

IN2110: Språkteknologiske metoder

Syntaktisk struktur

Stephan Oepen

Språkteknologigruppen (LTG)

26. mars 2019





- ▶ Short recap: The Viterbi algorithm
 - ▶ Filling in the Viterbi trellis
 - ▶ Recursive problem definition



- ▶ Short recap: The Viterbi algorithm
 - ▶ Filling in the Viterbi trellis
 - ▶ Recursive problem definition
- ▶ Move on to **grammatical structure**
 - ▶ The case for structure
 - ▶ Context-free grammars
 - ▶ Treebanks
 - ▶ Probability Estimation



- ▶ Short recap: The Viterbi algorithm
 - ▶ Filling in the Viterbi trellis
 - ▶ Recursive problem definition
- ▶ Move on to **grammatical structure**
 - ▶ The case for structure
 - ▶ Context-free grammars
 - ▶ Treebanks
 - ▶ Probability Estimation
- ▶ Quick review of anonymous questionnaire



Missing records of weather in Baltimore, MD, for Summer 2007

- ▶ Jason likes to eat ice cream.
- ▶ He records his daily ice cream consumption in his diary.
- ▶ The number of ice creams he ate was influenced, but not entirely determined by the weather.
- ▶ Today's weather is partially predictable from yesterday's.



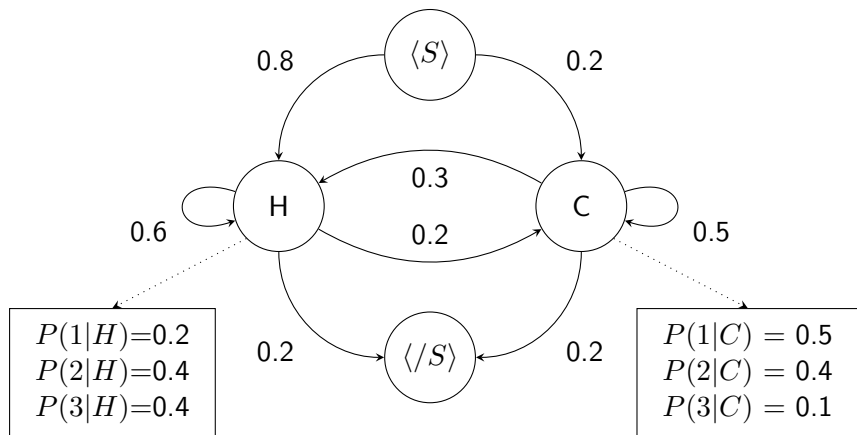
Missing records of weather in Baltimore, MD, for Summer 2007

- ▶ Jason likes to eat ice cream.
- ▶ He records his daily ice cream consumption in his diary.
- ▶ The number of ice creams he ate was influenced, but not entirely determined by the weather.
- ▶ Today's weather is partially predictable from yesterday's.

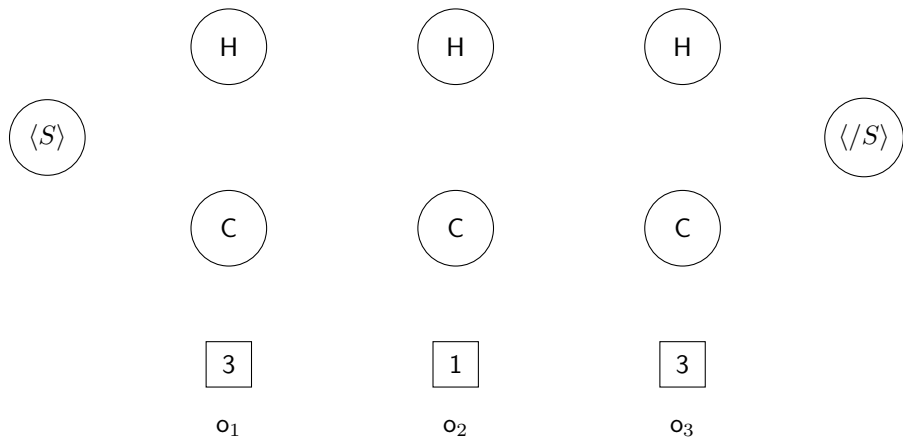
A Hidden Markov Model

- ▶ Hidden states: $\{H, C\}$ (plus pseudo-states $\langle S \rangle$ and $\langle /S \rangle$)
- ▶ Observations: $\{1, 2, 3\}$

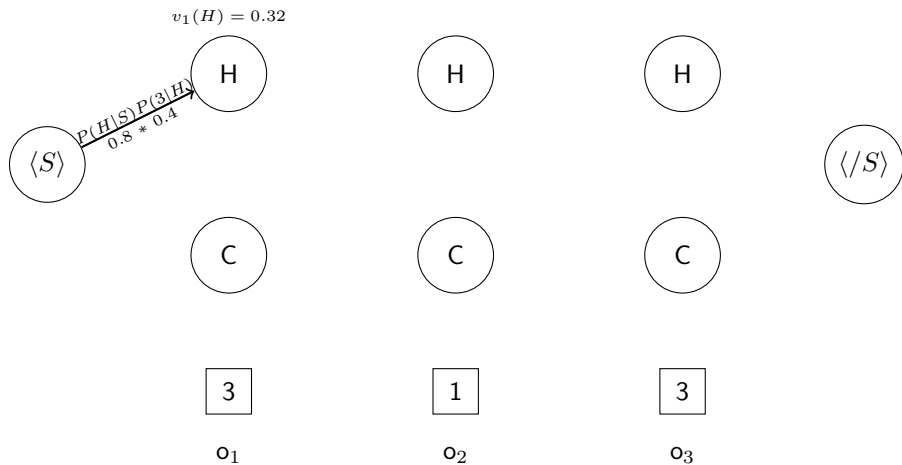
Recap: Ice Cream and Weather in Baltimore, MD



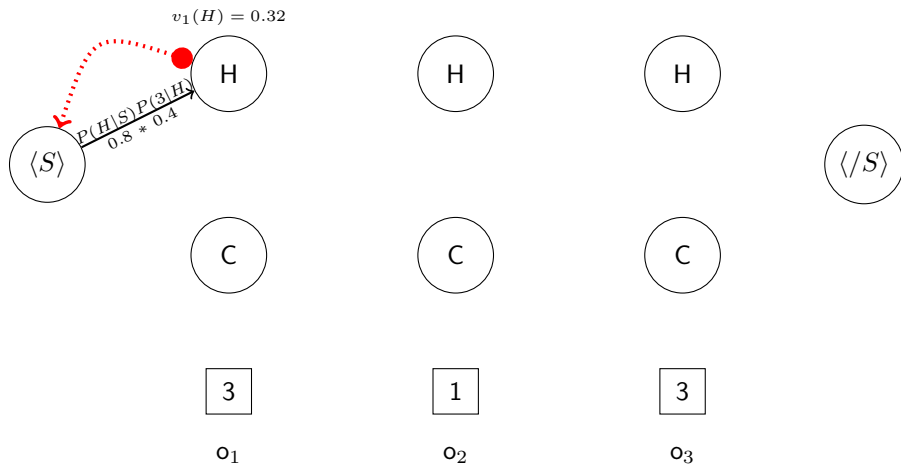
Recap: Viterbi Decoding—Thanks, Bec!



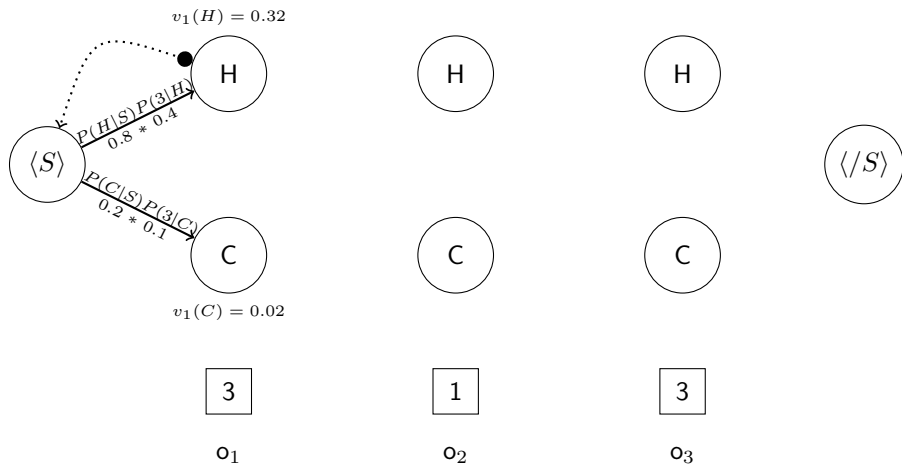
Recap: Viterbi Decoding—Thanks, Bec!



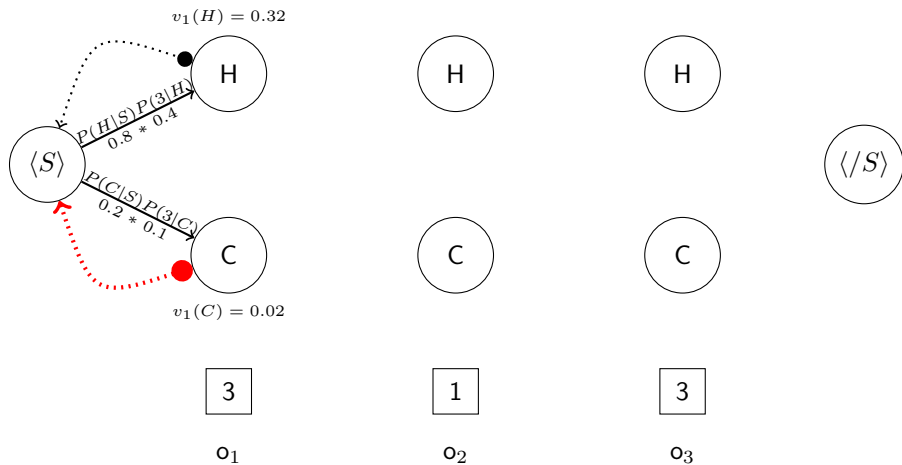
Recap: Viterbi Decoding—Thanks, Bec!



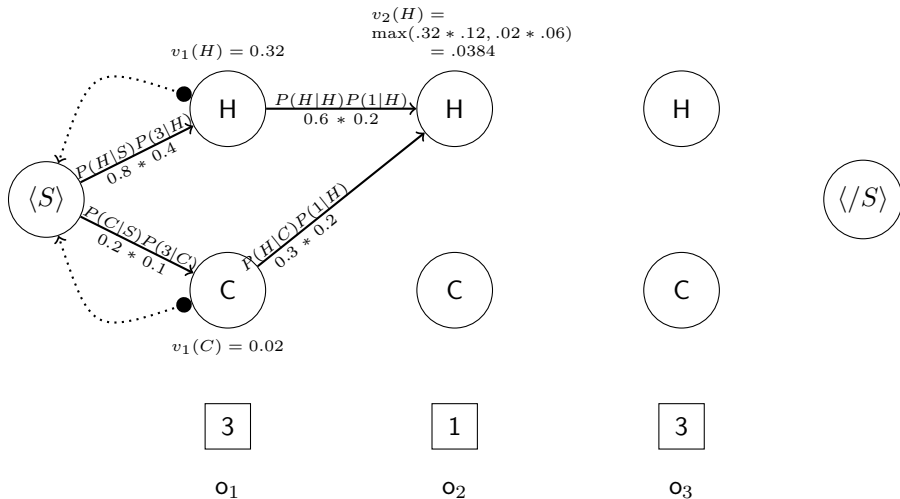
Recap: Viterbi Decoding—Thanks, Bec!



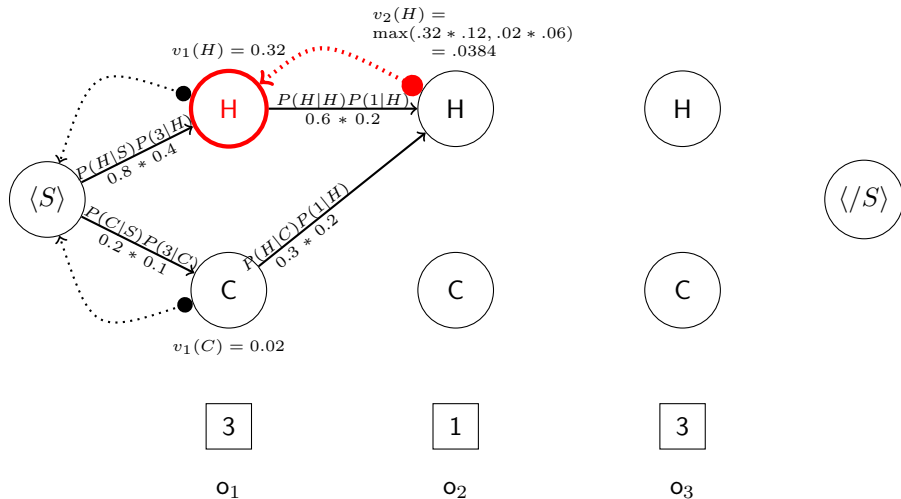
Recap: Viterbi Decoding—Thanks, Bec!



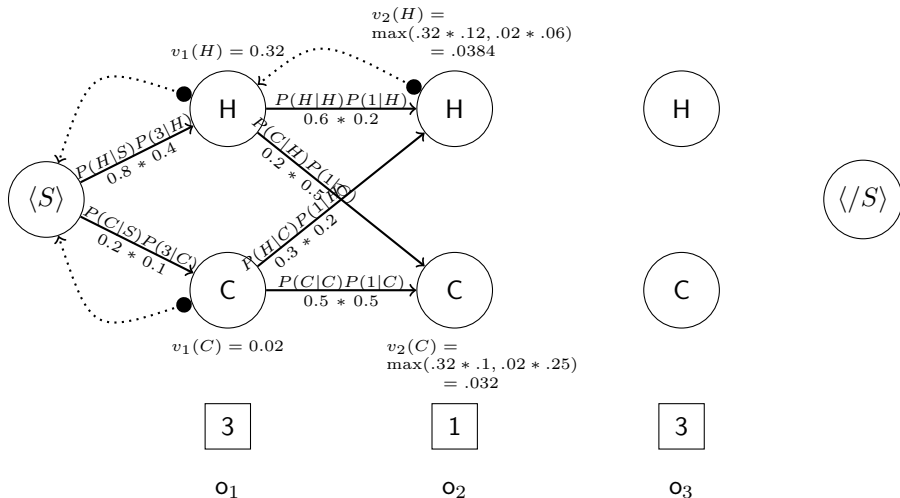
Recap: Viterbi Decoding—Thanks, Bec!



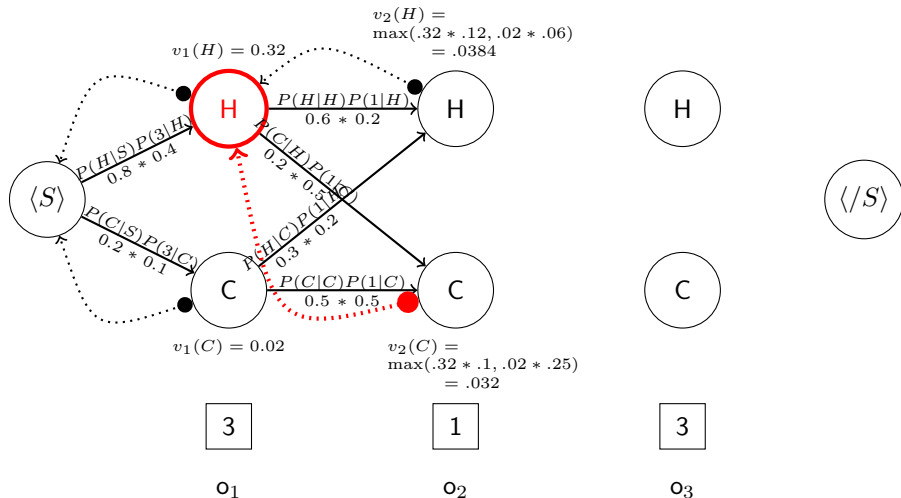
Recap: Viterbi Decoding—Thanks, Bec!



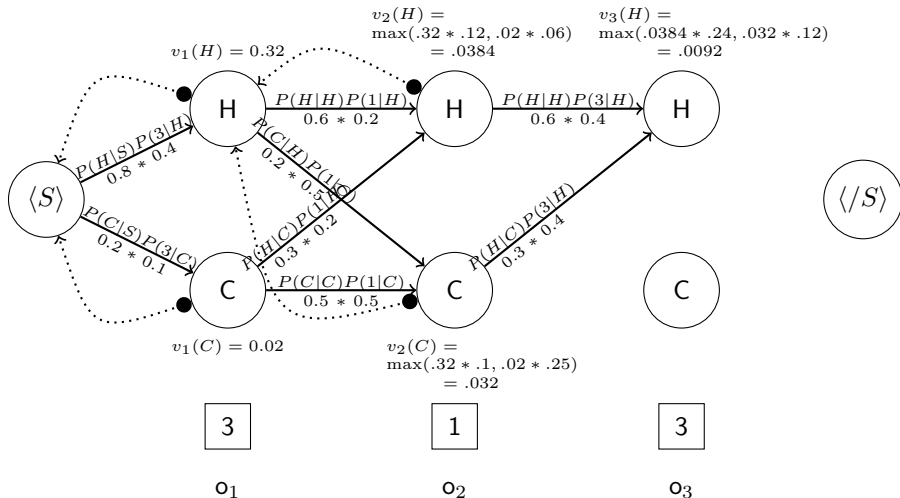
Recap: Viterbi Decoding—Thanks, Bec!



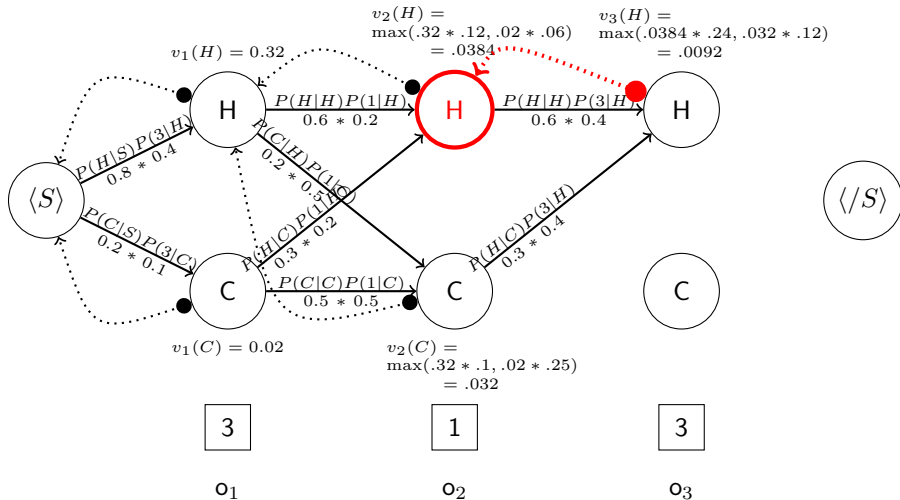
Recap: Viterbi Decoding—Thanks, Bec!



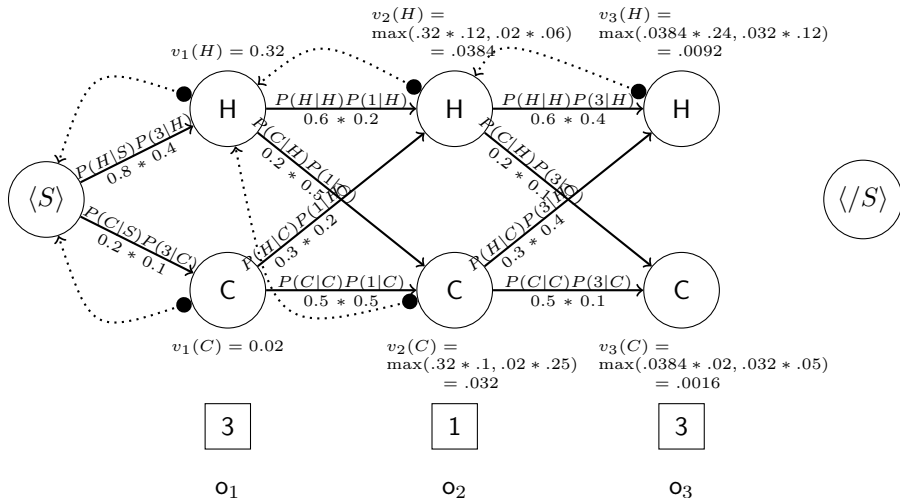
Recap: Viterbi Decoding—Thanks, Bec!



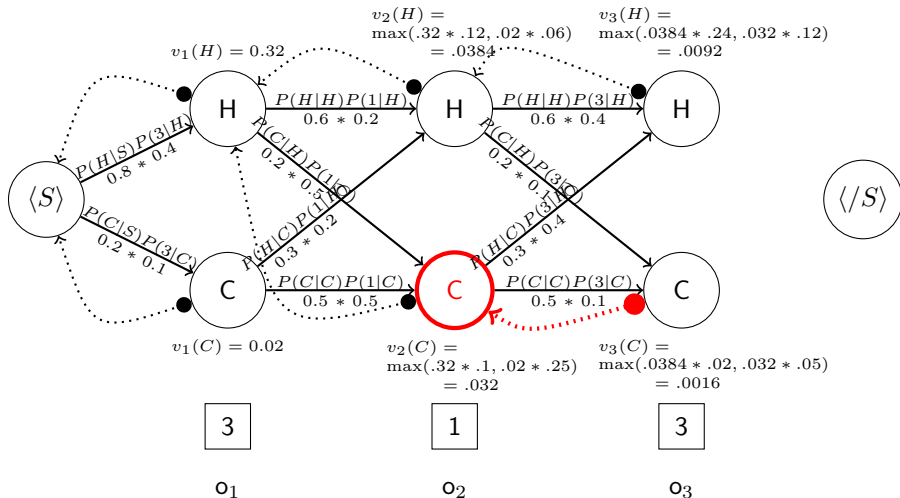
Recap: Viterbi Decoding—Thanks, Bec!



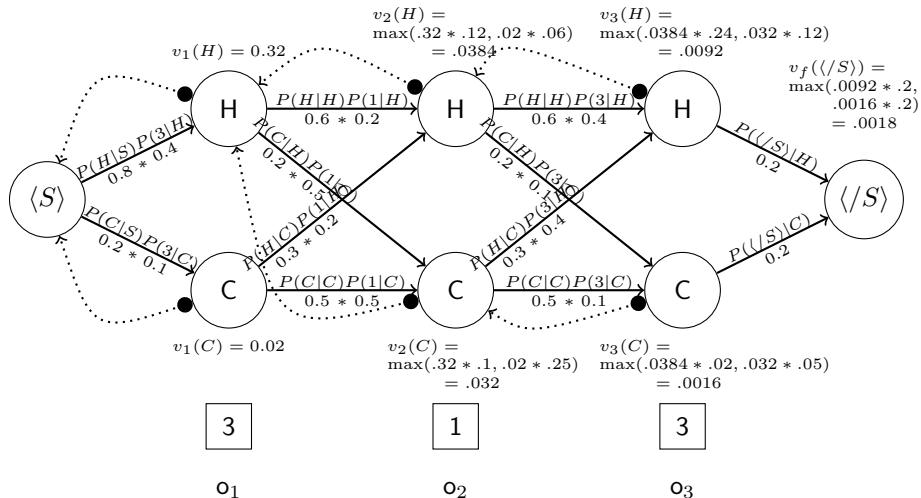
Recap: Viterbi Decoding—Thanks, Bec!



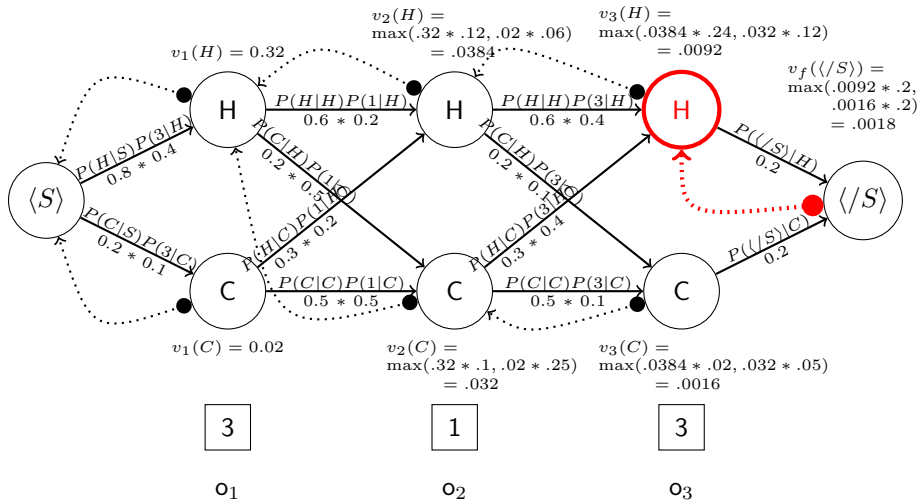
Recap: Viterbi Decoding—Thanks, Bec!



Recap: Viterbi Decoding—Thanks, Bec!

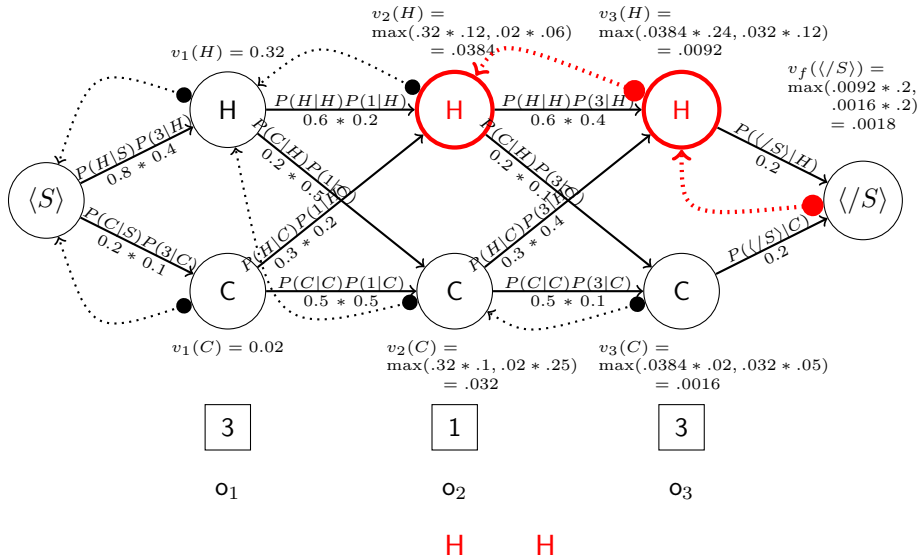


Recap: Viterbi Decoding—Thanks, Bec!

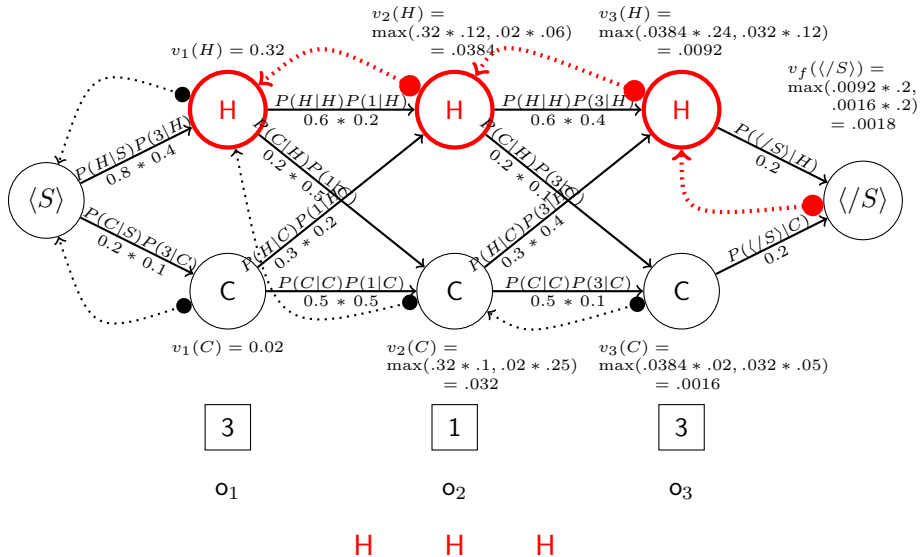


H

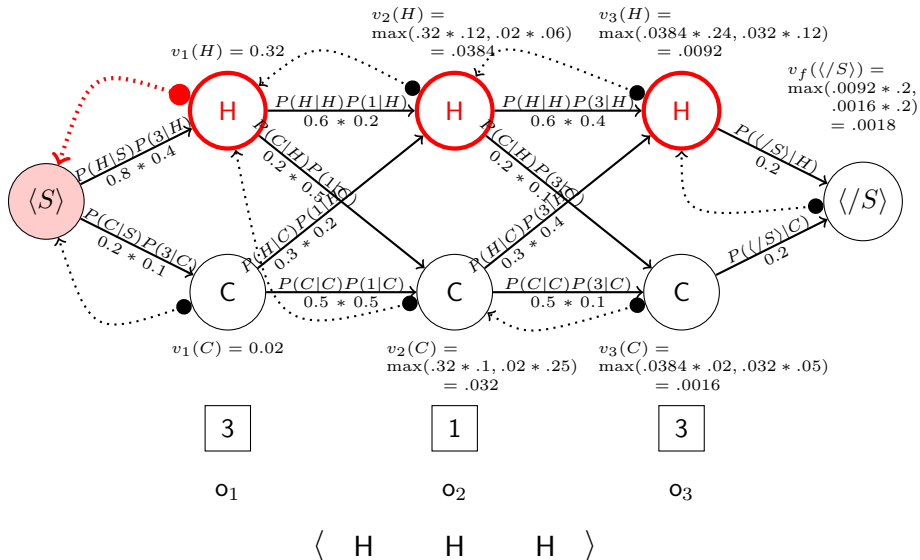
Recap: Viterbi Decoding—Thanks, Bec!



Recap: Viterbi Decoding—Thanks, Bec!



Recap: Viterbi Decoding—Thanks, Bec!





Abstract problem: Find the tag sequence $s_1 \dots s_n$ that maximizes

$$P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1 | s_0) P(o_1 | s_1) P(s_2 | s_1) P(o_2 | s_2) \dots$$



Abstract problem: **Find** the tag sequence $s_1 \dots s_n$ that **maximizes**

$$P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1 | s_0) P(o_1 | s_1) P(s_2 | s_1) P(o_2 | s_2) \dots$$

The Viterbi algorithm uses decomposition into recursive sub-problems:

$$v_i(x) = \max_{k=1}^L [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

Each trellis cell $v_i(x)$ represents the maximum probability that the i -th state is x , given that we have seen the observation prefix $o_1 \dots o_i$.



Abstract problem: Find the tag sequence $s_1 \dots s_n$ that maximizes

$$P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1 | s_0) P(o_1 | s_1) P(s_2 | s_1) P(o_2 | s_2) \dots$$

The Viterbi algorithm uses decomposition into recursive sub-problems:

$$v_i(x) = \max_{k=1}^L [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

Each trellis cell $v_i(x)$ represents the maximum probability that the i -th state is x , given that we have seen the observation prefix $o_1 \dots o_i$.



Abstract problem: **Find** the tag sequence $s_1 \dots s_n$ that **maximizes**

$$P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1 | s_0) P(o_1 | s_1) P(s_2 | s_1) P(o_2 | s_2) \dots$$

The Viterbi algorithm uses decomposition into recursive sub-problems:

$$v_i(x) = \max_{k=1}^L [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

Each trellis cell $v_i(x)$ represents the maximum probability that the i -th state is x , given that we have seen the observation prefix $o_1 \dots o_i$.

At each step, we also record **backpointers** (in a separate matrix), showing which previous state led to the maximum probability.

From Linear Order to Hierarchical Structure

- ▶ NLP approaches we have considered this far:

- ▶ Distributional representations of documents or words:

Cisco acquired Tandberg \equiv *Tandberg acquired Cisco*

- ▶ n -gram language models (Markov chains).

- ▶ Purely linear (sequential) and surface-oriented.

- ▶ sequence labeling: HMMs.

- ▶ One layer of abstraction: PoS as hidden states.

- ▶ Still only sequential in nature.

From Linear Order to Hierarchical Structure

- ▶ NLP approaches we have considered this far:
 - ▶ Distributional representations of documents or words:

Cisco acquired Tandberg \equiv *Tandberg acquired Cisco*
 - ▶ n -gram language models (Markov chains).
 - ▶ Purely linear (sequential) and surface-oriented.
 - ▶ sequence labeling: HMMs.
 - ▶ One layer of abstraction: PoS as hidden states.
 - ▶ Still only sequential in nature.
- ▶ **Syntax** adds hierarchical structure:
 - ▶ In NLP, being a sub-discipline of AI, we want our programs to '*understand*' natural language (on some level).
 - ▶ Finding the grammatical structure of sentences is an important step towards '*understanding*'.
 - ▶ Shift focus from **bags** or **sequences** to **hierarchical structure**.

The Case for Structure (1/3)

Constituency

- ▶ Words can 'lump together' into groups that behave like single units; these are called *constituents*.
- ▶ *Constituency tests* give evidence for syntactic structure:
 - ▶ interchangeable in similar syntactic environments.
 - ▶ can be co-ordinated (e.g. using *and* and *or*)
 - ▶ can be 'moved around' in a sentence as one unit

The Case for Structure (1/3)

Constituency

- ▶ Words can 'lump together' into groups that behave like single units; these are called *constituents*.
 - ▶ *Constituency tests* give evidence for syntactic structure:
 - ▶ interchangeable in similar syntactic environments.
 - ▶ can be co-ordinated (e.g. using *and* and *or*)
 - ▶ can be 'moved around' in a sentence as one unit
- (4) Kim read [a very interesting book about grammar]_{NP}.
Kim read [it]_{NP}.
- (5) Kim [read a book]_{VP}, [gave it to Sandy]_{VP}, and [left]_{VP}.
- (6) [Read the book]_{VP} I really meant to this week.

Examples from *Linguistic Fundamentals for NLP: 100 Essentials from Morphology and Syntax*. Bender (2013)

The Case for Structure (2/3)

Constituency

- ▶ Constituents as basic 'building blocks' of grammatical structure.
- ▶ Rules of grammar are sensitive to constituents.

The Case for Structure (2/3)

Constituency

- ▶ Constituents as basic 'building blocks' of grammatical structure.
- ▶ Rules of grammar are sensitive to constituents.
- ▶ A constituent usually has one **head daughter**, and is often named according to the type of its head:
 - ▶ A **noun phrase** (NP) has a **nominal** head:
 - (This is) [a book]_{NP}
 - (This is) [a very interesting book about grammar]_{NP}
 - ▶ A **verb phrase** (VP) has a **verbal** head:
 - (She) [eats]_{VP}
 - (She) [gives books to students]_{VP}
 - (She) [bet me ten bucks that it would rain]_{VP}

The Case for Structure (3/3)

Relations among Constituents

- ▶ Notions such as *subject* and *object* describe the **grammatical function** of a constituent in a larger structure.

The Case for Structure (3/3)

Relations among Constituents

- ▶ Notions such as *subject* and *object* describe the **grammatical function** of a constituent in a larger structure.
- ▶ **Agreement** establishes a symmetric relationship between properties of two constituents.
- ▶ **Government** allows one constituent to require a certain property of another constituent.
- ▶ The decision of the committee memberssurprises most of us.
- ▶ Why would a **purely linear** model have problems predicting this phenomenon?

The Case for Structure (3/3)

Relations among Constituents

- ▶ Notions such as *subject* and *object* describe the **grammatical function** of a constituent in a larger structure.
- ▶ **Agreement** establishes a symmetric relationship between properties of two constituents.
- ▶ **Government** allows one constituent to require a certain property of another constituent.
- ▶ The decision of the committee memberssurprises most of us.
- ▶ Why would a **purely linear** model have problems predicting this phenomenon?
- ▶ Verb agreement has to reflect the **grammatical structure** of the sentence, not merely the sequential order of words.

Grammars: A Tool to Aid Understanding

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Grammars: A Tool to Aid Understanding

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Meaning

- ▶ *Kim gave Sandy the book.*
- ▶ *Kim gave the book to Sandy.*
- ▶ *Sandy was given the book by Kim.*

Grammars: A Tool to Aid Understanding

Formal grammars describe a language, providing key notions of:

Wellformedness

- ▶ *Kim was happy because _____ passed the exam.*
- ▶ *Kim was happy because _____ final grade was an A.*
- ▶ *Kim was happy when she saw _____ on television.*

Meaning

- ▶ *Kim gave Sandy the book.*
- ▶ *Kim gave the book to Sandy.*
- ▶ *Sandy was given the book by Kim.*

Ambiguity

- ▶ *Kim ate sushi with chopsticks.*
- ▶ *Have her report on my desk by Friday!*

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

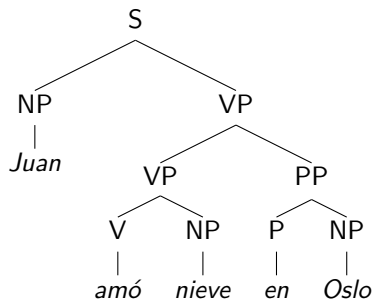
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

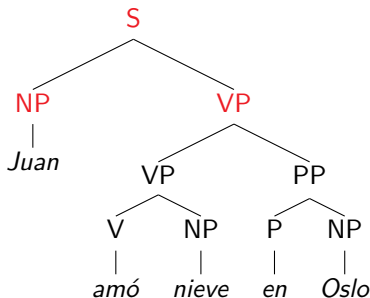
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

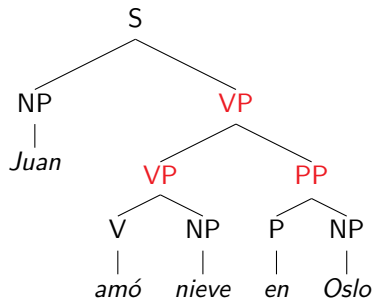
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

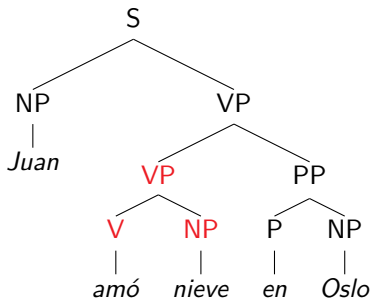
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

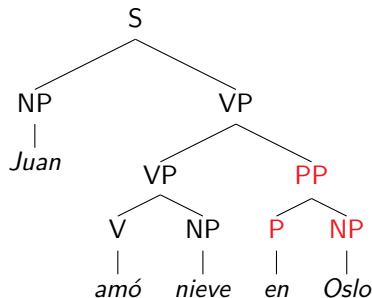
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

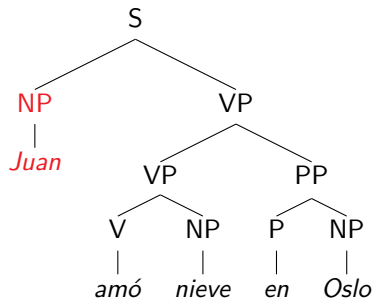
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

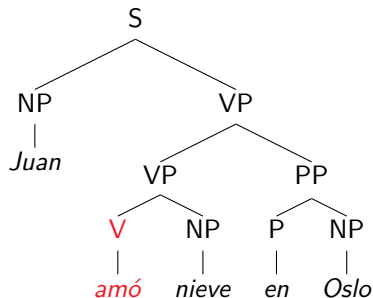
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

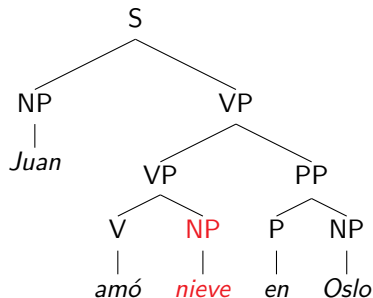
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

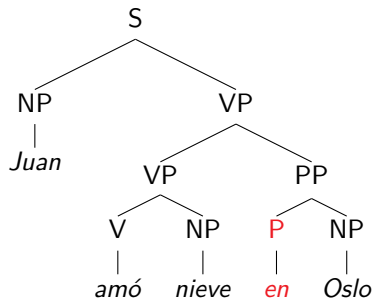
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$

$VP \rightarrow V NP$

$VP \rightarrow VP PP$

$PP \rightarrow P NP$

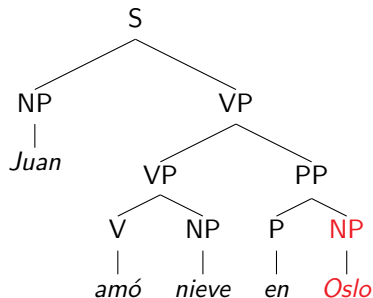
$NP \rightarrow \text{"nieve"}$

$NP \rightarrow \text{"Juan"}$

$NP \rightarrow \text{"Oslo"}$

$V \rightarrow \text{"amó"}$

$P \rightarrow \text{"en"}$



Juan amó nieve en Oslo

A Simplified Example

The Grammar of Spanish

$S \rightarrow NP VP$ $\{ VP (NP) \}$

$VP \rightarrow V NP$ $\{ V (NP) \}$

$VP \rightarrow VP PP$ $\{ PP (VP) \}$

$PP \rightarrow P NP$ $\{ P (NP) \}$

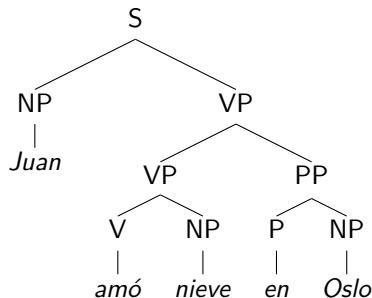
$NP \rightarrow \text{"nieve"}$ $\{ \text{snow} \}$

$NP \rightarrow \text{"Juan"}$ $\{ \text{John} \}$

$NP \rightarrow \text{"Oslo"}$ $\{ \text{Oslo} \}$

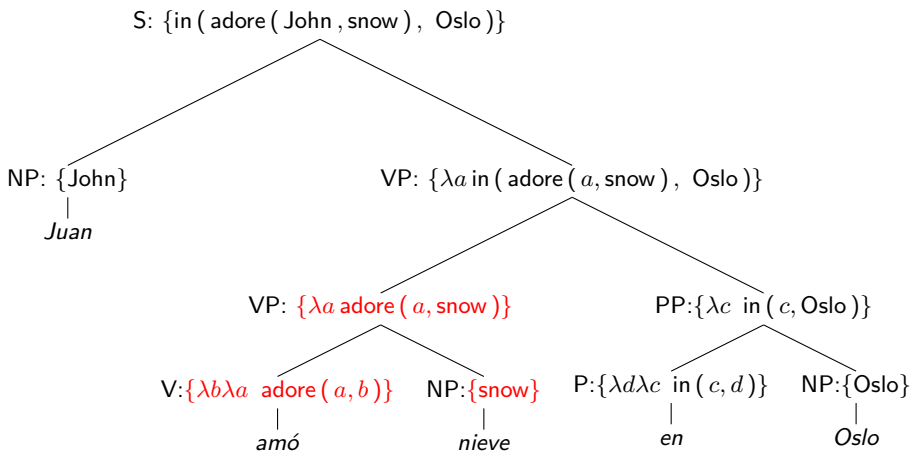
$V \rightarrow \text{"amó"}$ $\{ \lambda b \lambda a \text{ adore}(a, b) \}$

$P \rightarrow \text{"en"}$ $\{ \lambda d \lambda c \text{ in}(c, d) \}$



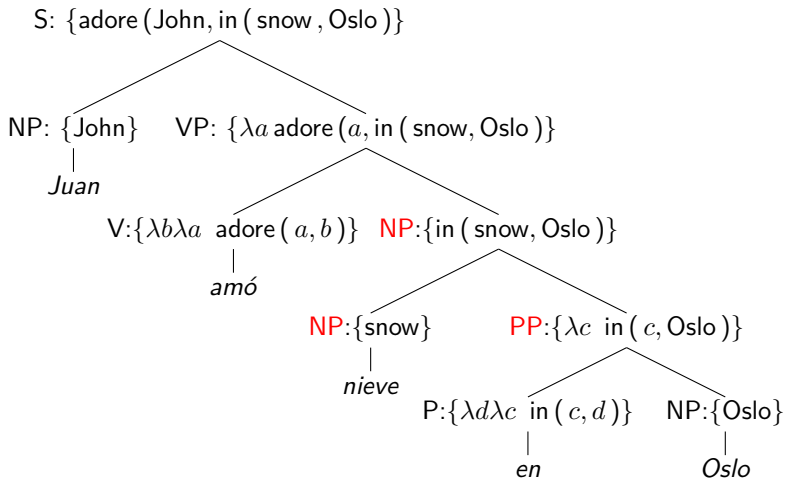
Juan amó nieve en Oslo

Meaning Composition (Still Very Simplified)



VP \rightarrow V NP $\{ V (NP) \}$

Another Interpretation



NP \rightarrow NP PP { PP (NP) }

Context Free Grammars (CFGs)

- ▶ Formal system for modeling constituent structure.
- ▶ Defined in terms of a lexicon and a set of rewrite rules.

Context Free Grammars (CFGs)

- ▶ Formal system for modeling constituent structure.
- ▶ Defined in terms of a lexicon and a set of rewrite rules.
- ▶ Precise, abstract models of 'language' in a broad sense
 - ▶ natural languages, programming languages, communication protocols, ...

Context Free Grammars (CFGs)

- ▶ Formal system for modeling constituent structure.
- ▶ Defined in terms of a lexicon and a set of rewrite rules.
- ▶ Precise, abstract models of 'language' in a broad sense
 - ▶ natural languages, programming languages, communication protocols, ...
- ▶ Can be expressed in the 'meta-syntax' of the Backus-Naur Form (BNF) formalism.
 - ▶ The standard Python documentation (or much other technical writing) often uses BNF.

Context Free Grammars (CFGs)

- ▶ Formal system for modeling constituent structure.
- ▶ Defined in terms of a lexicon and a set of rewrite rules.
- ▶ Precise, abstract models of 'language' in a broad sense
 - ▶ natural languages, programming languages, communication protocols, . . .
- ▶ Can be expressed in the 'meta-syntax' of the Backus-Naur Form (BNF) formalism.
 - ▶ The standard Python documentation (or much other technical writing) often uses BNF.
- ▶ Powerful enough to express sophisticated relations among words and constituents, yet computationally tractable.

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$
- ▶ Σ is the vocabulary (aka *terminals*),
 - ▶ $\{\text{Kim, snow, adores, in}\}$

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$
- ▶ Σ is the vocabulary (aka *terminals*),
 - ▶ $\{\text{Kim, snow, adores, in}\}$
- ▶ P is a set of category rewrite rules (aka *productions*)

$S \rightarrow NP VP$

$NP \rightarrow \text{Kim}$

$VP \rightarrow V NP$

$NP \rightarrow \text{snow}$

$V \rightarrow \text{adores}$

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$
- ▶ Σ is the vocabulary (aka *terminals*),
 - ▶ $\{\text{Kim, snow, adores, in}\}$
- ▶ P is a set of category rewrite rules (aka *productions*)

$S \rightarrow NP VP$	$NP \rightarrow \text{Kim}$
$VP \rightarrow V NP$	$NP \rightarrow \text{snow}$
	$V \rightarrow \text{adores}$
- ▶ $S \in C$ is the *start symbol*, a filter on complete results;

CFGs (Formally, this Time)

Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$
- ▶ Σ is the vocabulary (aka *terminals*),
 - ▶ $\{\text{Kim, snow, adores, in}\}$
- ▶ P is a set of category rewrite rules (aka *productions*)

$S \rightarrow NP VP$

$NP \rightarrow \text{Kim}$

$VP \rightarrow V NP$

$NP \rightarrow \text{snow}$

$V \rightarrow \text{adores}$

- ▶ $S \in C$ is the *start symbol*, a filter on complete results;
- ▶ for each rule $\alpha \rightarrow \beta_1, \beta_2, \dots, \beta_n \in P$: $\alpha \in C$ and $\beta_i \in C \cup \Sigma$

Foundations of formal language theory:

- ▶ For a grammar G , the language \mathcal{L}_G is defined as the **set of strings** that can be derived from S .
- ▶ To derive w_1^n from S , we use the rules in P to recursively rewrite S into the sequence w_1^n (where each $w_i \in \Sigma$)

Foundations of formal language theory:

- ▶ For a grammar G , the language \mathcal{L}_G is defined as the **set of strings** that can be derived from S .
- ▶ To derive w_1^n from S , we use the rules in P to recursively rewrite S into the sequence w_1^n (where each $w_i \in \Sigma$)
- ▶ The grammar can be seen as **generating** strings.
- ▶ **Grammatical strings** are defined as terminal sequences that can be generated by the grammar.

Foundations of formal language theory:

- ▶ For a grammar G , the language \mathcal{L}_G is defined as the **set of strings** that can be derived from S .
- ▶ To derive w_1^n from S , we use the rules in P to recursively rewrite S into the sequence w_1^n (where each $w_i \in \Sigma$)
- ▶ The grammar can be seen as **generating** strings.
- ▶ **Grammatical strings** are defined as terminal sequences that can be generated by the grammar.
- ▶ The 'context-freeness' of CFGs refers to the fact that we rewrite non-terminals without regard to the overall context in which they occur.

Next week

- ▶ Parsing: Computing the language of a CFG
- ▶ (More on) Statistical parsing
- ▶ Dependency Syntax
- ▶ Transition-based dependency parsing