INF4820: Algorithms for Artificial Intelligence and Natural Language Processing

HMMs & Context-Free Grammars

Stephan Oepen & Erik Velldal

Language Technology Group (LTG)

October 21, 2015

Overview

Last Time

- Sequence Labeling
- Dynamic programming
- Viterbi algorithm

Today

- Mid-Way Evaluation
- Forward Algorithm
- Quiz and Bonus Points
- Syntactic structure
  - Context-free grammar
  - Treebanks

Recall: Training an *N*-Gram Model

How to estimate the probabilities of *n*-grams?
Maximum Likelihood Estimation; counting (e.g. for trigrams):

$$P\,(\text{bananas}|\text{i like}) = \frac{C\,(\text{i like bananas})}{C\,(\text{i like})}$$

Using Laplace ('add-one') smoothing:

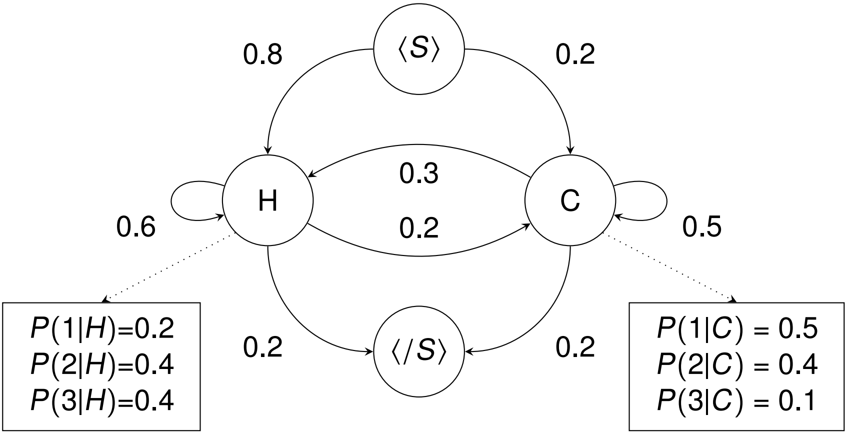
$$P_{\text{L}}\,(w_n|w_{n-2}, w_{n-1}) = \frac{C\,(w_{n-2}w_{n-1}w_n) + 1}{C\,(w_{n-2}w_{n-1}) + V}$$

$$C\,(w_{n-2}w_{n-1}) + V = \sum_{w \in V} C\,(w_{n-2}w_{n-1}w) + 1$$

Recall: Dynamic Programming

- Dynamic programming algorithms
  - solve large problems by compounding answers from smaller sub-problems
  - record sub-problem solutions for repeated use
- They are used for complex problems that
  - can be described recursively
  - require the same calculations over and over again
- Examples:
  - Dijkstra’s shortest path
  - minimum edit distance
  - longest common subsequence
  - Viterbi

Recall: Ice Cream and Global Warming



Recall: Viterbi Algorithm

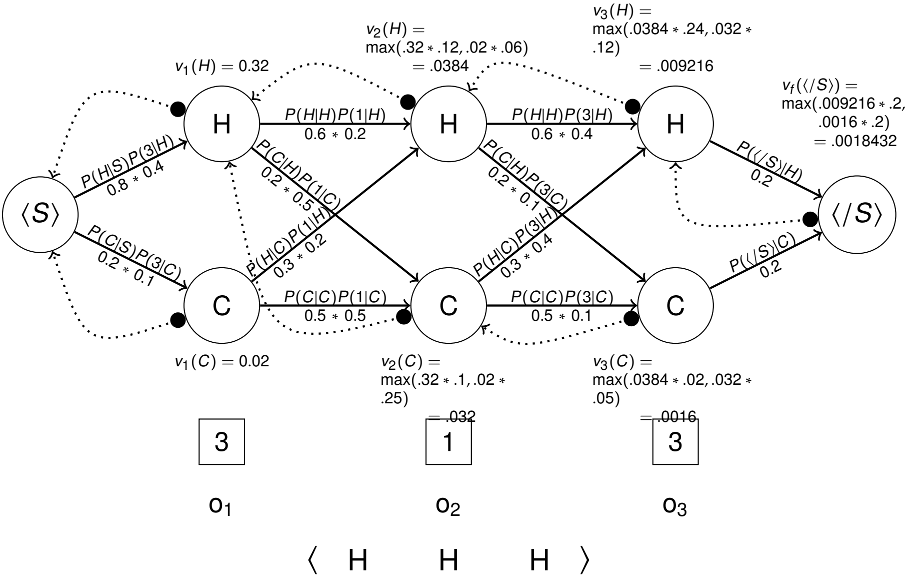
- To find the best state sequence, maximize:

  $P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1|s_0)P(o_1|s_1)P(s_2|s_1)P(o_2|s_2)\dots$
- The value we cache at each step:

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

- The variable $v_i(x)$ represents the maximum probability that the i-th state is x, given that we have seen O_1^i.
- At each step, we record backpointers showing which previous state led to the maximum probability.

Recall: An Example of the Viterbi Algorithmn



Recall: Using HMMs

The HMM models the process of generating the labelled sequence. We can use this model for a number of tasks:

- $P(S, O)$ given S and O
- $P(O)$ given O
- S that maximizes $P(S|O)$ given O
- $P(s_x|O)$ given O
- We can learn model parameters from a set of observations.

Computing Likelihoods

Task

Given an observation sequence O, determine the likelihood $P(O)$, according to the HMM.

Compute the sum over all possible state sequences:

$$P(O) = \sum_{S} P(O, S)$$

For example, the ice cream sequence 3 1 3:

$$\begin{aligned} P(3\ 1\ 3) = \quad & P(3\ 1\ 3, \mathrm{C\ C\ C}) + \\ & P(3\ 1\ 3, \mathrm{C\ C\ H}) + \\ & P(3\ 1\ 3, \mathrm{H\ H\ C}) + \dots \end{aligned}$$

The Forward Algorithm

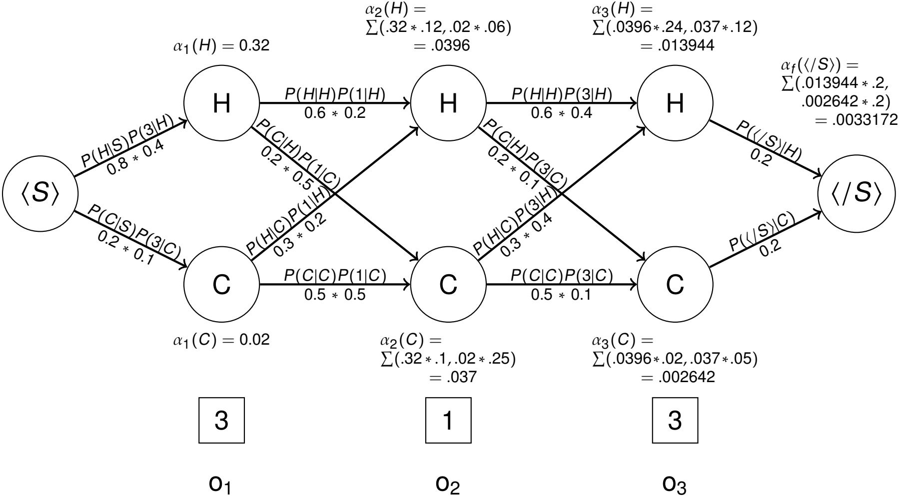
Again, we use dynamic programming—storing and reusing the results of partial computations in a trellis α.

Each cell in the trellis stores the probability of being in state x after seeing the first i observations:

$$
\begin{aligned}
\alpha_i(x) &= P(o_1 \ldots o_i, s_i = x) \\
&= \sum_{k=1}^{L} \alpha_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)
\end{aligned}
$$

Note $\sum$, instead of the max in Viterbi.

An Example of the Forward Algorithmn



$$P(3\,1\,3) = 0.0033172$$

An Experiment in High-Tech Teaching

- For student involvement and incremental exam preparation:
- two more short quiz sessions with extra points towards exercises.

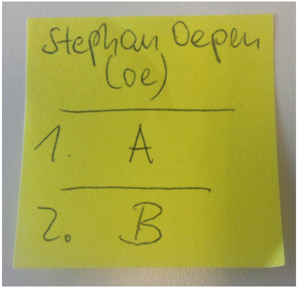
Example Quiz (0 + 0 Points)

1. Live programming can be useful?

   A: yes; B: no

2. Lisp was first developed by:

   A: Alan Turing; B: John McCarthy



Give us Those Bonus Points

Rules of the Game

- Up to two bonus points towards Exercise (2) or (3).
- Get one post-it; at the top, write your first and last name.
- Write down your UiO user name (e.g. `oe`, in my case).
- Write each answer on a line of its own; prefix each with the question number.
- Do not consult with your neighbors; they will likely mess things up (also, this is an exam-related activity).

After the Quiz

- Post your answers at the front of your table, we will come around and collect all notes.
- Discuss your answers with your neighbor(s); explain why you are right (in a professional tone).

Question (1): The Monty Hall Problem

On a gameshow, there are three doors.

Behind two doors, there is a goat.
Behind the third door, there is a car.

The contestant selects a door, hoping for the car.
Before she opens that door, the gameshow host opens one of the other doors and reveals a goat.

The contestant can now open the door she originally chose, or switch to the other unopened door.

(1) What is the probability of finding the car when switching?

Question (2): Language Modelling

Group members at the Language Technology Group supervise a variety of topics for MSc projects in natural language processing. Many candidate projects are available on-line. Please make contact with us.

(2) What is the probability of the bi-gram

language technology

when ignoring case and punctuation, and using Laplace smoothing?

Question (3): Space Complexity

We have discussed the time complexity
of the Viterbi algorithm in relation to two variables:

the number of distinct states L
and the length of the observation sequence N.

(3) What is the order of growth for memory space used by the Viterbi algorithm, relative to L and N?

Question (4): HMM Viterbi vs. Forward

Recall the recursive formulation of the Viterbi Algorithm:

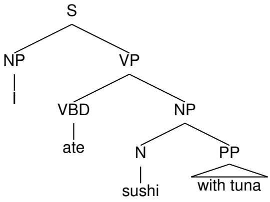
$$v_i(x) = \max_{k=1}^{L} [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

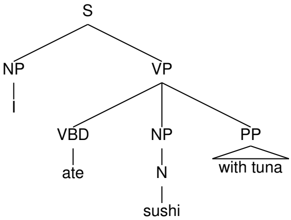
(4) What is different in the Forward Algorithm; and what HMM-related task does it compute?

Moving Onwards

Determining

- which string is most likely: ✓
  - *How to recognize speech* vs. *How to wreck a nice beach*
- which tag sequence is most likely for *flies like flowers*: ✓
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic structure is most likely:





From Linear Order to Hierarchical Structure

- The models we have looked at so far:
  - *n*-gram models (Markov chains).
    - Purely linear (sequential) and surface oriented.
  - sequence labeling: HMMs.
    - Adds one layer of abstraction: PoS as hidden variables.
    - Still only sequential in nature.
- Formal grammar 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 *sequences* to *syntactic structures*.

Why We Need Structure (1/3)

Constituency

- Words tends to lump together into groups that behave like single units: we call them *constituents*.
- *Constituency tests* give evidence for constituent structure:
  - interchangeable in similar syntactic environments.
  - can be co-ordinated
  - can be moved within a sentence as a unit

(1) Kim read [a very interesting book about grammar]$_{NP}$.
Kim read [it]$_{NP}$.

(2) Kim [read a book]$_{VP}$, [gave it to Sandy]$_{VP}$, and [left]$_{VP}$.

(3) You said I should read the book and [read it]$_{VP}$ I did.

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

Why We Need Structure (2/3)

Constituency

- Constituents are theory-dependent, and are not absolute or language-independent.
- Language word order is often described in terms of constituents, and word order may be more or less free within constituents or between them.
- A constituent usually has a *head* element, and is often named according to the type of its head:
  - A noun phrase (NP) has a nominal (noun-type) head:

    (4) [a very interesting <u>book</u> about grammar]$_{\mathrm{NP}}$

  - A verb phrase (VP) has a verbal head:

    (5) [<u>gives</u> books to students]$_{\mathrm{VP}}$

Why We Need Structure (3/3)

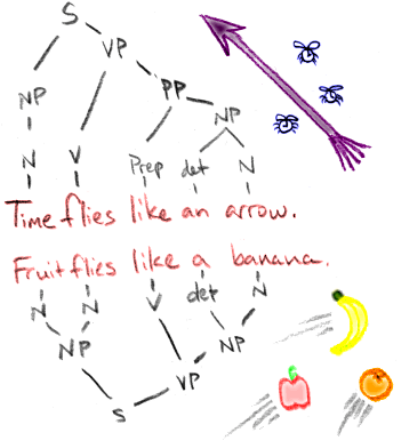
Grammatical functions

- Terms such as *subject* and *object* describe the grammatical function of a constituent in a sentence.
- *Agreement* establishes a symmetric relationship between grammatical features.

  The *decision* of the Nobel committee member*s* surprise*s* most of us.

- Why would a purely linear model have problems predicting this phenomenon?
- Verb agreement reflects the grammatical structure of the sentence, not just the sequential order of words.

Syntactic Ambiguity



(*Speculative Grammarian, The Journal of Satirical Linguistics*)

Grammars: A Tool to Aid Understanding

Formal grammars describe a language, giving us a way to:

- judge or predict well-formedness

  Kim was happy because ________ passed the exam.

  Kim was happy because ________ final grade was an A.

- make explicit structural ambiguities

  Have her report on my desk by Friday!

  I like to eat sushi with { chopsticks | tuna }.

- derive abstract representations of meaning

  Kim gave Sandy a book.

  Kim gave a book to Sandy.

  Sandy was given a book by Kim.