i, k, Y) \times \alpha(k + 1, j, Z)$$

CKY Parsing Algorithm: Bottom-Up Maximization

- For $1 \leq i \leq j \leq n$, for all $X \in N$,

$$\pi(i, j, X) = \max_{t \in \textbf{GEN}(x_i \ldots x_j):\ \mathrm{root}(t) = X} p(t)$$

- Base: $\pi(i, j, X) = q(X \rightarrow x_i)$
- Main: $\pi(i, j, X) = \max_{i \leq k < j,\ X \rightarrow Y\ Z \in R} q(X \rightarrow Y\ Z) \times \pi(i, k, Y) \times \pi(k + 1, j, Z)$

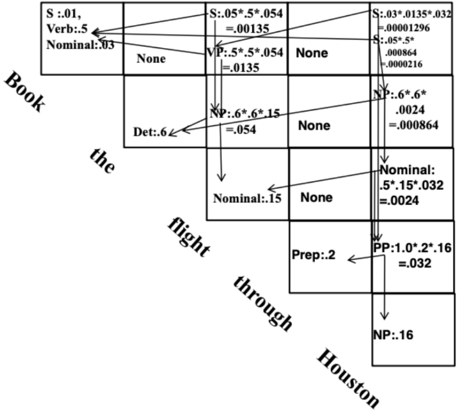
- We have

$$\pi(1, n, S) = \max_{t \in \textbf{GEN}(x_1 \ldots x_n)} p(t)$$

The optimal tree can be retrieved by backtracking

CKY Backtracking

$$b(i,j,X) = \arg\max_{\substack{i \leq k < j \\ X \to Y\ Z \in R}} q(X \to Y\ Z) \times \pi(i,k,Y) \times \pi(k+1,j,Z)$$



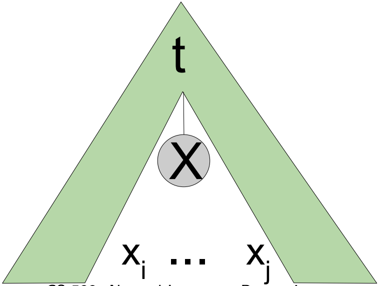
Computing Marginals Under PCFG

- Can we calculate something like

$$\mu(i, j, X) = \sum_{t \in \textbf{GEN}(x_1 \ldots x_n):\ \text{root}(t,i,j)=X} p(t)$$

- Yes, by combining the inside algorithm with the **outside algorithm**

$$\beta(i, j, X) = \sum_{t \in \textbf{OUT}(x_i \ldots x_j):\ \text{foot}(t)=X} p(t)$$



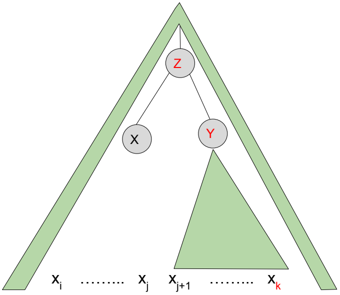
Outside Algorithm: Top-Down Marginalization

- **Base.** $\beta(1, n, S) = 1$ and $\beta(1, n, X) = 0$ for all $X \neq S$

- **Main.** For $l = n - 2 \ldots 1$, for $i = 1 \ldots n - l$ (set $j = i + l$), for $X \in N$,

$$\beta(i, j, X) = \sum_{\substack{j < k \leq n \\ Z \to X\ Y \in R}} \beta(i, k, Z) \times \alpha(j+1, k, Y) \times q(Z \to X\ Y) +$$

$$\sum_{\substack{1 \leq k < i \\ Z \to Y\ X \in R}} \beta(k, j, Z) \times \alpha(k, i-1, Y) \times q(Z \to Y\ X)$$



Max Marginal Parsing

- Inside + outside: for $1 \leq i \leq j \leq n$, for all $X \in N$,

$$\mu(i,j,X) = \sum_{t \in \textbf{GEN}(x_1 \ldots x_n):\ \mathrm{root}(t,i,j)=X} p(t) = \alpha(i,j,X) \times \beta(i,j,X)$$

- New parsing objective: find max marginal parse

$$t^* = \operatorname*{arg\,max}_{t \in \textbf{GEN}(x_1 \ldots x_n)} \sum_{(i,j,X) \in t} \mu(i,j,X)$$

- Labeled recall algorithm $O(n^3 |N|)$ (Goodman, 1996)

$$\gamma(i,j) = \max_X \mu(i,j,X) + \max_{i \leq k < j} \gamma(i,k) + \gamma(k+1,j)$$

- How many algorithms do we need for max marginal parsing??

Evaluating Parser Predictions

- Precision

$$p = \frac{\text{number of correctly predicted } (i, j, X)}{\text{number of predicted } (i, j, X)}$$

- Recall

$$r = \frac{\text{number of correctly predicted } (i, j, X)}{\text{number of ground-truth } (i, j, X)}$$

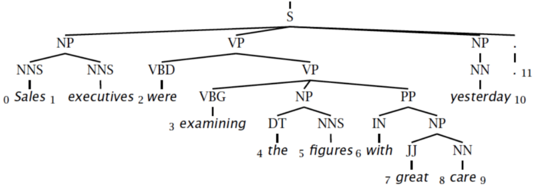
- Labeled F_1

$$F_1 = \frac{2 \times p \times r}{p + r}$$

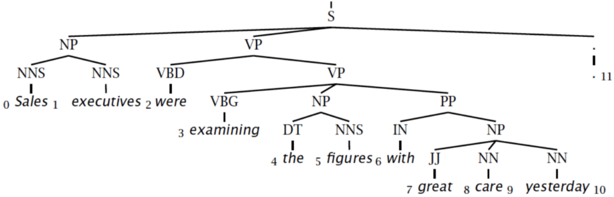
Can also consider unlabeled F_1

Example

Gold standard brackets: **S-(0:11), NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), NP-(9:10)



Candidate brackets: **S-(0:11), NP-(0:2)**, VP-(2:10), VP-(3:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

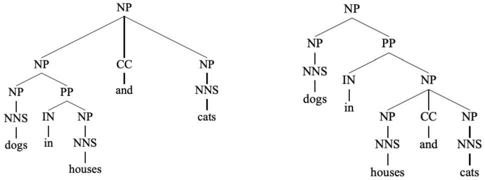


Precision 3/7 (42.9%), recall 3/8 (37.5%), labeled F_1 40

- What is the tagging accuracy?

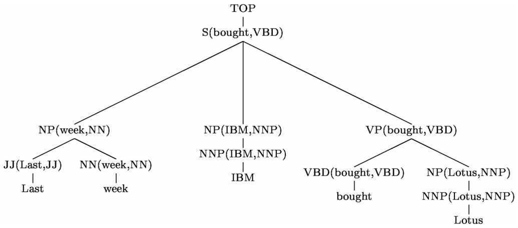
Lexicalized PCFGs

- PCFG: extremely strong conditional independence assumption



Same probability

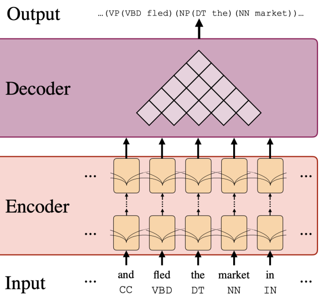
- Can consider lexicalizing the grammar (Collins, 2003)



Constituency Parsing Performance

Labeled precision/recall on PTB-WSJ

- Vanilla PCFG: 70.6% recall, 74.8% precision
- Lexicalized PCFG: 88.1% recall, 88.3% precision
- Neuralized constituency parsing (Kitaev and Klein, 2018): 94.9% recall, 95.4% precision



Neural encoding followed by max marginal decoding: no independence assumption (read the paper)