Goals for Today

- Learn about **structured prediction** problems
- Understand the **structured perceptron**
- Understand the difference between the **greedy** and the **Viterbi** inference version

Classification

Positive

My hovercraft is really bling

CLASSIFICATION: 1 LABEL FOR THE ENTIRE INPUT

Sequence Classification

DET NOUN VERB ADP VERB

DET NOUN VERB ADP ADJ

DET NOUN VERB ADP NOUN

TOO MANY POSSIBLE LABELS!

My hovercraft is really bling

CLASSIFICATION: 1 LABEL FOR THE ENTIRE INPUT

Structured Prediction

WHAT IS THE BEST LABEL SEQUENCE (y) FOR A GIVEN SENTENCE (x)?

DET NOUN VERB ADP ADJ

My hovercraft is really bling

STRUCTURED PREDCITION: 1 LABEL PER SEQUENCE ELEMENT

Examples

NER	O	O	B-PERS	I-PERS	O
POS	PRON	VERB	PROPN	PROPN	PUNCT
	I	admire	Rosa	Parks	.

POS tagging

Open class words	Closed class words	Other
ADJ adjectives: *awesome, red*	**ADP** adpositions: *over, before*	**PUNCT** punctuation marks: *!, ?, –*
ADV adverbs: *quietly, where, never*	**AUX** auxiliary/modal verbs: *have* (been), *could* (do), *will* (change)	**SYM** symbols: *%, $, :)*
INTJ interjections: *ouch, shhh*	**CCONJ** coordinating conjunctions: *and, or, but*	**X** other: *pffffrt*
NOUN nouns: *book, war*	**DET** determiners: *a, they, which*	
PROPN proper nouns: *Rosa, Twitter*	**NUM** numbers. Exactly what you would think it is...	
VERB full verbs: (she) *codes*, (they) *submitted*	**PART** particles: 's	
	PRON pronouns: *you, her, myself*	
	SCONJ subordinating conjunctions: *since, if, that*	

Ambiguity & Context

can : {VERB, NOUN, AUX}

Exact Label Depends on the Labels of Context!

PRON VERB

I ***can*** tuna

We ***can*** it for later

DET NOUN

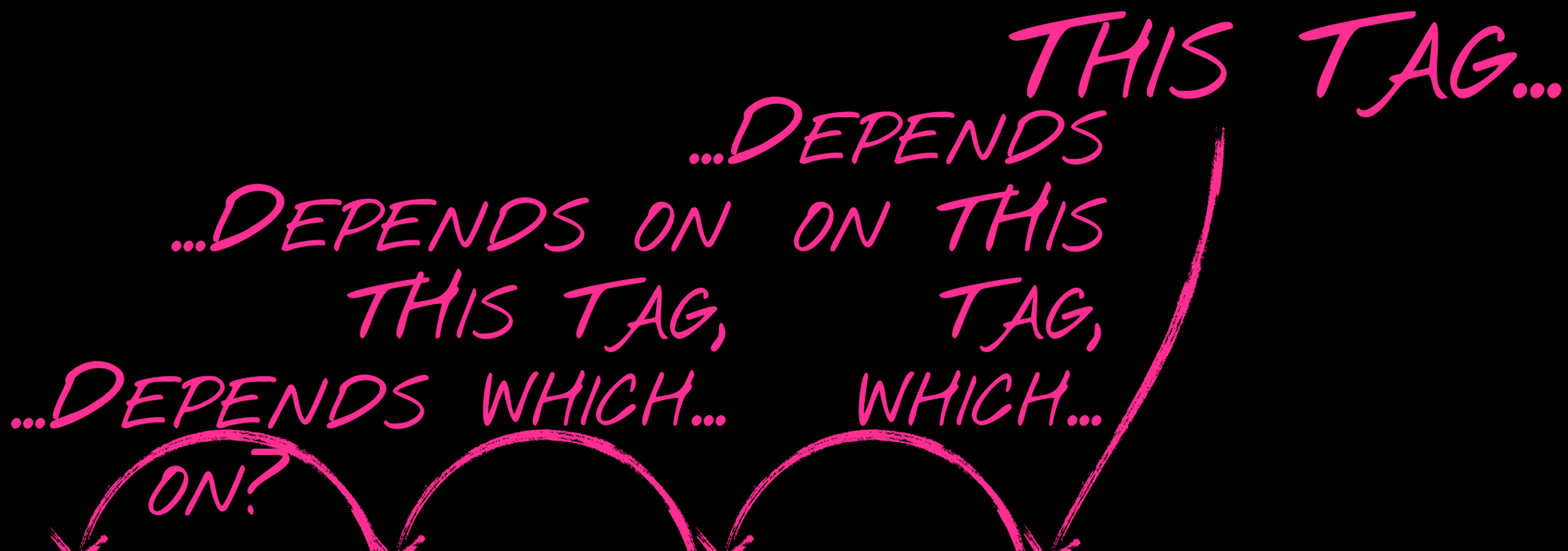
A ***can*** of tuna

The ***can*** is empty

PRON AUX

I ***can*** show you

A Recursive Problem?



START DET NOUN AUX VERB ADV

This can can fly away

SOME WORDS ARE UNAMBIGUOUS (= ONLY ONE POSSIBLE TAG)

THE START OF A SENTENCE IS UNAMBIGUOUS

The Idea:

Structured Prediction

Bocconi

A Recursive Solution

```
guess tag for first word, given previous tag was START
for each word:
    guess tag based on current word and previous tags
```

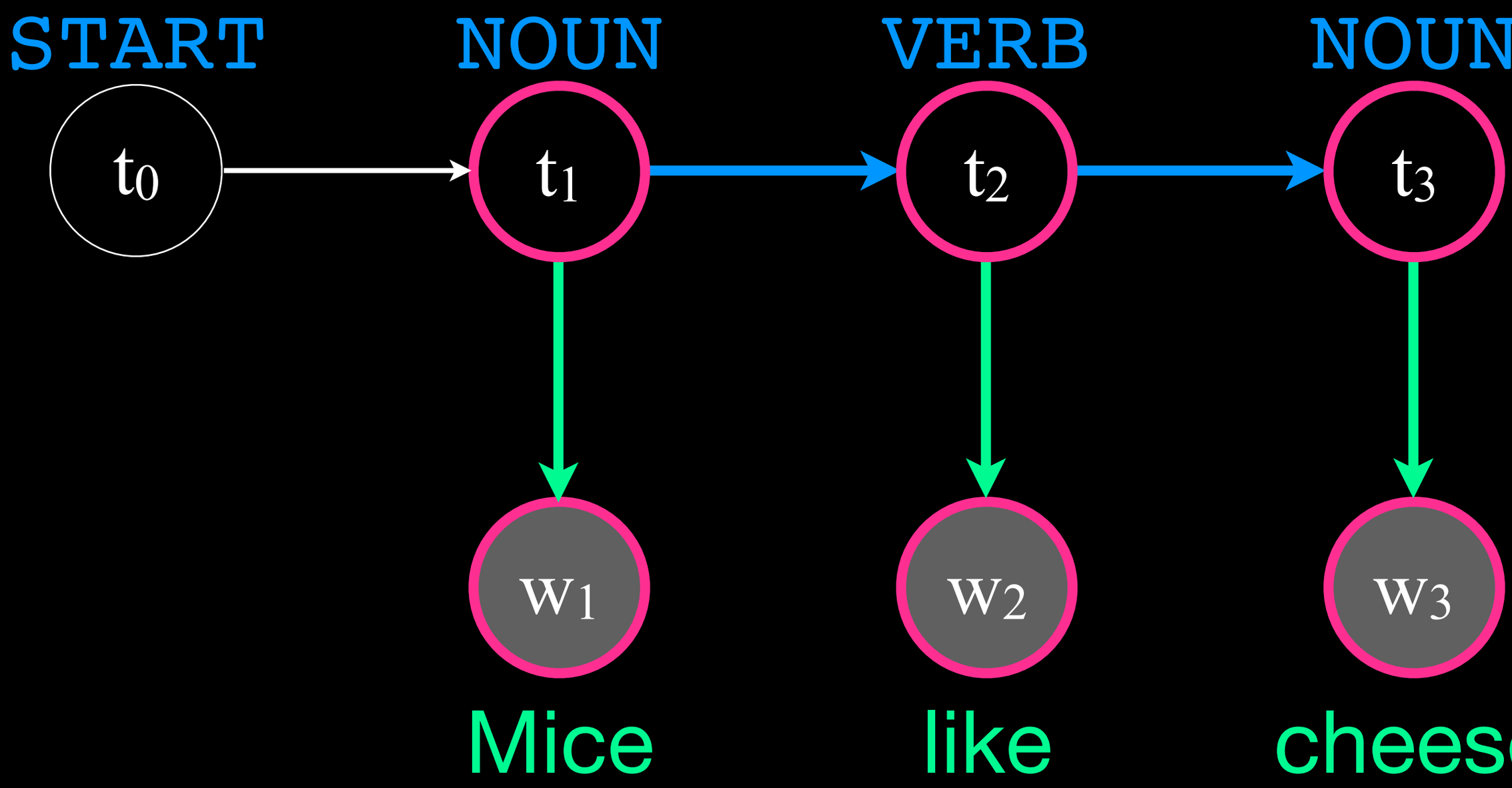
The Structure:

Hidden Markov Models

Bocconi

Structure

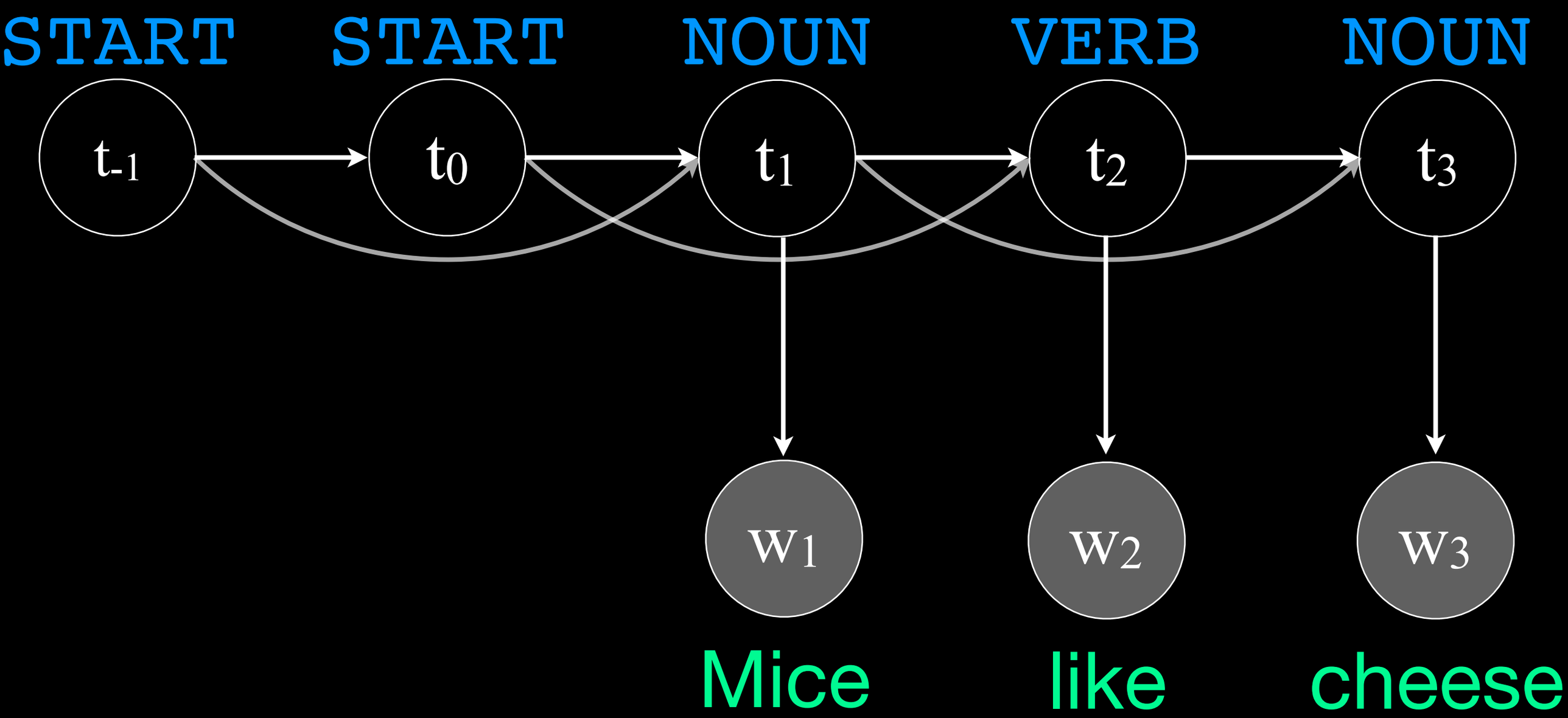
First order Markov chain



Bocconi

Structure

Second Order Markov Chain



Bocconi

The Algorithm:

Structured Perceptron

Bocconi

Collins (2002)

Structured Perceptron

```
for each iteration:

    for words, true_tags in examples:

        features = get_features(words, true_tags)

        prediction = predict(features)

        if prediction != true_tags:

            for feature in features:

                weights[feature][true_tags] += 1

                weights[feature][prediction] -= 1
```

TWO OPTIONS

UPDATE!

The Works

Inference

Dimensions

Feature

$$T|W = \arg\max_{Y \in \mathcal{Y}} \sum_{d=1}^{D} \theta_d \cdot \Phi_d(W, Y)$$

Parameters

Updates

True Features

$$\theta_d = \theta_d + \Phi_d(W, T) - \Phi_d(W, Y)$$

Predicted Features

Features

tokens

$$\Phi_d(W, T) = \sum_{i=1}^{N} \phi_d(w_i, t_{i-2}, t_{i-1}, W, i)$$

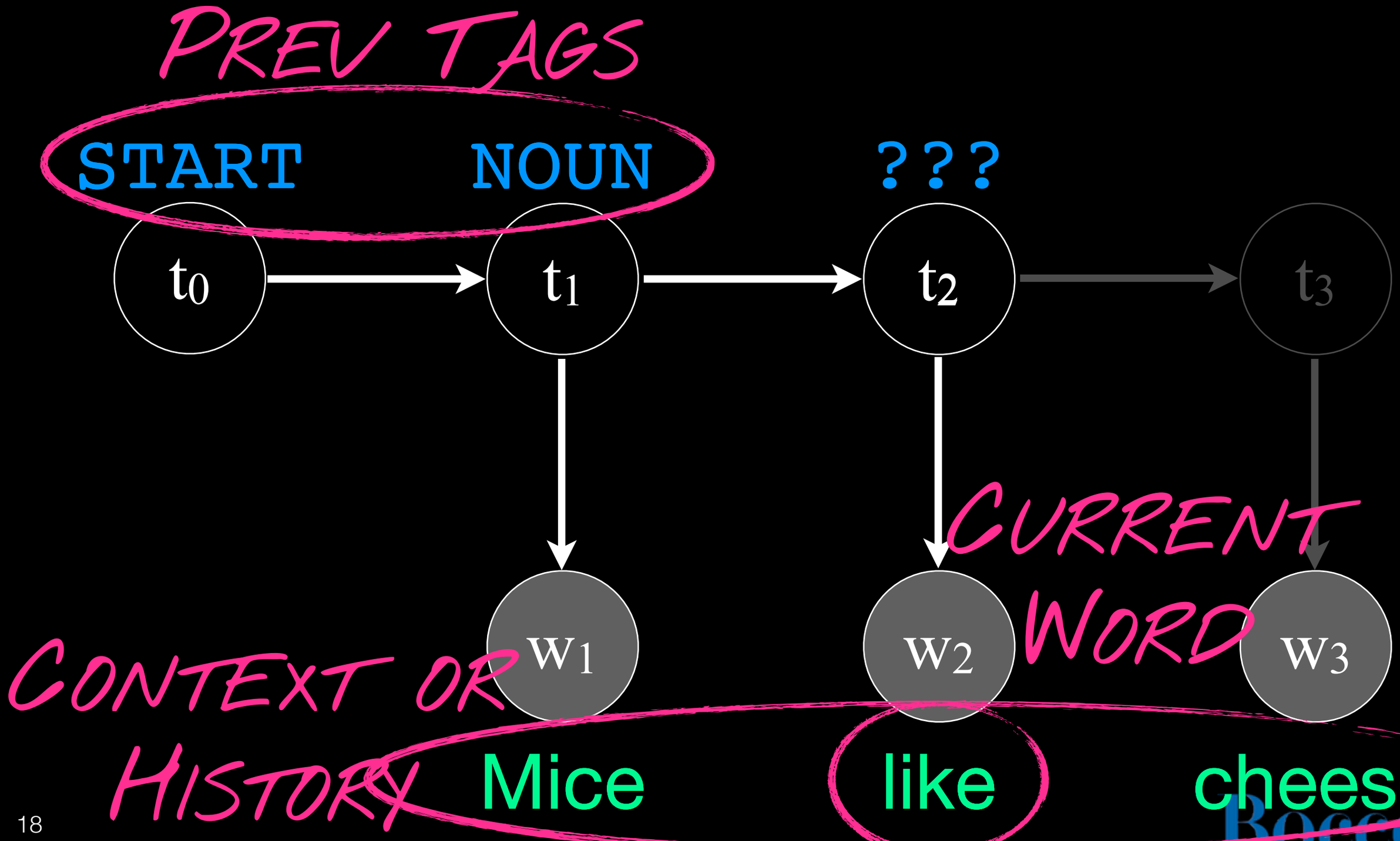
Word

Sentence

Tag History

Local Features

$\phi_2 = (\text{like}, \texttt{START}, \texttt{NOUN}, \text{Mice like cheese}, 2)$



Local Feature Function

```
def features(word, prev_tag2, prev_tag, context, i):
    return {'BIAS',
            'WORD={}'.format(word),
            'PREV_WORD={}'.format(context[i-1]),
            'PREV_WORD+WORD={}+{}'.format(context[i-1], word),
            'NEXT_WORD={}'.format(context[i+1]),
            'WORD+NEXT_WORD={}'.format(word, context[i+1]),
            'PREV_TAG={}'.format(prev_tag),
            'PREV_TAG2={}'.format(prev_tag2),
            'PREV_2TAGS={}+{}'.format(prev_tag2, prev_tag),
            'PREV_TAG+WORD={}+{}'.format(prev_tag, word)
}
```

DERIVE FEATURES FROM THE

- CURRENT WORD
- PREVIOUS TAGS
- CONTEXT

Inference

Inference Tricks

Constrain Possible Tags Based on Observed Options

NOUN	NOUN	NOUN	NOUN	NOUN
VERB	VERB	VERB	VERB	VERB
ADV	ADV	ADV	ADV	ADV
DET	DET	DET	DET	DET
This	can	can	fly	away

Unambiguous!

Greedy Inference

Predict Most likely label in each position

NOUN	NOUN	NOUN	NOUN	NOUN
VERB	VERB	VERB	VERB	VERB
ADV	ADV	ADV	ADV	ADV
DET	DET	DET	DET	DET
This	can	can	fly	away

Greedy Inference

```
for each words, tags:
    predictions = []
    for each word in words:
        scores = θ ϕ(word, tags[-2], tags[-1], words)
        tag = argmax(scores)
        predictions(tag)
```

Viterbi Inference

Predict each label based on best path to it

Bocconi

Viterbi Inference

```
# initialize scores
features = get_features(words[0], START, START, context, 1)
scores = get_scores(features)

for each allowed tag on word 1:
    Q[tag, word] = scores[tag]

# filling the lattice, for every position and every tag find Viterbi score Q
for each word i:
    for each allowed prev_tag on prev_word:
        best_score = float('-Inf')
        prev_score = Q[prev_tag, prev_word] # score of previous tag on previous word
        for each allowed prev2_tag:
            if i == 1:
                prev2_tag = START

            # get features of word i with the two previous tags
            features = get_features(word, prev2_tag, prev_tag, context, i)
            scores = get_scores(features)

            # update best score
            for each allowed tag on current word:
                tag_score = prev_score + scores[tag]

                if tag_score > best_score:
                    Q[tag, word] = tag_score
                    best_score = tag_score
                    backpointers[tag, word] = prev_tag
```

Viterbi Algorithm

This	can	can	fly	away
SCORE				
VERB	5.10	9.12	12.76	10.01
NOUN	1.34	8.32	9.03	11.47	11.43
DET	3.66	DET	DET	DET	DET
ADV	ADV	ADV	9.86	15.01

$Q(\mathit{TAG}, 0) + \phi(\text{can}, \texttt{START}, \mathit{TAG}, \textbf{\textit{words}}, 0) \cdot \theta$

$\phi(\text{This}, \texttt{START}, \texttt{START}, \textbf{\textit{words}}, 0) \cdot \theta$

INITIALIZE FIRST ROWS

Decoding

DET	NOUN	VERB	VERB	ADV
This	can	can	fly	away

SCORE

VERB	5.10	9.12	12.76	10.01
NOUN	1.34	8.32	9.03	11.47	11.43
DET	3.66	DET	DET	DET	DET
ADV	ADV	ADV	9.86	15.01

BACKPOINTERS

	DET	NOUN	VERB	VERB
START	DET	VERB		ADV
START			VERB	
				VERB

Hyper-Parameters

- Independent of inference, you need to tune
    - learning rate
    - number of iterations
    - features you generate

Wrapping Up

Bocconi

Inference Comparison

	greedy	Viterbi
speed	fast	slower
accuracy	good	better on small data
sparsity	lower	higher
type	approximate	exact

Take-home points

- **Structured prediction** finds a tag for each token, based on context
- Assume invisible **START** tags at beginning
- **Structured Perceptron** finds best sequence
- Derive good **features** from context
- **Inference** can be approximate (**greedy**) or exact (**Viterbi**)