Topics in Natural Language Processing

Shay Cohen

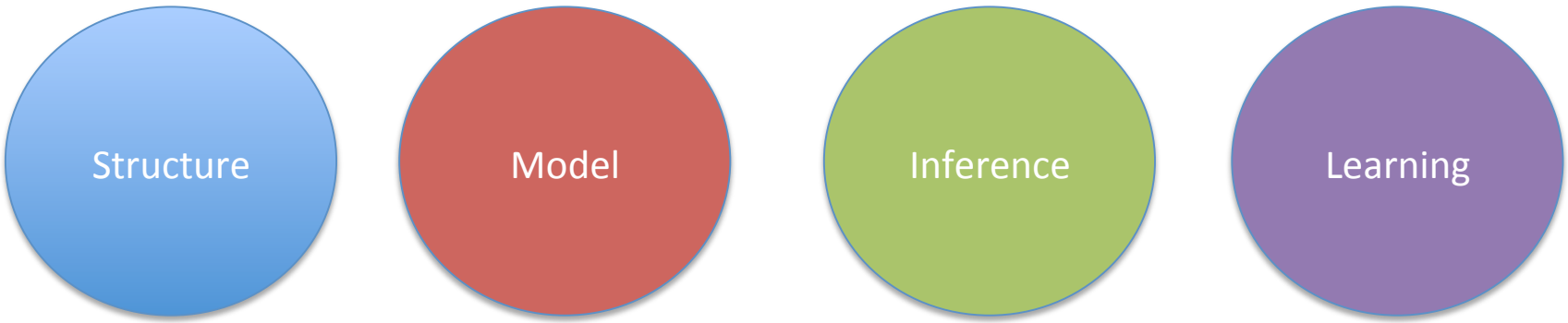
Institute for Language, Cognition and Computation

University of Edinburgh

Lecture 5

Solving an NLP Problem

When modelling a new problem in NLP, need to address four issues:



Inference

A process in which one has to take a model and an input and predict or calculate some quantity in the output space

Most commonly:

$$y = \arg\max_x p(y \mid x, \theta)$$

or

$$y = \arg\max_x \text{score}(x, y)$$

Types of Inference Algorithms

Types of Inference Algorithms

- Search algorithms
- Dynamic programming
- Sampling algorithms
- Integer linear programming

Search Algorithms

A traditional AI approach to find a solution

There is a “search space,” each element is a node in the graph

There is a “cost” associated with each node

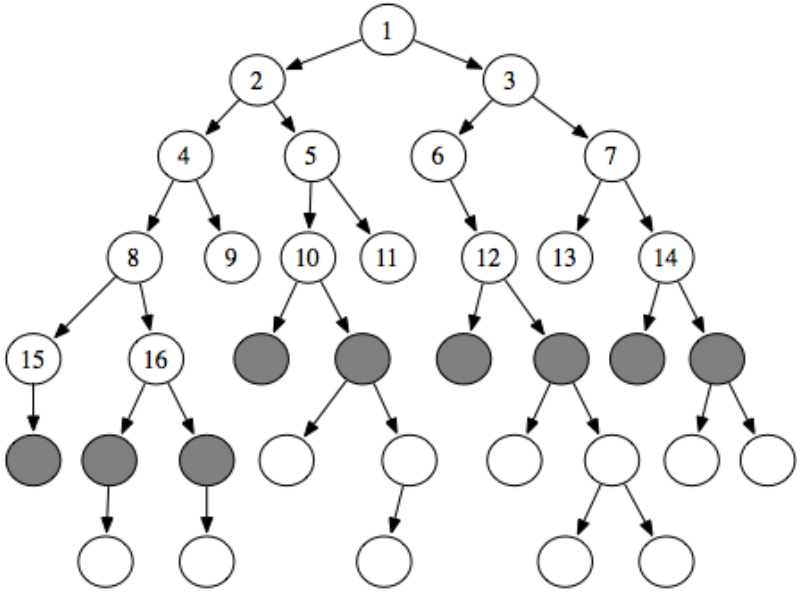
There are edges between these nodes according to simple operators that change one node into another

There is a “goal” node which is the solution we are looking for (with the smallest cost)

There is a strategy to explore the graph and find the goal node

Example Strategy

Breadth-first search:



Best-first search (BFS):
Visit the nodes in order of their value.

Often coupled with “beam” search
Keep only partial set of nodes open (according to their value)

Another Example: A* Search

Exploring the graph by increasing cost. Each step adds to the cost.

Develop paths in the graph using a heuristic of the cost left to reach the goal.

If $h(v)$ is the estimated cost to reach the goal from node v and $f(v)$ is the cost associated with node v, then explore the node with minimum $\min_v f(v) + h(v)$

If $h(v)$ never *overestimates* the cost ("h is admissible") the first time the algorithm finds a goal, that goal is the correct one.

Dynamic Programming Algorithms

- Solve a “bigger” problem by breaking it into “smaller” parts
- The smaller parts are now the bigger problems – work recursively
- Examples: Viterbi algorithm, parsing algorithms such as CKY

Inside and CKY

What is the connection between the inside algorithm and CKY?

CKY:

$$\alpha(A,i,j) = \max_{i \leq k \leq j-1} \max_{A \to B\,C} p(A \to B\,C|A)\alpha(B,i,k)\alpha(C,k+1,j)$$

Inside and CKY

What is the connection between the inside algorithm and CKY?

CKY:

$$\alpha(A,i,j) = \max_{i \leq k \leq j-1} \max_{A \to B\,C} p(A \to B\,C|A)\alpha(B,i,k)\alpha(C,k+1,j)$$

Inside:

$$\alpha(A,i,j) = \sum_{k=i}^{j-1} \sum_{A \to B\,C} p(A \to B\,C|A)\alpha(B,i,k)\alpha(C,k+1,j)$$

The inside algorithm computes the *total probability* of a string – *summing* out all derivations instead of maximising over them

Semirings

What is a semiring?

- A set R
- Two operations: $\oplus$ and $\otimes$
- Identity element $\overline{1}$ for $\otimes$
- Identity element $\overline{0}$ for $\oplus$
- (... and a few more important properties)

CKY and Semirings

CKY:

$$\alpha(A,i,j) = \max_{i \le k \le j-1} \max_{A \to B\,C} p(A \to B\,C|A)\alpha(B,i,k)\alpha(C,k+1,j)$$

What is the semiring?

$\oplus$

$$a \oplus b = \max\{a,b\}$$

$\otimes$

$$a \otimes b = a \times b$$

$\overline{1}$	1
$\overline{0}$	0

Inside Algorithm and Semirings

CKY:

$$\alpha(A,i,j) = \sum_{i \le k \le j-1} \sum_{A \to B\,C} p(A \to B\,C|A)\alpha(B,i,k)\alpha(C,k+1,j)$$

What is the semiring?

$\oplus$

$$a \oplus b = a + b$$

$\otimes$

$$a \otimes b = a \times b$$

$\overline{1}$	1
$\overline{0}$	0

Log-Domain Trick and Semirings

CKY:

$$\alpha(A,i,j) = \max_{i \le k \le j-1} \max_{A \to B\,C} \log p(A \to B\,C|A) + \alpha(B,i,k) + \alpha(C,k+1,j)$$

What is the semiring?

$$a \oplus b = \max\{a,b\}$$

$\otimes$

$$a \otimes b = a + b$$

$\overline{1}$

$$\log 1 = 0$$

$\overline{0}$

$$\log 0 = -\infty$$

Examples of Semirings

Name	Domain	$\oplus$	$\otimes$	0	1	Description
Viterbi	$\mathbb{R}^+$	$\max$	$\times$	0	1	Find most likely derivation
Inside	$\mathbb{R}^+$	$+$	$\times$	0	1	Sum over derivations
Count	$\mathbb{N}$	$+$	$\times$	0	1	Count the number of derivations
Arctic	$\mathbb{R} \cup \{-\infty\}$	$\max$	$+$	$-\infty$	0	Find most likely derivation in $\log$-domain
Tropical	$\mathbb{R}^+ \cup \{\infty\}$	$\min$	$+$	∞	0	Same as arctic
Boolean	$\{\mathrm{t}, \mathrm{f}\}$	$\vee$	$\wedge$	f	t	Recognition
String	$\Sigma^* \cup \{s_\infty\}$	$\wedge$	$\circ$	s_∞	ε	Outputting strings

Parsing as Weighted Logic Programming

$$\text{constit}(a, i, j) \oplus = \text{constit}(b, i, k) \otimes \text{constit}(c, k+1, j) \otimes \text{rule}(a \rightarrow b\, c)$$

$$\text{constit}(a, i, i) \oplus = \text{rule}(a \rightarrow w)$$

Goal: $\text{constit}(S, 0, n)$

Weighted Logic Programmes

A succinct useful representation for dynamic programming algorithms

It represents inference algorithms in a generic way

Does not commit to a specific “execution model” – but dynamic programming is often used

Example of a Weighted Logic Programme

We are given a sequence $w_1, \ldots, w_n$ of some symbols.

$$\text{prob}(b, i) \oplus = \text{prob}(a, i-1) \otimes \text{transition}(a \to b) \otimes \text{emission}(b, w_i)$$

$$\text{prob}(a, 1) \oplus = \text{start\_state}(b) \otimes \text{emission}(a, w_1)$$

Example of a Weighted Logic Programme

We are given a sequence $w_1, \ldots, w_n$ of some symbols.

$$\text{prob}(b, i) \oplus = \text{prob}(a, i-1) \otimes \text{transition}(a \to b) \otimes \text{emission}(b, w_i)$$

$$\text{prob}(a, 1) \oplus = \text{start\_state}(b) \otimes \text{emission}(a, w_1)$$

Hidden Markov models and the forward algorithm:

- emission are the emission probabilities
- transition are the transition probabilities
- start_state are the initial probabilities
- $\text{prob}(b, i)$ gives the probability $p(w_1, \ldots, w_i, S_i = b)$

Solving Weighted Logic Programmes

- Memoisation and dynamic programming
- Agenda algorithms. Roughly:
  - Keep a queue ("agenda") of unprocessed items and a chart of processed items
  - Dequeue an item x and create or update all items by using information from the chart together with x. Put all the new items in the queue
  - Depending on the order by which we dequeue from the agenda, we might process items several times
  - Examples of agenda priority values: the value of an item, the value of an item with A* heuristic

Reminder about Bayesian Inference

Bayesian inference:

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{p(x)}$$

Bayesian inference with latent variables:

$$p(z, \theta \mid x) = \frac{p(\theta)p(x, z \mid \theta)}{p(x)}$$

Sampling

Instead of finding the most likely structure, we randomly sample a structure from the underlying distribution

If the distribution is peaked, then it will be roughly the same as finding the highest scoring structure

One can also use “annealing” with sampling to find the highest scoring structure

Sampling Algorithms

Goal: sample from a target distribution $p(u)$

For example, $p(u)$ could be the posterior from a Bayesian model, in which case $u = (z, \theta)$

Most common sampling algorithms: Markov Chain Monte Carlo methods

Ideal for cases in which we cannot compute the “normalisation” constant such as with Bayesian models

Markov Chain Monte Carlo

The big picture:

- Our sample space becomes a space of “states”
- There is some strategy to probabilistically move between states
- The strategy ensures that the transition between states will at some point converge to the target distribution we are interested in
- The samples do *not* have to be independent, and usually are not!
- Very useful for cases in which we can calculate the target distribution up to its normalisation constant (such as with Bayesian inference)

Example of MCMC: Gibbs Sampling

Break a "state" into several parts, for example, two parts: z and θ

Algorithm:

Let θ^* and z_0 be some random values.
Repeat until convergence

- Sample z^* from $p(z \mid \theta^*)$
- Sample θ^* from $p(\theta \mid z_0)$
- Set z_0 to be z^*

Collect the samples θ^* and z^*

We can also break z into further parts and use "operators" to move between states, making small changes to a bigger structure.

Strong relationship to search algorithms

More General: Metropolis-Hastings Algorithm

We are interested in sampling from $p(u)$, but can only sample from a *proposal* distribution $q(u'|u)$

Algorithm:

Initialise u with some value from Ω
Repeat until convergence

- Sample u' from $q(u'|u)$
- Calculate an acceptance ratio

$$\alpha = \min\left\{1, \frac{p(u')q(u|u')}{p(u)q(u'|u)}\right\}$$

- Set $u \leftarrow u'$ with probability α

Collect the samples u all through

Integer Linear Programming

Formulate the problem of inference as maximising a linear objective with constraints

The variables in the objective are “pieces” of the structure

Useful technique to add “global” constraints to the inference algorithm

Use off-the-shelf tools to find a solution, such as CPLEX or Gurobi

Summary

- Inference in our context refers to finding an output for a given input (decoding)
- ... or other type of information about the output
- Most common ways to do that in NLP: search algorithms, dynamic programming, sampling algorithms, integer linear programming