ETH zürich

Part-of-Speech Tagging with CRFs

Ryan Cotterell and Clara Meister

ETH zürich

Administrivia

Project and Course Assignment Update

- **Course Research Project:**
  - Guidelines for the research-based course assignment have been released: https://rycolab.github.io/classes/intro-nlp/materials/project_guidelines.pdf
  - Project proposals are due Nov. 1st 23:59 CEST!
    - See guidelines for content/formatting requirements
    - Have one group member send as email to Clara

- **Course Assignment:**
  - We have released Course Assignment Episode 1: https://rycolab.github.io/classes/intro-nlp/materials/assignment.pdf
    - These questions cover Lectures 2–5
    - We will update the assignment with clarifications and corrections
  - Episode 2 will cover lectures 6–10 and will be released later

Structure of this Lecture

1. Part-of-Speech Tagging
   - Task Definition
   - Tag Sequences as Graph Paths
2. Conditional Random Fields
   - Formal Equation
   - Scoring Function
   - Efficient Global Normalization
3. Semirings
   - Definition
   - In Action
4. Reframing CRFs
   - Softmax Interpretation
   - The Structured Perceptron

Supplementary Material

Reading: Eisenstein Ch. 7 and Ch. 8

Supplementary Reading: Tim Vieira's Blog, McCallum et al. (2000), Lafferty et al. (2001), Sutton and McCallum (2011), Koller and Friedman (2009)

ETH zürich

Review

⓪ **Recap:** What is Structured Prediction?

Big picture: It's just multi-class classification combined with an algorithm of our choosing!

In more detail:

- A class of problems that involves predicting structured objects (e.g., strings, trees or graphs) rather than scalar discrete or real values (e.g., sentiment class)
- Structured prediction is the intersection of algorithms and high-dimensional statistics
- The output space of the prediction problem is exponentially large and so we must now think carefully about how to evaluate outputs

⓪ **Recap:** What is Structured Prediction?

Recall: How do we approach many tasks?

Modelling the probability distribution over outputs given inputs

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\{\text{score}(\mathbf{y}, \mathbf{x})\}}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp\{\text{score}(\mathbf{y}', \mathbf{x})\}}$$

Question: What if $|\mathcal{Y}|$ is really, really big? Can we find an efficient algorithm for computing that sum in the denominator?

⓪ **Recap:** What is Structured Prediction?

Recall: How do we approach many tasks?

Modelling the probability distribution over outputs given inputs

function of our choosing...

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\{\text{score}(\mathbf{y},\mathbf{x})\}}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp\{\text{score}(\mathbf{y}',\mathbf{x})\}}$$

softmax

Question: What if $|\mathcal{Y}|$ is really, really big? Can we find an efficient algorithm for computing that sum in the denominator?

⓪ **Recap:** What is Structured Prediction?

Consider the size of the output space of various NLP tasks...

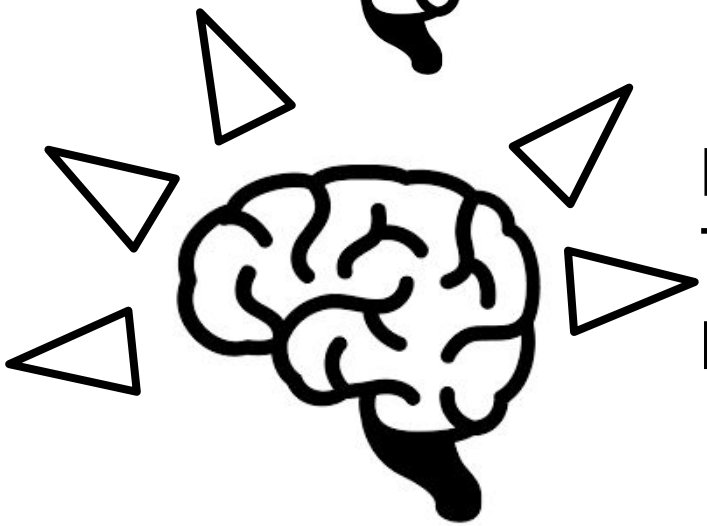
$$|\mathcal{Y}| = 2$$

Sentiment Analysis:
Is sentiment positive or negative?

$$|\mathcal{Y}| = n$$

Movie Genre Prediction:
Which genre is this script?

$$|\mathcal{Y}| = 2^n$$



Part-of-Speech Tagging:
This sentence has which part-of-speech-tag sequence?

⓪ **Recap:** What is Structured Prediction?

Consider the size of the output space of various NLP tasks...

$$|\mathcal{Y}| = 2$$

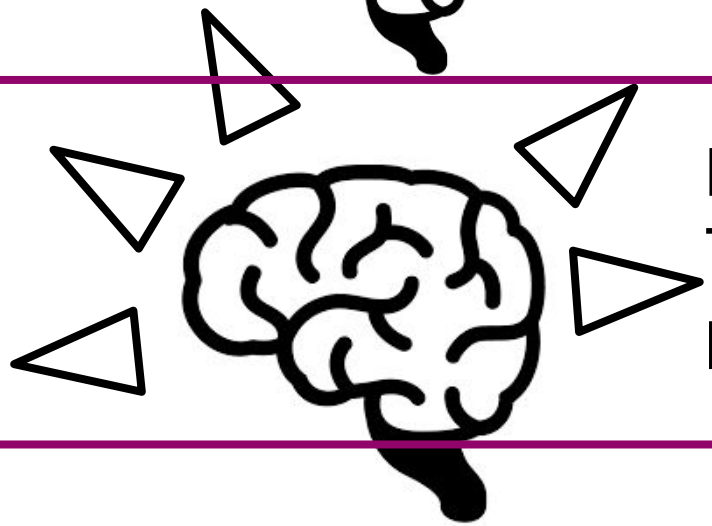
Sentiment Analysis:
Is sentiment positive or negative?

Bad idea to brute force solve this...

$$|\mathcal{Y}| = n$$

Movie Genre Prediction:
Which genre is this script?

$$|\mathcal{Y}| = 2^n$$



Part-of-Speech Tagging:
This sentence has which part-of-speech-tag sequence?

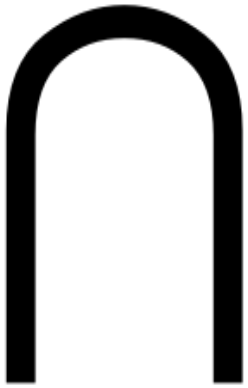
⓪ **Recap:** What is Structured Prediction?

Bottom Line: We need an additional set of tools when our output space is exponential

Statistics ∩ **(Theoretical) Computer Science**

⓪ **Recap:** What is Structured Prediction?

Bottom Line: We need an additional set of tools when our output space is exponential

Modeling  **Combinatorial Algorithms**

ETH zürich

Part-of-Speech Tagging

1 Part-of-Speech Tagging

Overview

- Assign each word in a sentence a coarse-grained grammatical category
  - Noun, Verb, Adjective, Adverb, Determiner, etc…
  - Is this syntax? We'll discuss that point next lecture
- Arguably, the simplest structured prediction problem in NLP

The boy eats rösti

1 Part-of-Speech Tagging

Overview

- Assign each word in a sentence a coarse-grained grammatical category
  - Noun, Verb, Adjective, Adverb, Determiner, etc…
  - Is this syntax? We'll discuss that point next lecture
- Arguably, the simplest structured prediction problem in NLP

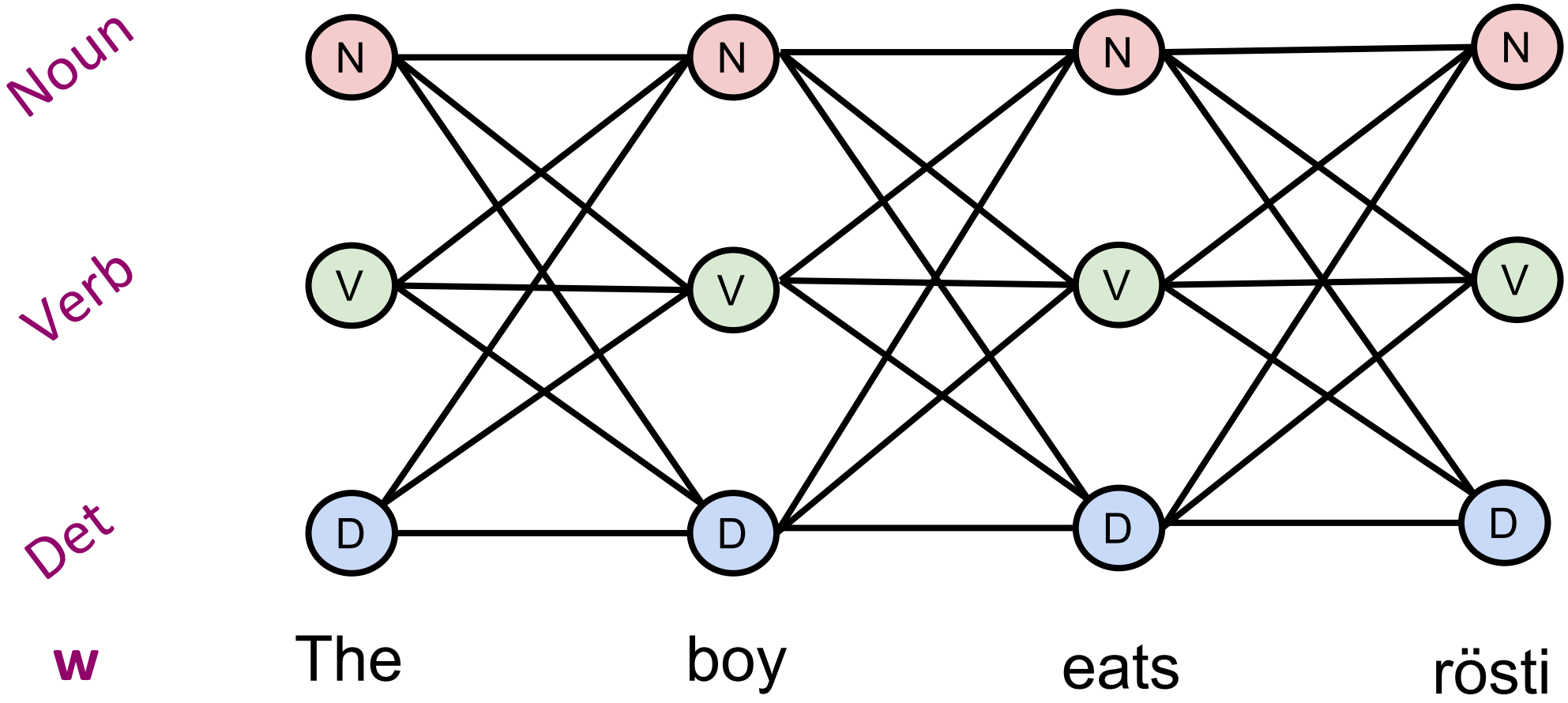
1 Part-of-Speech Tagging

Overview

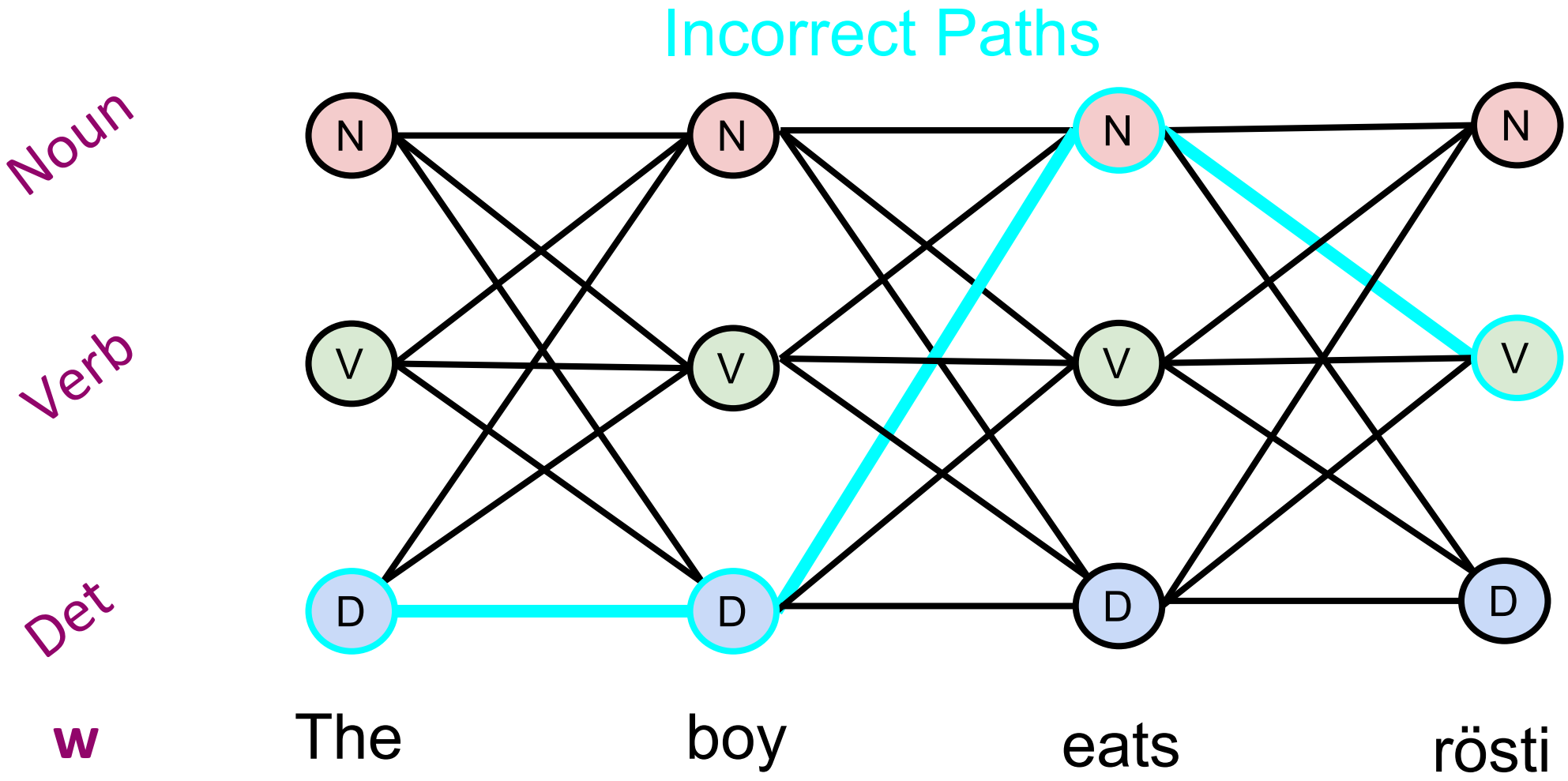
- Assign each word in a sentence a coarse-grained grammatical category
  - Noun, Verb, Adjective, Adverb, Determiner, etc…
  - Is this syntax? We'll discuss that point next lecture
- Arguably, the simplest structured prediction problem in NLP

Question: Given just 3 tags (D, N, V), what is the number of possible assignments for this sentence?

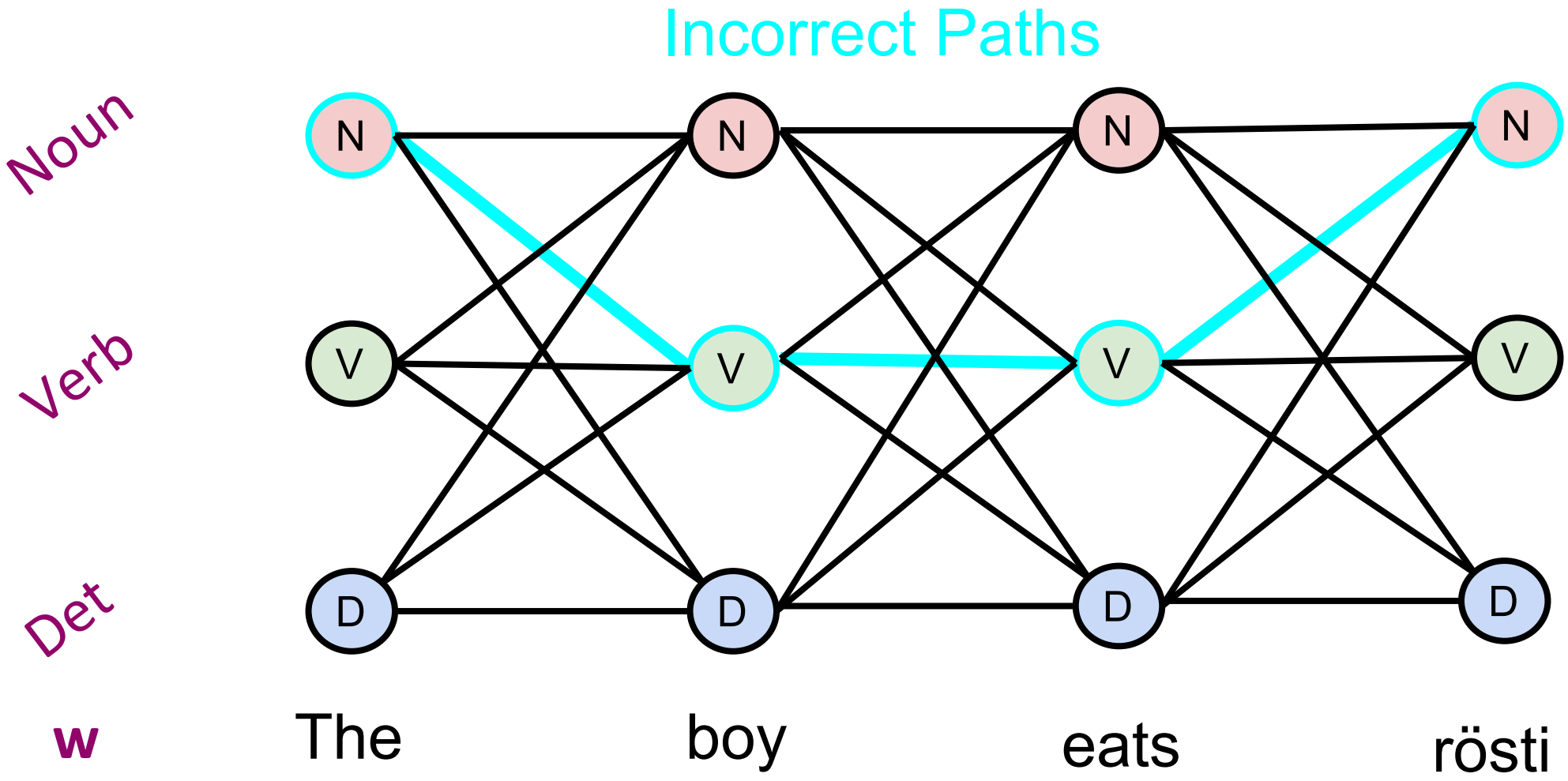
① Part-of-Speech Tagging is like Path Search in a Graph



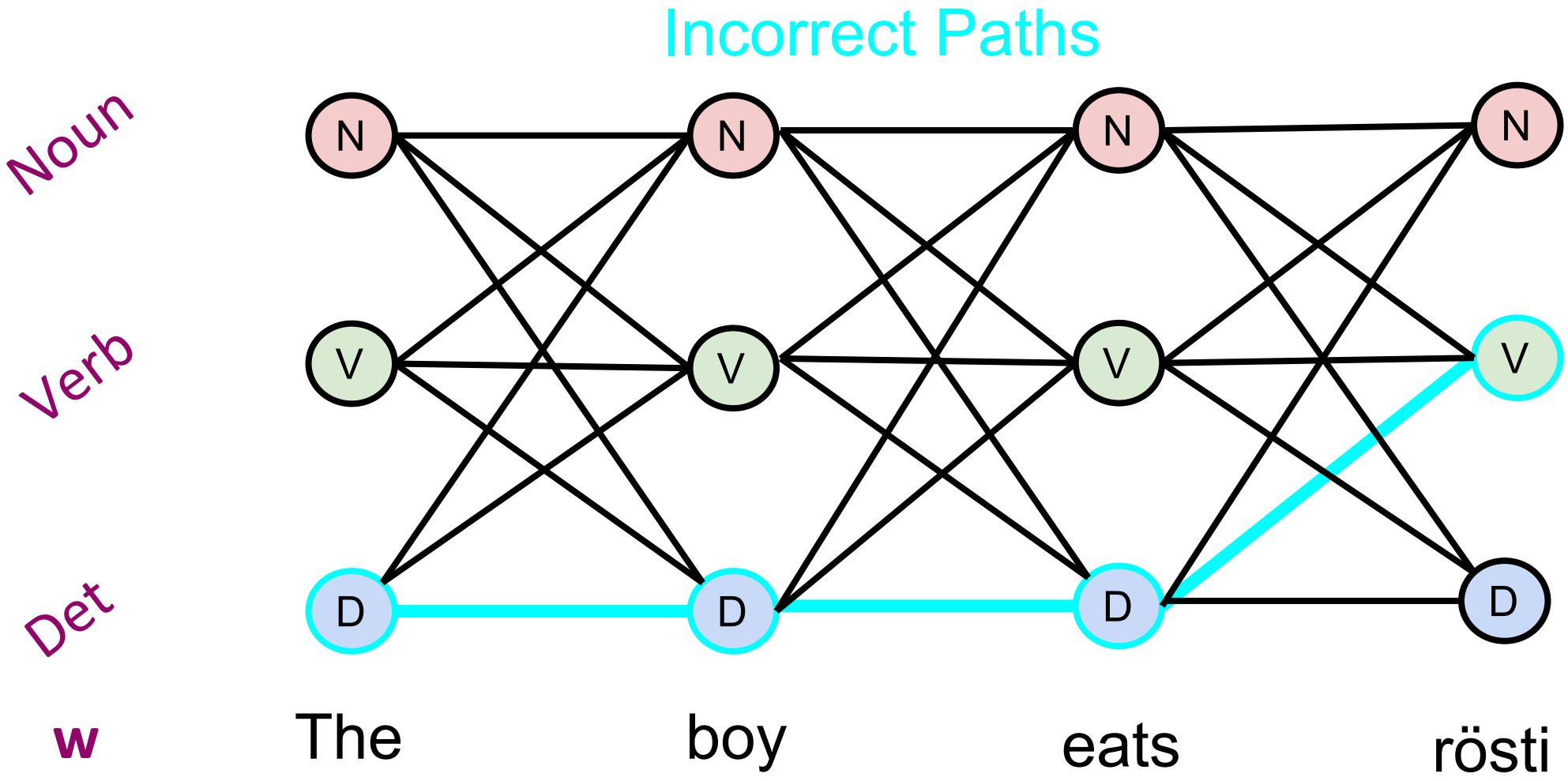
① Part-of-Speech Tagging is like Path Search in a Graph



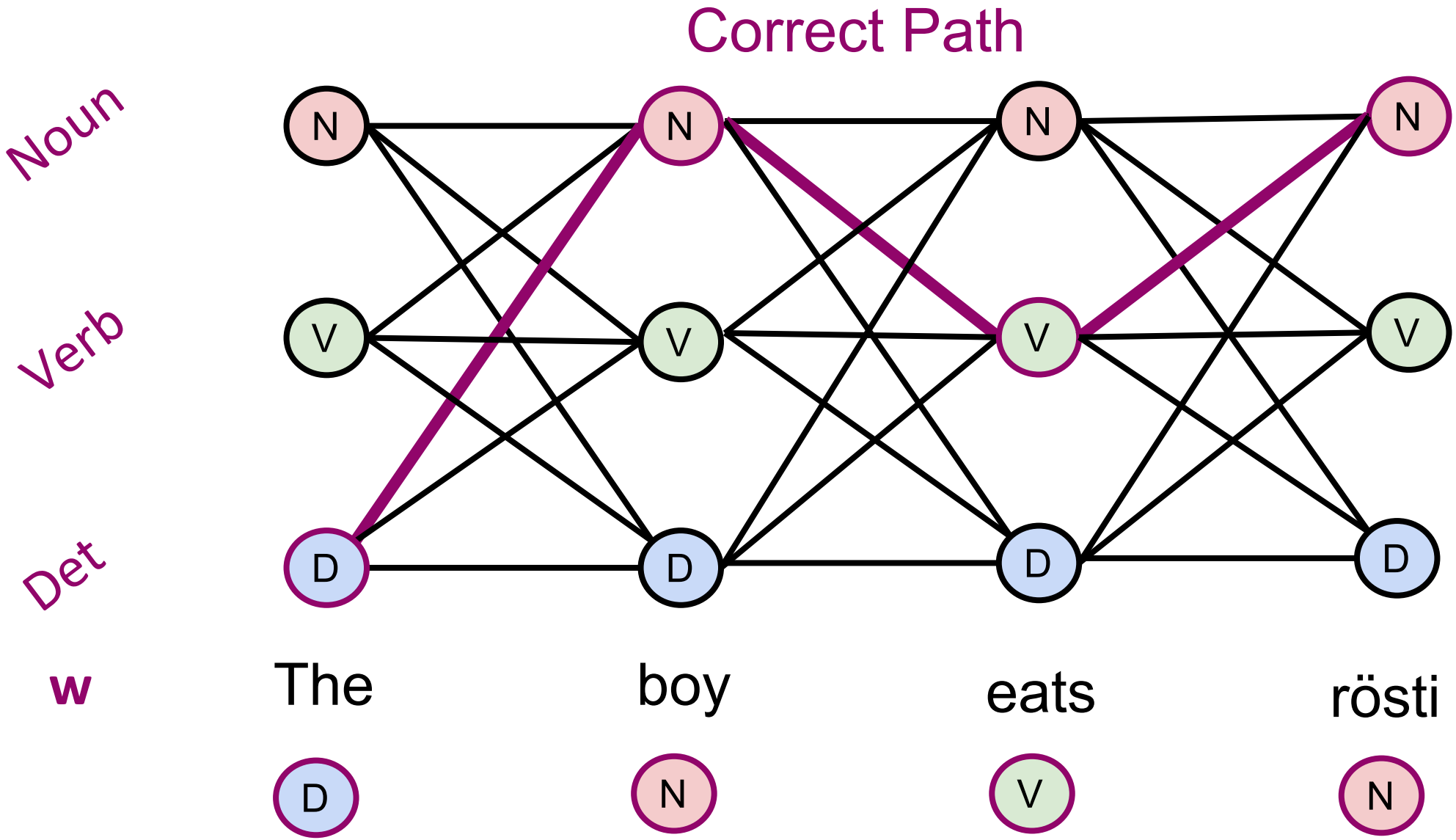
① Part-of-Speech Tagging is like Path Search in a Graph



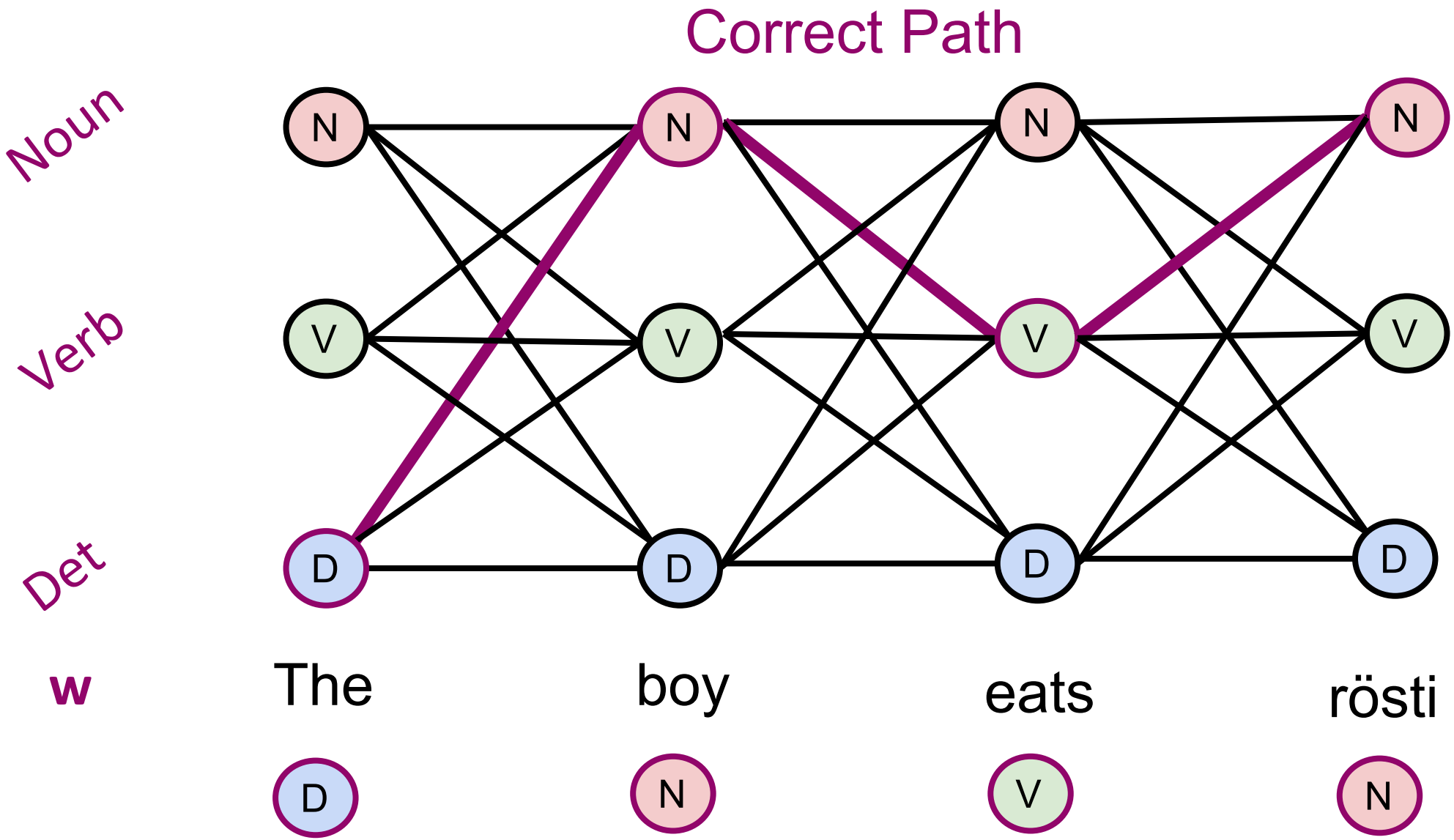
① Part-of-Speech Tagging is like Path Search in a Graph



① Part-of-Speech Tagging is like Path Search in a Graph

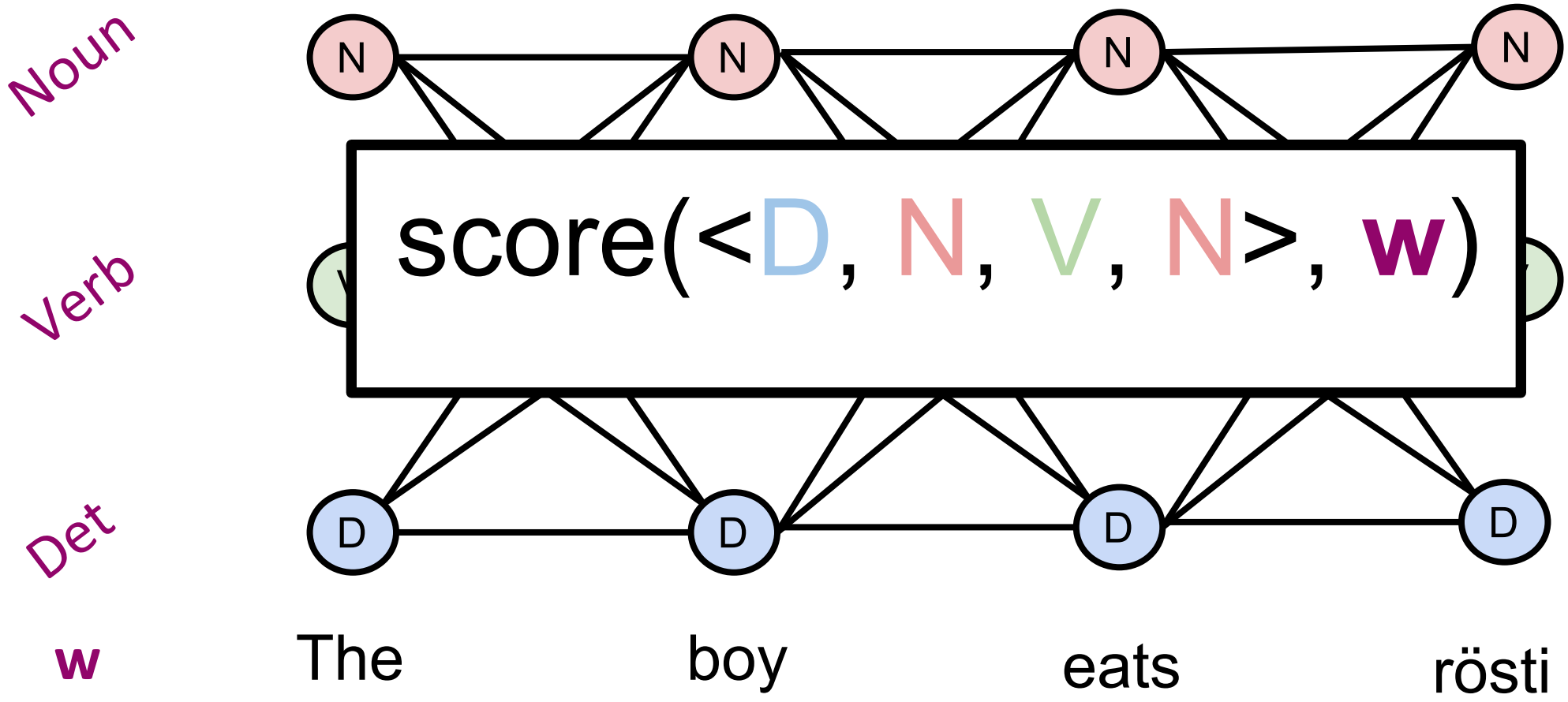


① Part-of-Speech Tagging is like Path Search in a Graph



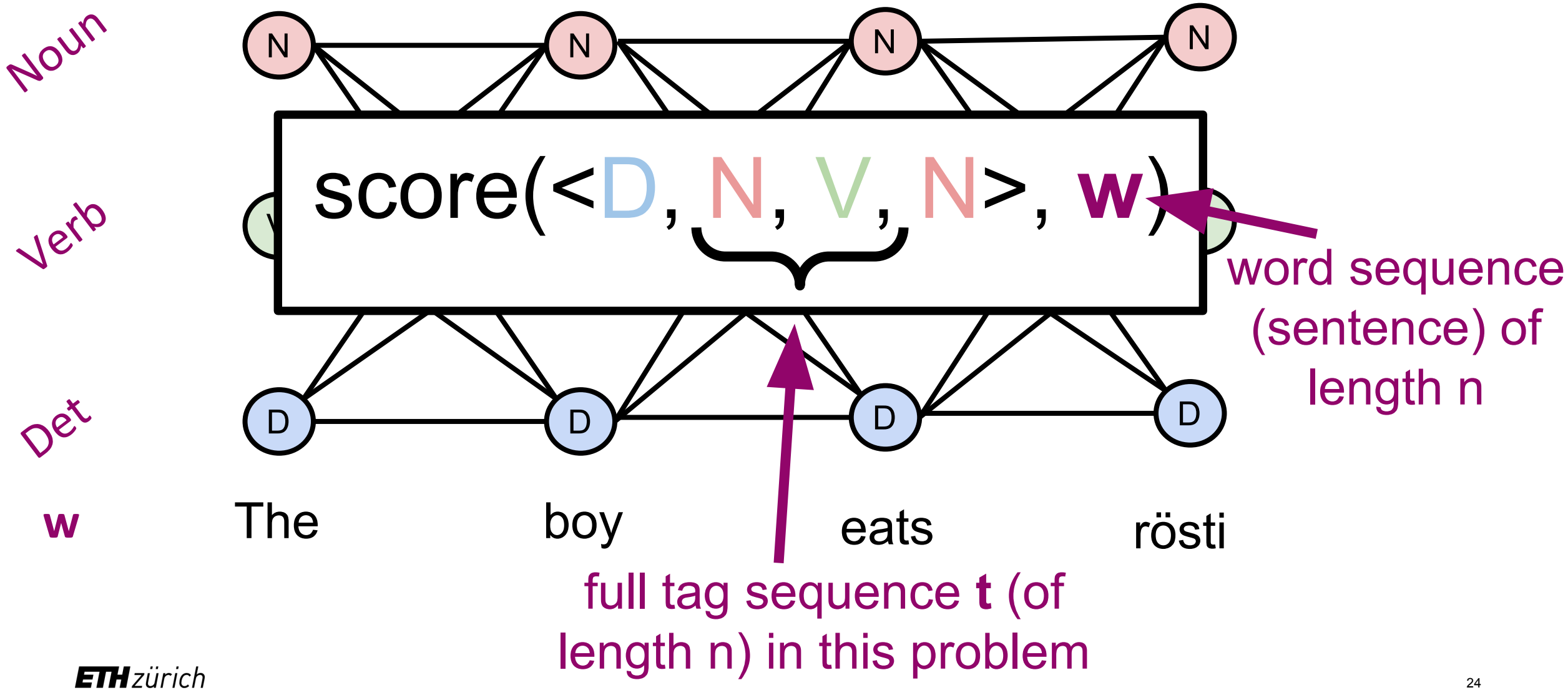
How do we score the paths?

① Part-of-Speech Tagging is like Path Search in a Graph



How do we score the paths?

① Part-of-Speech Tagging is like Path Search in a Graph



ETH zürich

Conditional Random Fields (a.k.a. Log-Linear Models on Structure)

② Log-linear Models for Part-of-Speech Tagging

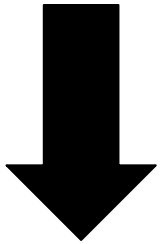
$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}$$

The **conditional random field** is a conditional probabilistic model for sequence labeling; conditional random fields are built on the logistic regression classifier.

from Eisenstein Natural Language Processing, Chapter 7

② Log-linear Models for Part-of-Speech Tagging

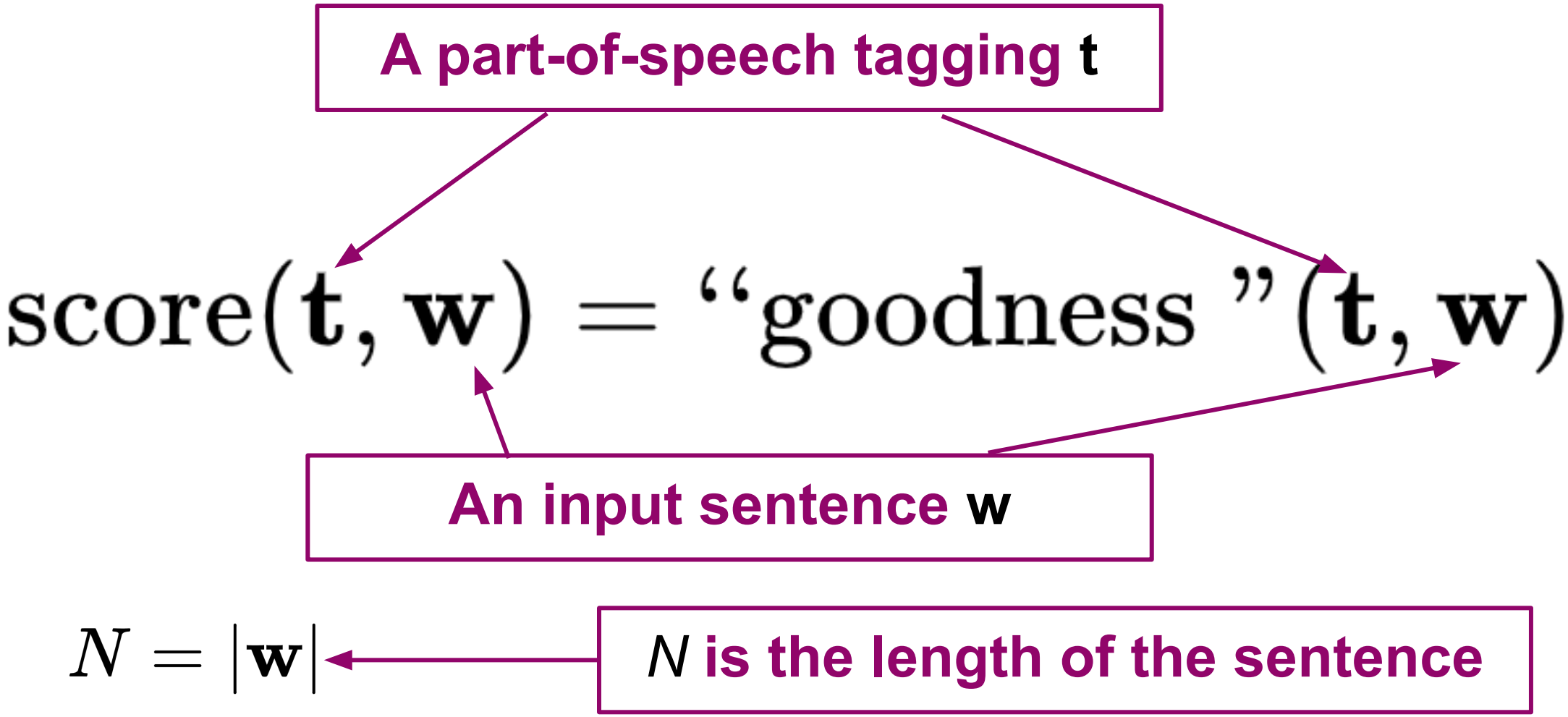
$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t},\mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}',\mathbf{w})\}}$$



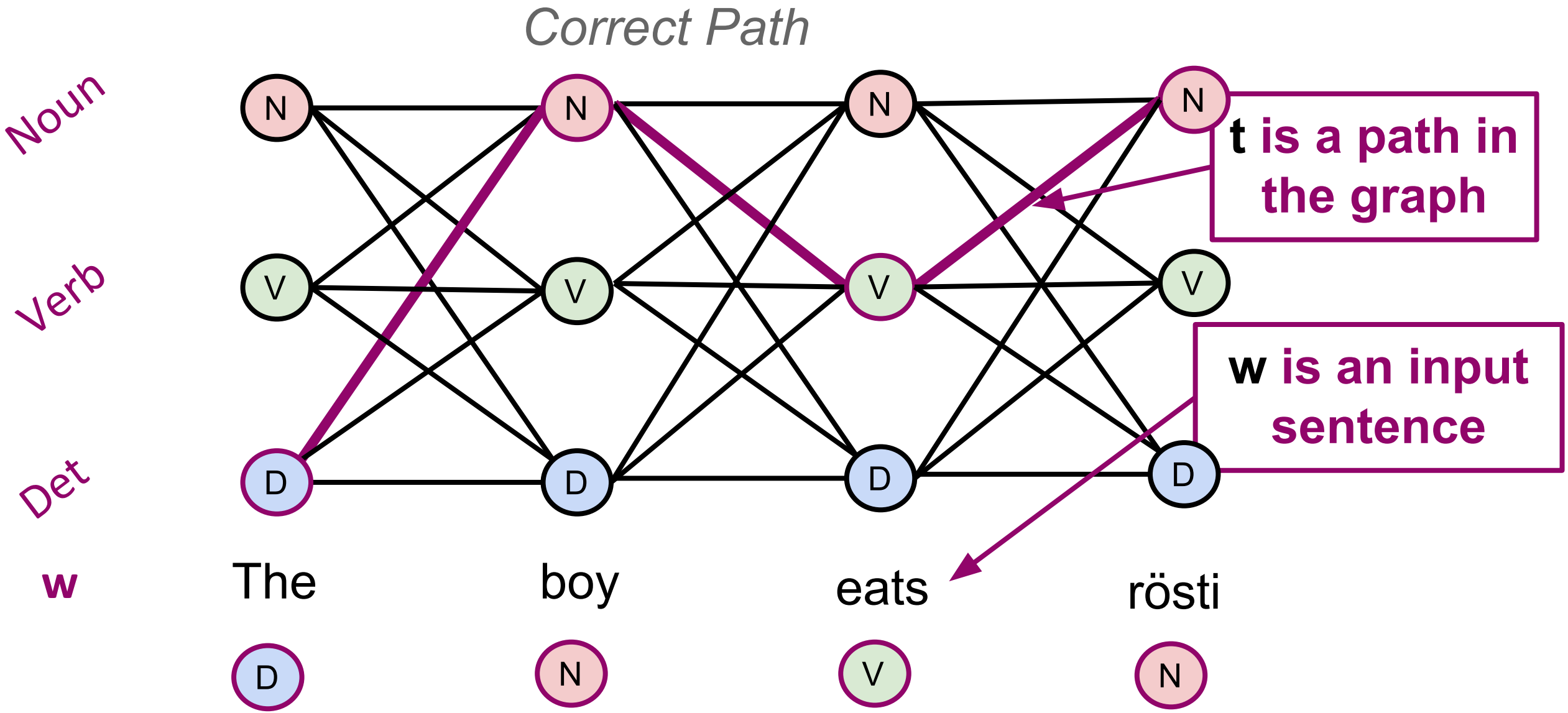
$$\text{score}(\mathbf{t}, \mathbf{w}) = \text{“goodness ”}(\mathbf{t}, \mathbf{w})$$

Higher score means $\mathbf{t}$ is a better tagging for $\mathbf{w}$

② Log-linear Models for Part-of-Speech Tagging



② Log-linear Models for Part-of-Speech Tagging



② How do we Define the Score Function?

Your score function can be any function!

Linear Function
(dot product of a weight vector and a feature function)

$$= \boldsymbol{\theta} \cdot \boldsymbol{f}(\mathbf{t}, \mathbf{w})$$

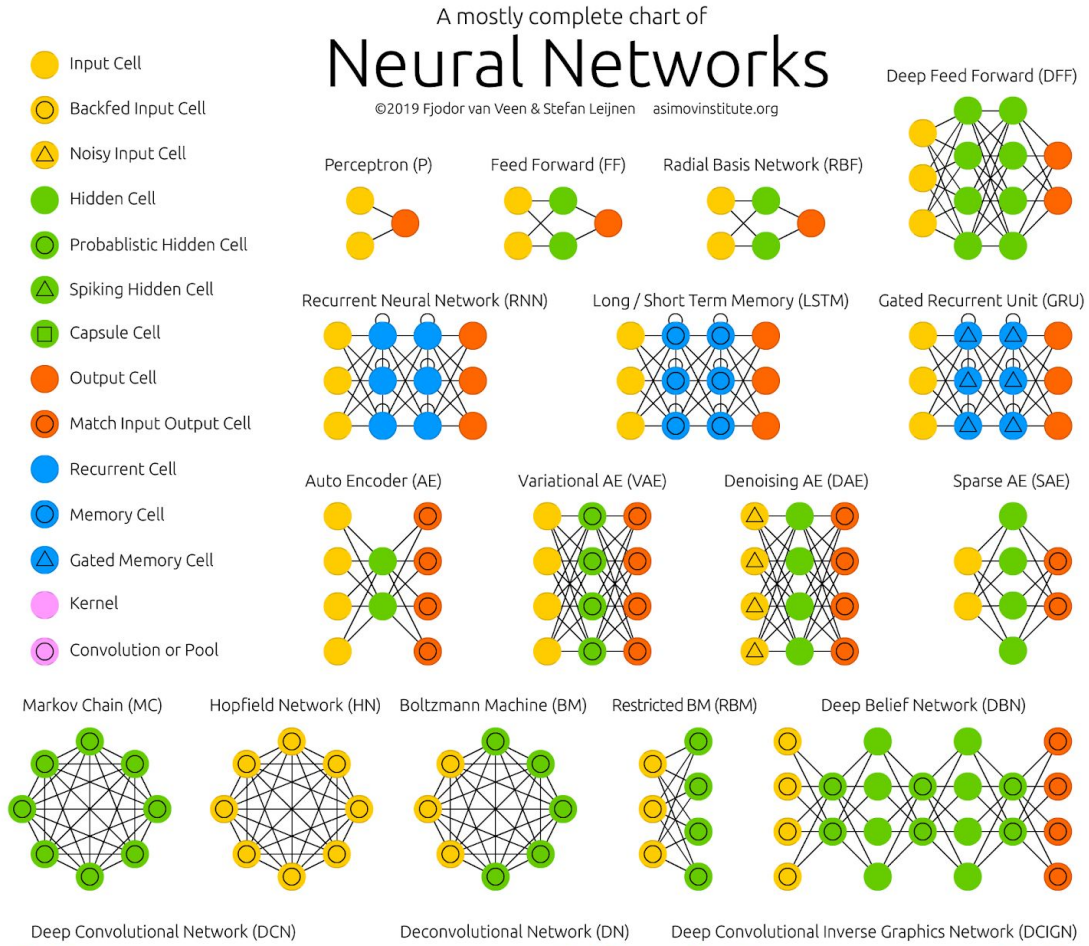
For the purposes of this lecture, it does not matter!!!

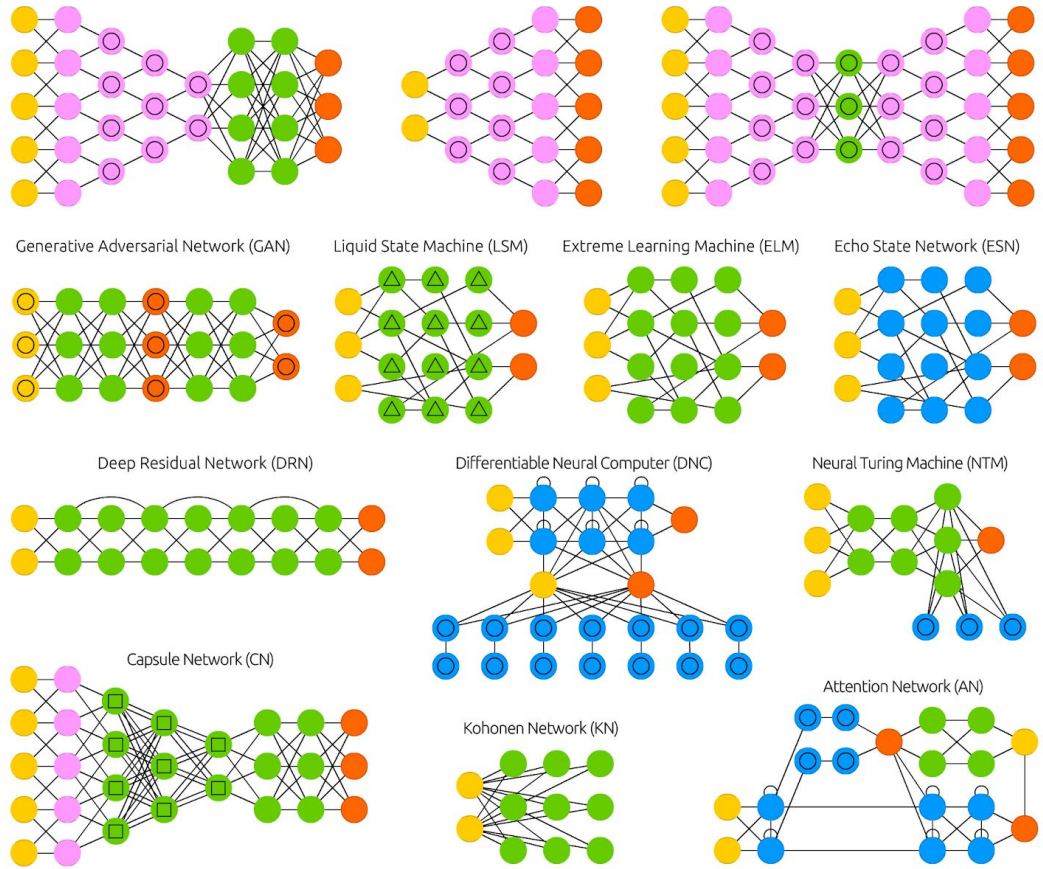
$$\text{score}(\mathbf{t}, \mathbf{w}) = \text{NN}_{\boldsymbol{\theta}}(\mathbf{t}, \mathbf{w})$$

Non-linear Function
(neural network)

② Slide Redux: Take Your Pick!

- NLP stopped engineering features and started engineering **feature extractors** (architectures)





https://www.asimovinstitute.org/neural-network-zoo/

② Conditional Random Fields = Log-Linear Models

So where's the catch?

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}$$

② Conditional Random Fields = Log-Linear Models

So where's the catch?

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\underbrace{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}}$$

- There are an exponential number of assignments

$$|\mathcal{Y}| = |\mathcal{T}^N| = |\mathcal{T}|^N$$

- Naïvely computing normalizer (a.k.a. partition function a.k.a. Zustandssumme) runs in

$$\mathcal{O}\left(|\mathcal{T}|^N\right)$$

② Conditional Random Fields = Log-Linear Models

So where's the catch?

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\underbrace{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}}$$

- There are an exponential number of assignments

$$|\mathcal{Y}| = |\mathcal{T}^N| = |\mathcal{T}|^N$$

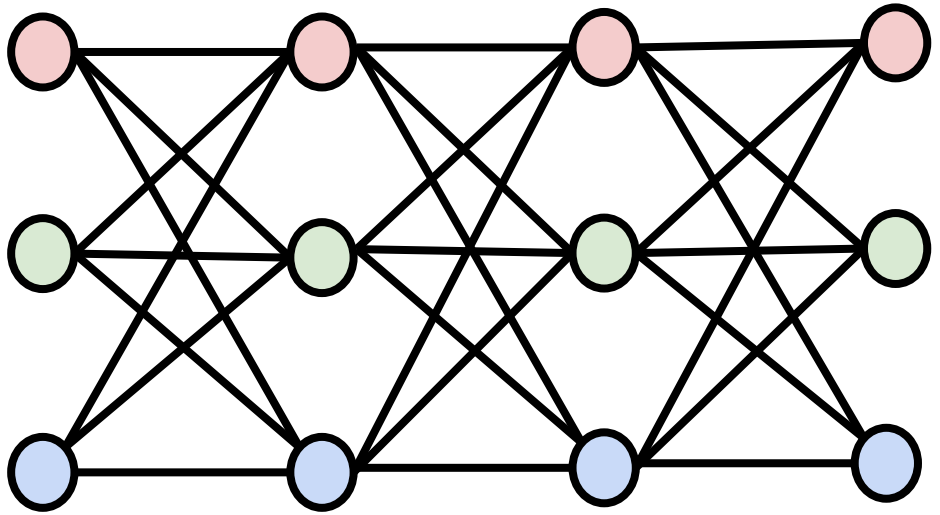
- Naïvely computing normalizer (a.k.a. partition function a.k.a. Zustandssumme) runs in

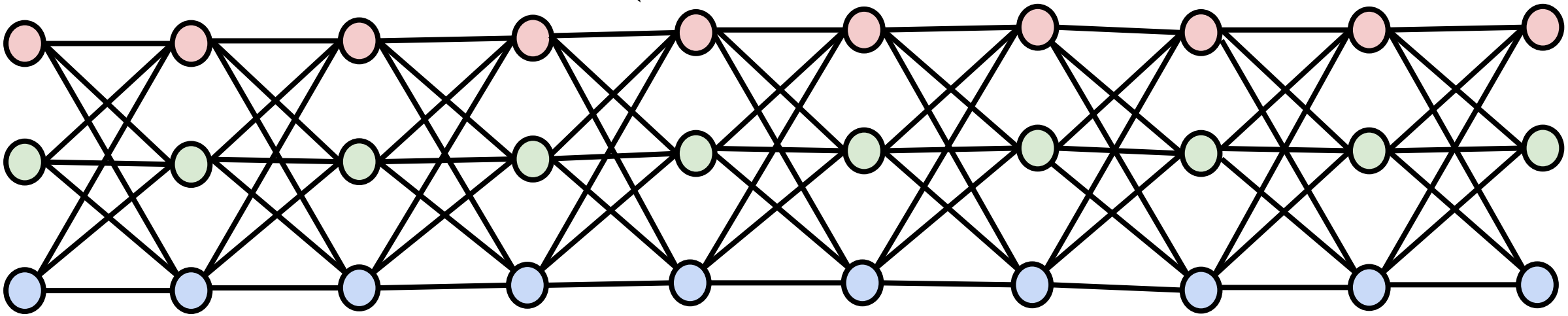
$$\mathcal{O}\left(|\mathcal{T}|^N\right)$$ 

② A Combinatorial Explosion

- In our toy example, there were 3^4 = 81 assignments
- In this example?

  3^{10} = 54049 assignments

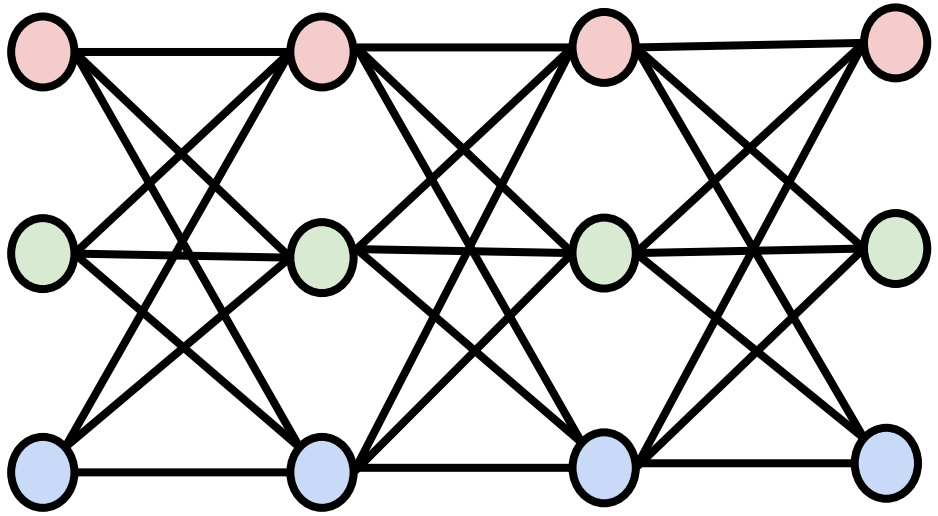




② A Combinatorial Explosion

What if we assume structure in our problem? Does it get any easier?

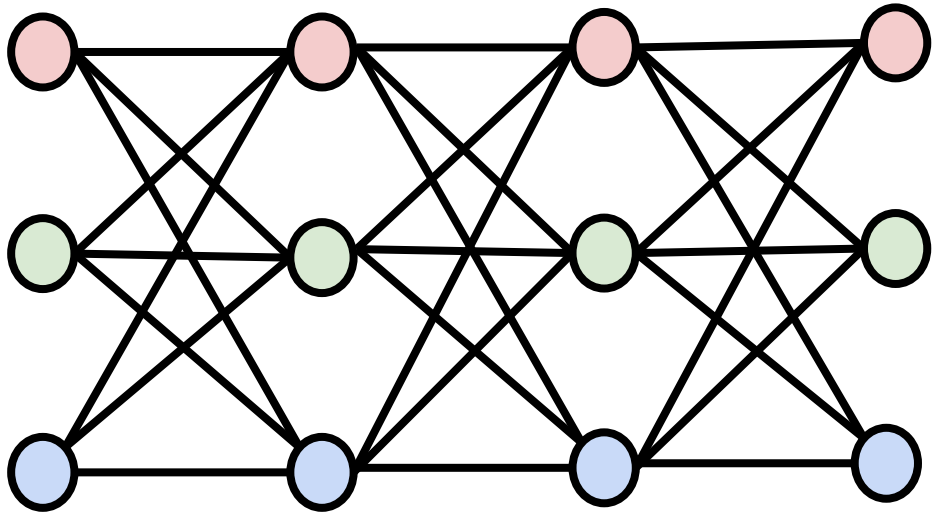
Answer: Yes. If we assume combinatorial structure, we can use combinatorial algorithms for great gains in efficiency



② A Combinatorial Explosion

What if we assume structure in our problem? Does it get any easier?

Answer: Yes. If we assume combinatorial structure, we can use combinatorial algorithms for great gains in efficiency



We've been hinting at the use of structure in this example!

② Conditional Random Fields = Log-Linear Models

- How do we encode the structure assumption?
- We ***define*** an additively decomposable score function

Quiz Question: Could we also use multiplicatively decomposable functions?

use special start symbol when n=0!

$$\text{score}(\mathbf{t}, \mathbf{w}) = \sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})$$

score of full tag $\mathbf{t}$ for word sequence $\mathbf{w}$

tag bigram

② Conditional Random Fields = Log-Linear Models

- How do we encode the structure assumption?
- We ***define*** an additively decomposable score function

score(< D, N, V, N>, **w**)

= score(< ■ , D>, **w**) + score(< D, N>, **w**) + score(< N, V>, **w**)+ score(< V, N>, **w**)

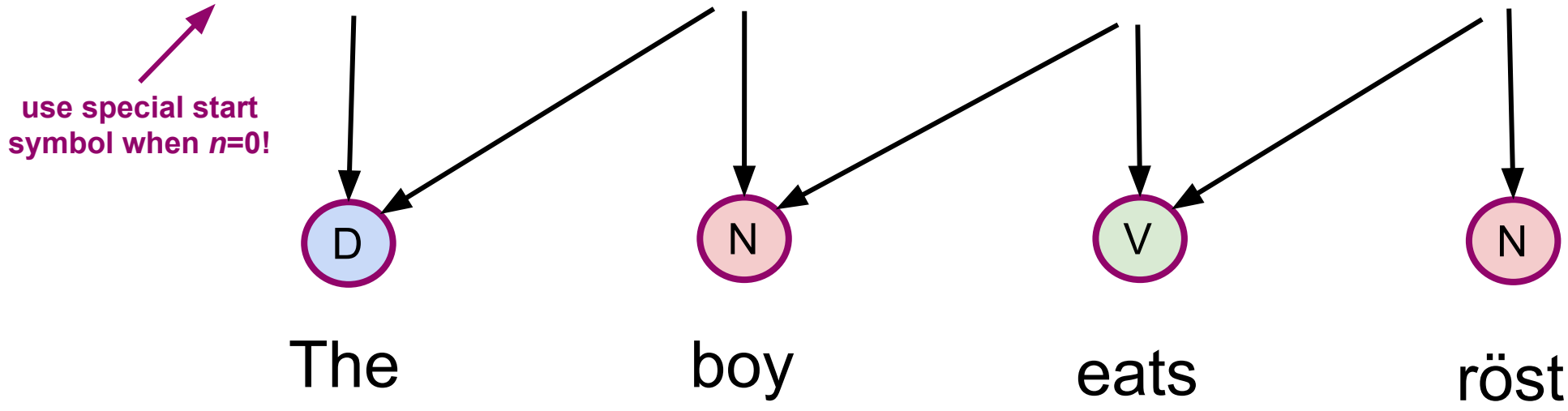
use special start symbol when *n*=0!

② Conditional Random Fields = Log-Linear Models

- How do we encode the structure assumption?
- We ***define*** an additively decomposable score function

score(< D, N, V, N>, **w**)

= score(< ■ , D>, **w**) + score(< D, N>, **w**) + score(< N, V>, **w**)+ score(< V, N>, **w**)



② Conditional Random Fields = Log-Linear Models

- How do we encode the structure assumption?
- We ***define*** an additively decomposable score function

score(< D, N, V, N>, **w**)

= score(< ■ , D>, **w**) + score(< D, N>, **w**) + score(< N, V>, **w**)+ score(< V, N>, **w**)

use special start symbol when *n*=0!

We will generally be concerned with probability models

$$p(\mathbf{t} \mid \mathbf{w}) \propto \exp\{\text{score}(\mathbf{t}, \mathbf{w})\}$$

$$= \exp\left\{\sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\right\}$$

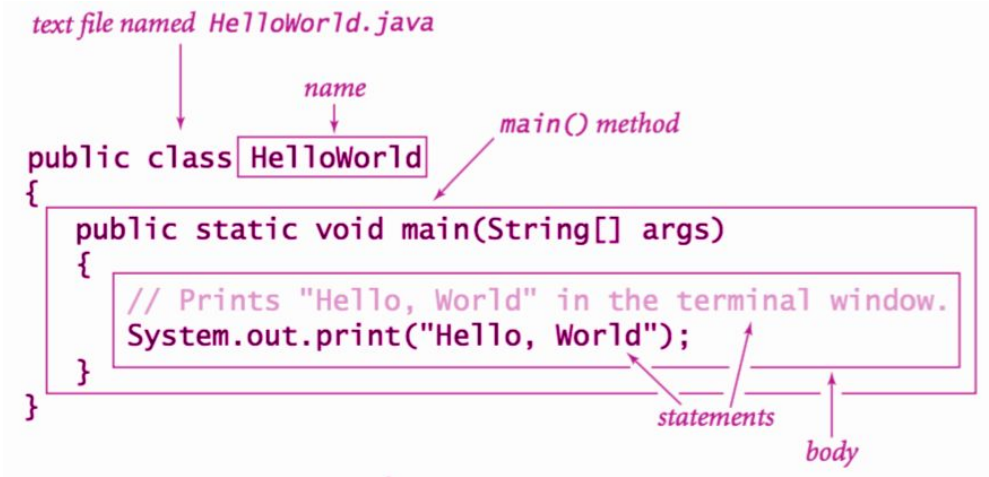
$$= \prod_{n=1}^{N} \exp\{\text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\}$$

② Score is Like a Class in Object-Oriented Programming

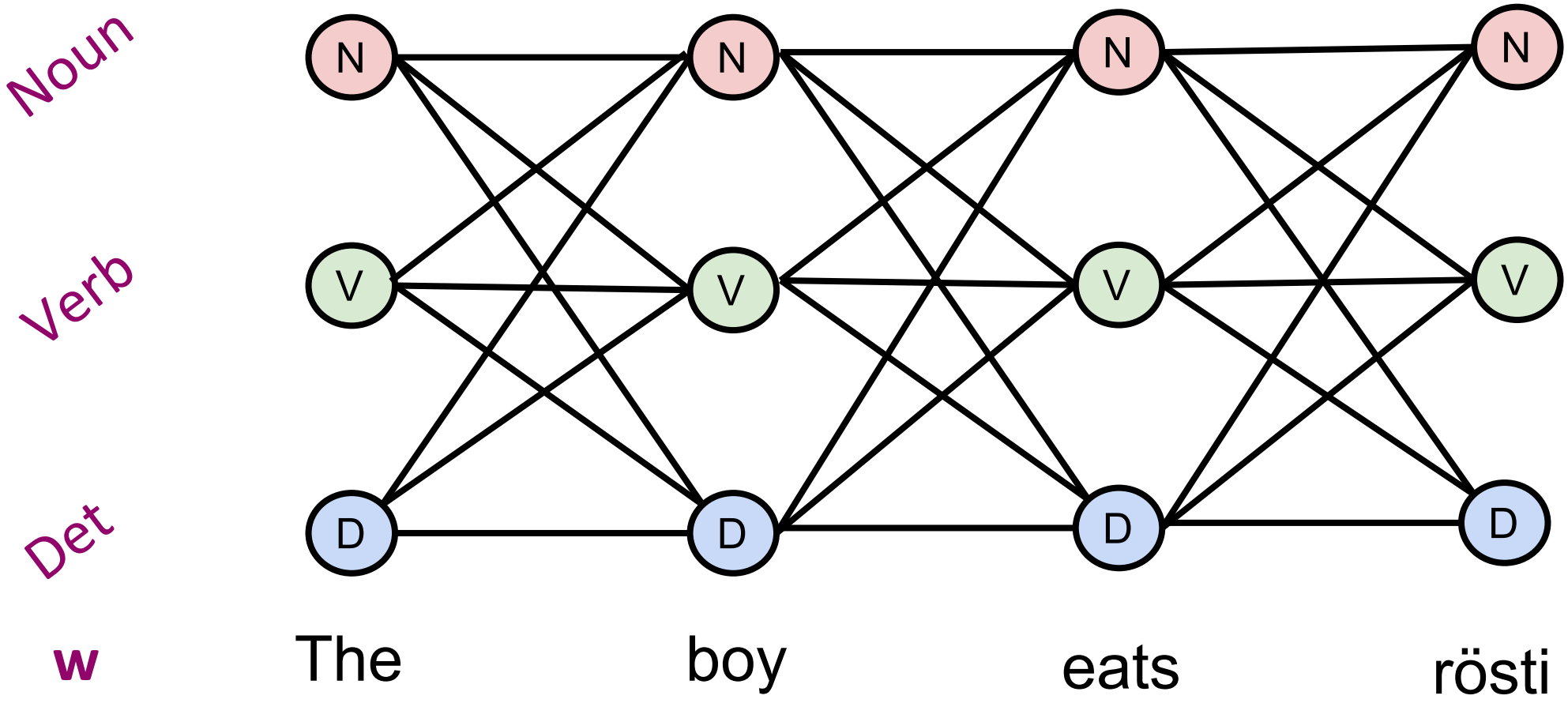
The function score should have the following properties

- Higher score means $\mathbf{t}$ and $\mathbf{w}$ are better joint sequences of tags and words
- score function decomposes additively: $\text{score}(\mathbf{t}, \mathbf{w}) = \sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})$
- If score is parameterized, e.g. a neural network, it should be differentiable in those parameters
  - Why? We will want to use backpropagation to compute the gradient so it should exist

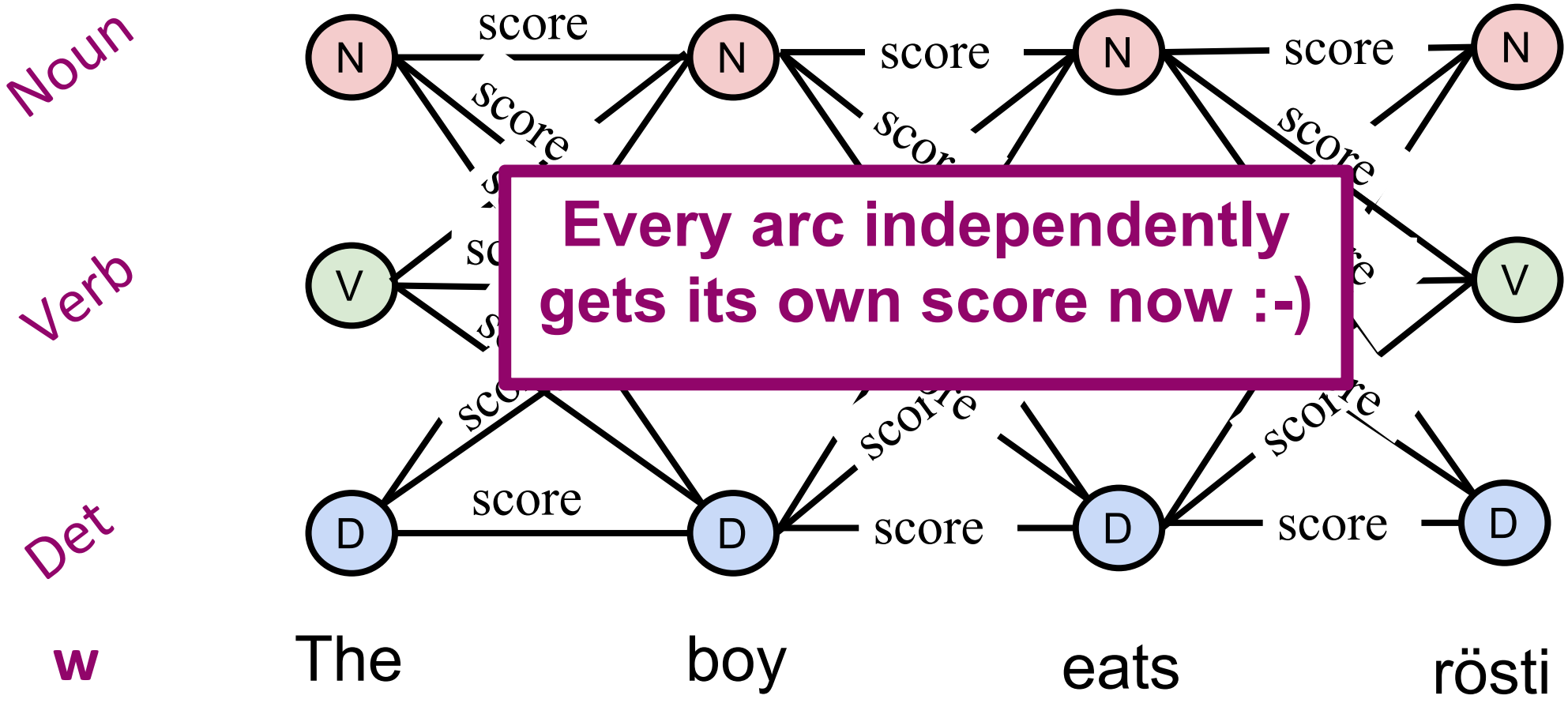
Your inference algorithms do not need to know the inside of score just like class internal members in object-oriented programming are kept hidden!



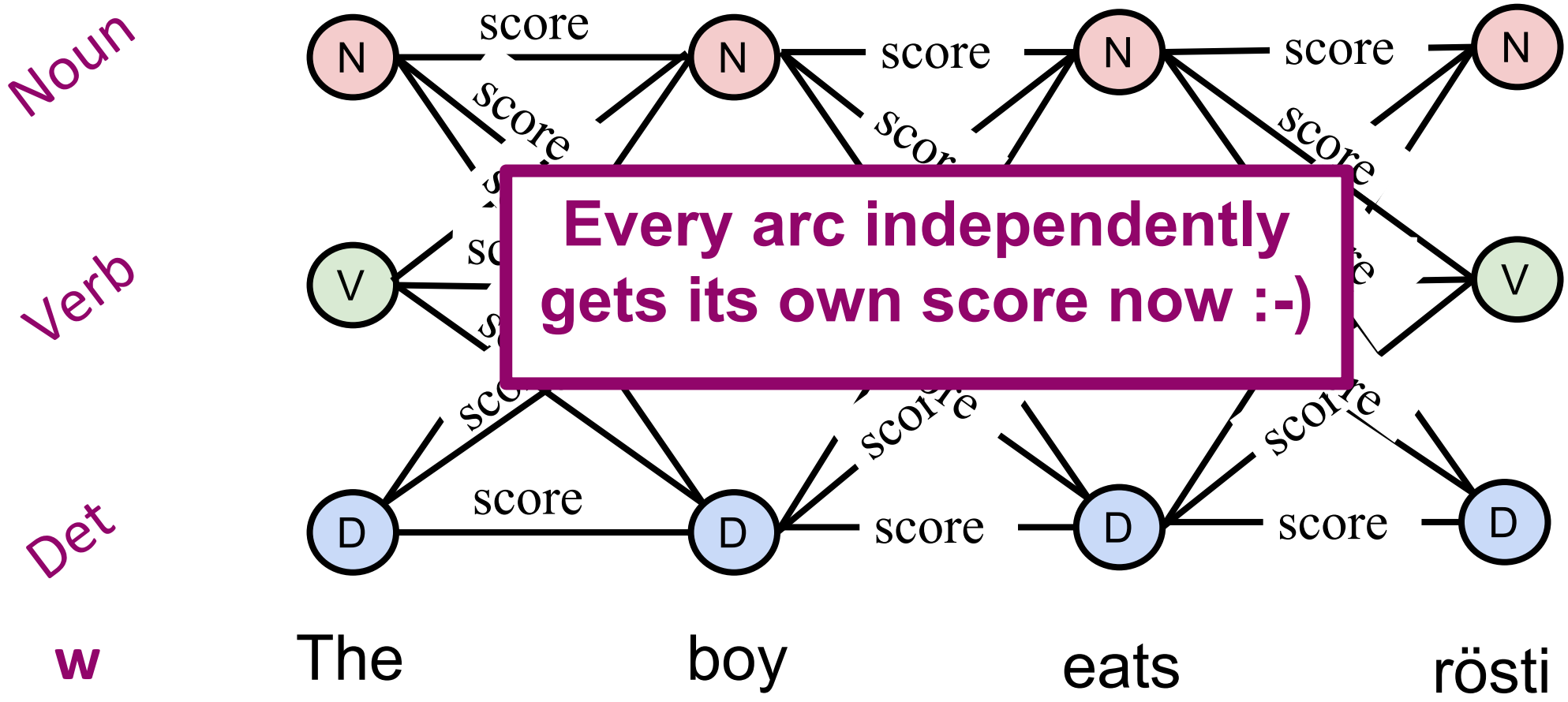
② What Does Decomposability Mean Visually?



② What Does Decomposability Mean Visually?



② You Should be Thinking: **Dynamic Programing**!



② Efficiently Computing the Normalizer

- Our original equation:

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}$$

② Efficiently Computing the Normalizer

- Our original equation:

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}$$

- Now add in our structured score function:

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\}}$$

② Efficiently Computing the Normalizer

- Our original equation:

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\text{score}(\mathbf{t}, \mathbf{w})\}}{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\text{score}(\mathbf{t}', \mathbf{w})\}}$$

- Now add in our structured score function:

$$p(\mathbf{t} \mid \mathbf{w}) = \frac{\exp\{\sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\}}{\underbrace{\sum_{\mathbf{t}' \in \mathcal{T}^N} \exp\{\sum_{n=1}^{N} \text{score}(\langle t_{n-1}, t_n \rangle, \mathbf{w})\}}}$$

Our Problem: How do we calculate this normalizing constant?

② Efficiently Computing the Normalizer

Exponential number of terms

$$\sum_{\mathbf{t} \in \mathcal{T}^N} \exp\left\{ \sum_{n=1}^{N} \text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right) \right\}$$

$$= \sum_{\mathbf{t}_{1:n} \in \mathcal{T}^N} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

$$= \sum_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \sum_{t_N \in \mathcal{T}} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

Product Distributes over Sum

$$= \sum_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \prod_{n=1}^{N-1} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\} \times \sum_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

$$= \sum_{t_1 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_0, t_1 \rangle, \mathbf{w}\right)\} \times \sum_{t_2 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_1, t_2 \rangle, \mathbf{w}\right)\} \times \cdots \times \sum_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

use special start symbol when *n*=0!

Linear number of terms

② Efficiently Computing the Normalizer

A Simple Dynamic Program

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

Where our goal is to calculate normalizing constant $Z = \beta(\mathbf{w}, t_0)$

② Efficiently Computing the Normalizer

A Simple Dynamic Program

What does this remind you of?

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1}\rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

Where our goal is to calculate normalizing constant $Z = \beta(\mathbf{w}, t_0)$

② Efficiently Computing the Normalizer

A Simple Dynamic Program

What does this remind you of?

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1}\rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

Where our goal is to calculate normalizing constant $Z = \beta(\mathbf{w}, t_0)$

② Efficiently Computing the Normalizer

A Simple Dynamic Program

What does this remind you of?

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1}\rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

Where our goal is to calculate normalizing constant $Z = \beta(\mathbf{w}, t_0)$

2 Efficiently Computing the Normalizer

A Simple Dynamic Program

What does this remind you of?

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

back-propagate$(f, \mathbf{x} \in \mathbb{R}^n)$

$\mathbf{v} \leftarrow$ **forward-propagate**$(f, \mathbf{x})$

$$\frac{\partial f}{\partial v_n} = 0, \ \forall n \in \{1, \ldots, N\}$$

for $n = N, \ldots, 1:$

$$\frac{\partial f}{\partial v_n} = \sum_{j : n \in \text{Pa}(j)} \frac{\partial f}{\partial v_j} \frac{\partial}{\partial v_n} p_j(\langle v_{\text{Pa}(j)} \rangle)$$

return $\left[\frac{\partial f}{\partial v_1}, \ldots, \frac{\partial f}{\partial v_N}\right]$

Backpropagation!

② Efficiently Computing the Normalizer

A Simple Dynamic Program

What does this remind you of?

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

Shortest-Path-esque Dynamic Programs!

$$\textbf{back-propagate}(f, \mathbf{x} \in \mathbb{R}^n)$$

$$\mathbf{v} \leftarrow \textbf{forward-propagate}(f, \mathbf{x})$$

$$\frac{\partial f}{\partial v_n} = 0, \ \forall n \in \{1, \ldots, N\}$$

$$\textbf{for } n = N, \ldots, 1:$$

$$\frac{\partial f}{\partial v_n} = \sum_{j: n \in \text{Pa}(j)} \frac{\partial f}{\partial v_j} \frac{\partial}{\partial v_n} p_j(\langle v_{\text{Pa}(j)} \rangle)$$

$$\textbf{return } \left[\frac{\partial f}{\partial v_1}, \ldots, \frac{\partial f}{\partial v_N} \right]$$

Backpropagation!

② The Decoding Problem

How can we find the highest-scoring tagging $\mathbf{t}$ for an input sentence $\mathbf{w}$?

- Say we are given a CRF with a scoring function
- We want to compute the following argmax:

$$\mathbf{t}^\star \leftarrow \underset{\mathbf{t}' \in \mathcal{T}^N}{\operatorname{argmax}} \operatorname{score}(\mathbf{t}', \mathbf{w})$$

- The general idea is to compute the max (the expression below) and lay ***backpointers*** to get the argmax

$$\max_{\mathbf{t}' \in \mathcal{T}^N} \operatorname{score}(\mathbf{t}', \mathbf{w})$$

② How do We Efficiently Compute the Max?

How can we find the highest-scoring tagging for an input sentence **w**?

- Times distributes over max (for non-negative values) as it does over sum
  - The same derivation for our normalizing constant holds for finding the highest-scoring path!

A(nother) Simple Dynamic Program (Viterbi):

$$\gamma(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\gamma(\mathbf{w}, t_n) \leftarrow \max_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \gamma(\mathbf{w}, t_{n+1})$$

② How do We Efficiently Compute the Max?

Viterbi Algorithm

$$\gamma(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\gamma(\mathbf{w}, t_n) \leftarrow \max_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \gamma(\mathbf{w}, t_{n+1})$$

Backward Algorithm

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \sum_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

② How do We Efficiently Compute the Max?

Viterbi Algorithm

$$\gamma(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\gamma(\mathbf{w}, t_n) \leftarrow \boxed{\max_{t_{n+1} \in \mathcal{T}}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \gamma(\mathbf{w}, t_{n+1})$$

Backward Algorithm

$$\beta(\mathbf{w}, t_N) \leftarrow 1$$

$$\textbf{for } n \leftarrow N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) \leftarrow \boxed{\sum_{t_{n+1} \in \mathcal{T}}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \times \beta(\mathbf{w}, t_{n+1})$$

② What do Backward and Viterbi Have in Common?

- The computation of the normalizing constant and highest-scoring paths are shortest-path problems in the abstract: Can we give this a more ***general*** framing? I.e., how do we go from:

Many Problems, Many Algorithms → *Many Problems,* ***One*** *Algorithm*

- **Solution**: Parameterizing our dynamic programs by a ***semiring*** will allow us to concisely represent our family of algorithms!

ETH zürich

Semirings

3 Semirings

A semiring is an algebraic structure

Definition 2. A *semiring* is a 5-tuple $R = (A, \oplus, \otimes, \overline{0}, \overline{1})$ such that

1. $(A, \oplus, \overline{0})$ is a commutative monoid.
2. $(A, \otimes, \overline{1})$ is a monoid.
3. $\otimes$ distributes over $\oplus$: for all a, b, c in A,

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),$$
$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).$$

4. $\overline{0}$ is an *annihilator* for $\otimes$: for all a in A, $\overline{0} \otimes a = a \otimes \overline{0} = \overline{0}$.

3 Semirings

A semiring is an algebraic structure

Definition 2. A *semiring* is a 5-tuple $R = (A, \oplus, \otimes, \overline{0}, \overline{1})$ such that

1. $(A, \oplus, \overline{0})$ is a commutative monoid.
2. $(A, \otimes, \overline{1})$ is a monoid.
3. $\otimes$ distributes over $\oplus$: for all a, b, c in A,

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),$$
$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).$$

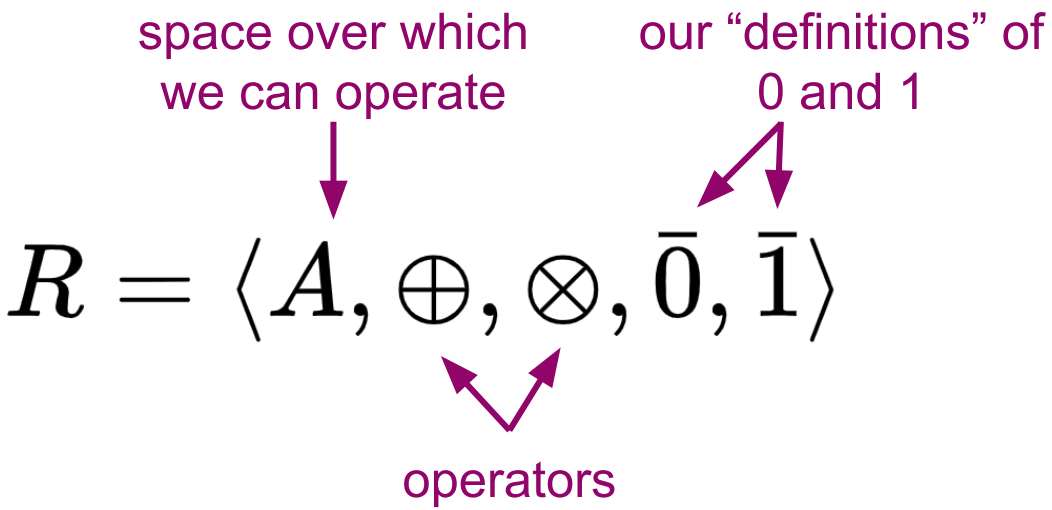
4. $\overline{0}$ is an *annihilator* for $\otimes$: for all a in A, $\overline{0} \otimes a = a \otimes \overline{0} = \overline{0}$.

The above is tedious! So, what's the intuition?

3 Semirings

$$R = \langle A, \oplus, \otimes, \bar{0}, \bar{1} \rangle$$

3 Semirings


space over which we can operate
our “definitions” of 0 and 1
$R = \langle A, \oplus, \otimes, \bar{0}, \bar{1} \rangle$
operators

3 Semirings

space over which we can operate

our “definitions” of 0 and 1

$$R = \langle A, \oplus, \otimes, \bar{0}, \bar{1} \rangle$$

operators

A simple example:

$$\langle \mathbb{R}_+, +, \times, 0, 1 \rangle$$

This is a semiring because $\forall\, a, b, c \in \mathbb{R}_+$:

1. $(a + b) + c = a + (b + c)$
$0 + a = a + 0 = a$
$a + b = b + a$

2. $(a \times b) \times c = a \times (b \times c)$
$1 \times a = a \times 1 = a$

3. $a \times (b + c) = (a \times b) + (a \times c)$
$(a + b) \times c = (a \times c) + (b \times c)$

4. $0 \times a = a \times 0 = 0$

③ Semirings

space over which we can operate

our "definitions" of 0 and 1

$$R = \langle A, \oplus, \otimes, \bar{0}, \bar{1} \rangle$$

A simple

This is a

Dynamic programming is just the distributive law on steroids!!!

1. $(a+b)+c = a+(b+c)$
2. $(a \times b) \times c = a \times (b \times c)$
3. $a \times (b+c) = (a \times b) + (a \times c)$

4. $0 \times a = a \times 0 = 0$

Key Property: $\otimes$ distributes over $\oplus$ on the set A

3 Semirings

Semiring	Set	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	intuition/application
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1	logical deduction, recognition
Viterbi	$[0, 1]$	max	$\times$	0	1	prob. of the best derivation
Inside	$\mathbb{R}^+ \cup \{+\infty\}$	$+$	$\times$	0	1	prob. of a string
Real	$\mathbb{R} \cup \{+\infty\}$	min	$+$	$+\infty$	0	shortest-distance
Tropical	$\mathbb{R}^+ \cup \{+\infty\}$	min	$+$	$+\infty$	0	with non-negative weights
Counting	$\mathbb{N}$	$+$	$\times$	0	1	number of paths

Table 2: Examples of semirings

https://www.aclweb.org/anthology/C08-5001.pdf

3 Semirings

Semiring	Set	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	intuition/application
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1	logical deduction, recognition
Viterbi	$[0,1]$	max	$\times$	0	1	prob. of the best derivation
Inside	$\mathbb{R}^+ \cup \{+\infty\}$	$+$	$\times$	0	1	prob. of a string
Real	$\mathbb{R} \cup \{+\infty\}$	min	$+$	$+\infty$	0	shortest-distance
Tropical	$\mathbb{R}^+ \cup \{+\infty\}$	min	$+$	$+\infty$	0	with non-negative weights
Counting	$\mathbb{N}$	$+$	$\times$	0	1	number of paths

Table 2: Examples of semirings

https://www.aclweb.org/anthology/C08-5001.pdf

③ Semirings in Action: Backwards Proof of Correctness

Exponential number of terms

$$\sum_{\mathbf{t}_{1:n} \in \mathcal{T}^N} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

Product Distributes over Sum

$$= \sum_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \sum_{t_N \in \mathcal{T}} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

$$= \sum_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \prod_{n=1}^{N-1} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\} \times \sum_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

$$= \sum_{t_1 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_0, t_1 \rangle, \mathbf{w}\right)\} \times \sum_{t_2 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_1, t_2 \rangle, \mathbf{w}\right)\} \times \cdots \times \sum_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

use special start symbol when *n*=0!

Linear number of terms

③ Semirings in Action: Backwards Proof of Correctness

$$\max_{\mathbf{t}_{1:N} \in \mathcal{T}^N} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

Exponential number of terms

Product Distributes over max

$$= \max_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \max_{t_N \in \mathcal{T}} \prod_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

$$= \max_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \prod_{n=1}^{N-1} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\} \times \max_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

$$= \max_{t_1 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_0, t_1 \rangle, \mathbf{w}\right)\} \times \max_{t_2 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_1, t_2 \rangle, \mathbf{w}\right)\} \times \cdots \times \max_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

use special start symbol when *n*=0!

Linear number of terms

③ Semirings in Action: Distributivity is All You Need

$$\bigoplus_{\mathbf{t}_{1:N} \in \mathcal{T}^N} \bigotimes_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

Exponential number of terms

⊗ Distributes over ⊕

$$= \bigoplus_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \bigoplus_{t_N \in \mathcal{T}} \bigotimes_{n=1}^{N} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\}$$

$$= \bigoplus_{\mathbf{t}_{1:N-1} \in \mathcal{T}^{N-1}} \bigotimes_{n=1}^{N-1} \exp\{\text{score}\left(\langle t_{n-1}, t_n \rangle, \mathbf{w}\right)\} \otimes \bigoplus_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

$$= \bigoplus_{t_1 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_0, t_1 \rangle, \mathbf{w}\right)\} \otimes \bigoplus_{t_2 \in \mathcal{T}} \exp\{\text{score}\left(\langle t_1, t_2 \rangle, \mathbf{w}\right)\} \otimes \cdots \otimes \bigoplus_{t_N \in \mathcal{T}} \exp\{\text{score}\left(\langle t_{N-1}, t_N \rangle, \mathbf{w}\right)\}$$

use special start symbol when *n*=0!

Linear number of terms

3 Semirings in Action

- **Backward Algorithm**: How to Find the Normalizing Constant *Z*

- Choose the semiring $R = \langle \mathbb{R}^+, \max, \times, \mathbf{0}, \mathbf{1} \rangle$ and compute:

$$\beta(\mathbf{w}, t_N) = \mathbf{1}$$
$$\textbf{for } n = N-1, \ldots, 0:$$
$$\beta(\mathbf{w}, t_n) = \bigoplus_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \otimes \beta(\mathbf{w}, t_{n+1})$$

- Correctness is guaranteed because *R* is a semiring!

③ Semirings in Action

- **Viterbi Algorithm**: How to Find the Highest-Scoring Path

- Choose the semiring $R = \langle \mathbb{R}^+, \max, \times, \mathbf{0}, \mathbf{1} \rangle$ and compute:

$$\beta(\mathbf{w}, t_N) = \mathbf{1}$$
$$\textbf{for } n = N-1, \ldots, 0:$$
$$\beta(\mathbf{w}, t_n) = \bigoplus_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1} \rangle, \mathbf{w})\} \otimes \beta(\mathbf{w}, t_{n+1})$$

- Correctness is guaranteed because *R* is a semiring!

③ One Algorithm to Rule Them All!

- We prove the correctness of ***one generic algorithm***
  - The proof of correctness for the special cases emerges automatically

$$\beta(\mathbf{w}, t_N) = \mathbf{1}$$

$$\textbf{for } n = N-1, \ldots, 0:$$

$$\beta(\mathbf{w}, t_n) = \bigoplus_{t_{n+1} \in \mathcal{T}} \exp\{\text{score}\,\langle t_n, t_{n+1}\rangle, \mathbf{w})\} \otimes \beta(\mathbf{w}, t_{n+1})$$

- The intuition of why dynamic programming works is clear:
  - It's all about distributivity
- Semirings are an answer to the question: How general can we go such that dynamic programming still works?

3 Shortest-Path Algorithms

- Beyond Viterbi, there are many shortest-path algorithms; it's a very well studied problem in CS
  - Dijkstra, Bellman–Ford, Floyd–Warshall, *inter alia*

Dijkstra

Bellman **Ford**

Floyd **Warshall**

4 Shortest-Path Algorithms

- Can we semiring-ify any shortest-path algorithm?
  - The first-order approximation is yes: You can semiring-ify any of the shortest-path algorithms you learned in algorithms (previous slide)

- For example, you can do do a prioritized decoding algorithm for a CRF with Dijkstra's
- You can also compute a normalizer Z with Dijkstra's if you change the semiring!
  - **Quiz Question:** Why might this not be a good idea?

ETH zürich

CRFs as a Softmax and the Structured Perceptron

4 Parameter Estimation in a CRF

- We talked about algorithms for computing *Z* and computing the one-best path in a CRF
- How do we estimate the parameters?
- Generally, we maximize the following log-likelihood

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)}) - \log \sum_{\mathbf{t}' \in \mathcal{T}^N} \exp \text{score}(\mathbf{t}', \mathbf{w}^{(i)}))$$

In part 2, we showed this sum with an exponentially (in |x|) number of summands can be computed in linear time!

4 Parameter Estimation in a CRF

- How do we compute the gradient of the CRF's log-likelihood?
  - ***Backpropagation***!
- How do we estimate its parameters?
  - Gradient descent using the gradient found with ***backpropagation***

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)}) - \log \sum_{\mathbf{t}' \in \mathcal{T}^N} \exp \text{score}(\mathbf{t}', \mathbf{w}^{(i)}))$$

In part 2, we showed this sum with an exponentially (in |x|) number of summands can be computed in linear time!

④ CRFs are Just a Big Structured Softmax

You guys saw the second term in Lecture 3: It's a softmax!

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)}) - \log \sum_{\mathbf{t}' \in \mathcal{T}^N} \exp \text{score}(\mathbf{t}', \mathbf{w}^{(i)}))$$

It doesn't matter that it has a lot of summands!

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)})) - T \log \sum_{\mathbf{t}' \in \mathcal{T}^N} \exp \frac{\text{score}(\mathbf{t}', \mathbf{w}^{(i)})}{T}$$

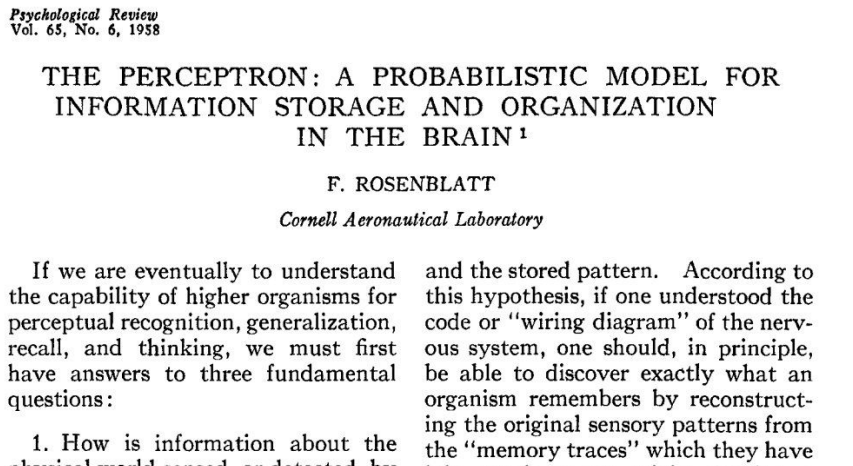
So, let's insert a temperature parameter $T > 0$ as we did in Lecture 2

④ Slide Redux: What is a Perceptron?

What is a perceptron?

- The perceptron is like a happy meal:
  - It's a lot of things in one and it comes as ***a package deal***
- Specifically, the perceptron (Rosenblatt 1958) is
  - A log-linear model like we discussed in Lecture #3
    - as the temperature $T \rightarrow 0$ (recall this means the softmax becomes a true max)
    - trained with stochastic gradient descent with a minibatch size of 1 (called the **perceptron update rule**)

- **The TL;DR**: The perceptron is a log-linear model with a specific setting of the hyperparameters
  - Fancy name for a special case of a log-linear model!



Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

1. How is information about the

and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have

4 CRFs are Just a Big Softmax

Viterbi

- So, what are we going to do?
- Consider the following objective:

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)})) - T \log \sum_{\mathbf{t}' \in \mathcal{T}^N} \exp \frac{\text{score}(\mathbf{t}', \mathbf{w}^{(i)})}{T}$$

in the limit as $T \rightarrow 0$

- We end up with:

In part 2, we showed this max with an exponentially (in |x|) number of aggregands can be computed in linear time!

$$\sum_{i=1}^{I} (\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)}) - \max_{\mathbf{t}' \in \mathcal{T}^N} \text{score}(\mathbf{t}', \mathbf{w}^{(i)}))$$

4 The Structured Perceptron

- When we take a CRF and anneal it, we get a model called the **structured perceptron** in the limit as $T \to 0$
- What does structured mean again?
  - It means we have a big output space
- We can optimize the the objective below with ***sub***gradient descent
  - Why ***sub***gradient? Because $\max$ is not differentiable everywhere
  - When score is linear the objective is convex and we get convergence
  - When the minibatch size is 1 and the learning rate is 1 we get the **structured perceptron update rule**

$$\sum_{i=1}^{I} \left(\text{score}(\mathbf{t}^{(i)}, \mathbf{w}^{(i)}) - \max_{\mathbf{t}' \in \mathcal{T}^N} \text{score}(\mathbf{t}', \mathbf{w}^{(i)}) \right)$$

4 The Structured Perceptron

- Invented by ACL fellow Michael Collins
- Near and dear to the hearts of NLPers because it appeared in our literature first
  - EMNLP 2002
- Simple algorithm that adds a slight modification to the perceptron algorithm in order to accomodate output space in structured prediction setting

Proceedings of the Conference on Em
Language Processing (EMNLP), Philade
Association for

Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms

Michael Collins
AT&T Labs-Research, Florham Park, New Jersey.
mcollins@research.att.com

Abstract

We describe new algorithms for training tagging models, as an alternative to maximum-entropy models or conditional random fields (CRFs). The algorithms rely on Viterbi decoding of training examples, combined with simple additive updates. We describe theory justifying the algorithms through a modification of the proof of convergence of the perceptron algorithm for classification problems. We give experimental results on part-of-speech tagging and base noun phrase chunking, in both cases showing improvements over results for a maximum-entropy tagger.

1 Introduction

Maximum-entropy (ME) models are justifiably a very popular choice for tagging problems in

in (Freund & Schapire 99). These algorithms have been shown by (Freund & Schapire 99) to be competitive with modern learning algorithms such as support vector machines; however, they have previously been applied mainly to classification tasks, and it is not entirely clear how the algorithms can be carried across to NLP tasks such as tagging or parsing.

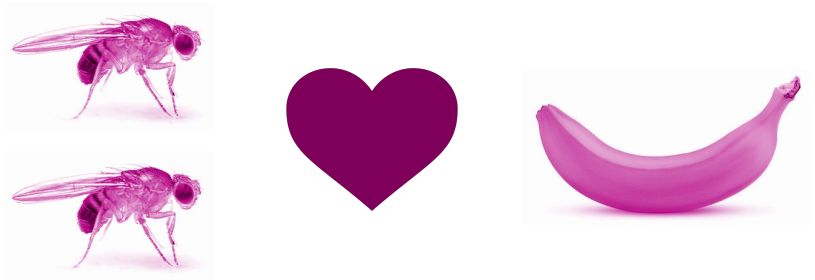
This paper describes variants of the perceptron algorithm for tagging problems. The algorithms rely on Viterbi decoding of training examples, combined with simple additive updates. We describe theory justifying the algorithm through a modification of the proof of convergence of the perceptron algorithm for classification problems. We give experimental results on part-of-speech tagging and base noun phrase chunking, in both cases showing improvements over results for a maximum-entropy tagger (a 11.9% relative reduction in error for POS tagging, a 5.1% relative reduction in error for NP chunking). Although we concentrate on tagging

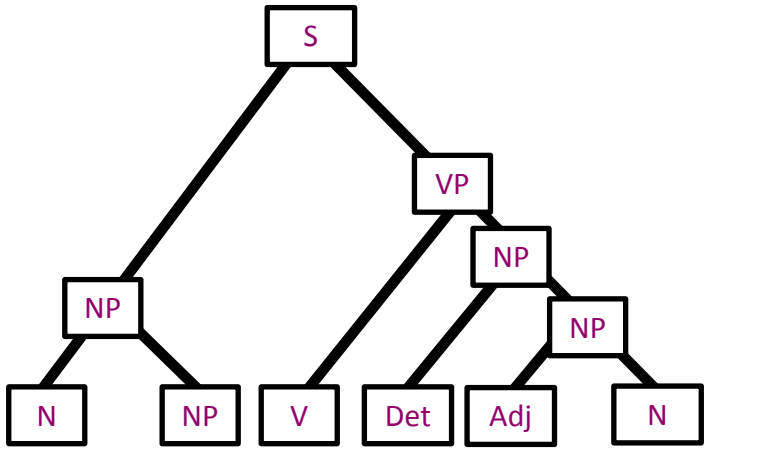
ETH zürich

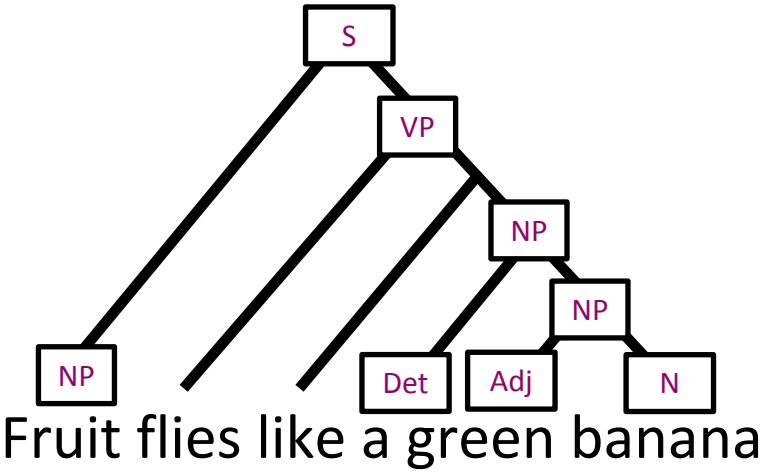
Sneak Preview of Next Lecture

Context-Free Parsing with CKY

- We provide an overview of the constituency parsing problem
- We will teach CKY (another dynamic program) for context-free parsing







ETH zürich

Fin