Lecture 3

Mark-Jan Nederhof

1 probabilities

Recapitulation

Where CFGs (may) generate non-regular languages, we can approximate them by FAs

We can measure percentages of incorrect strings that are accepted, and correct strings that are rejected, by approximating FA

But often the distinction between grammatical and ungrammatical is not sharp

Grammars (and automata) are often weighted/probabilistic

Weights can express degrees of 'correctness'

What are implications for regular approximation of CFGs?

Weighted devices

Fix some semiring $(\mathcal{W}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$

E.g. probabilistic semiring $(\mathbb{R}^{\geq 0}, +, \cdot, 0, 1)$

(Usually only weights between 0 and 1 are used)

In **weighted CFG** every rule ρ is given weight $p(\rho)$

Weight $p(d)$ of derivation d is product of (occurrences of) rules used in that derivation

(Assume some canonical form of derivations, e.g. left-most)

Weight $p(w)$ of string w is sum of weights of its derivations

In **weighted FA** every transition is given weight, etc.

Multiplication distributes over addition

(By definition) in a semiring: $c \cdot a + c \cdot b = c \cdot (a + b)$

Essential for effectively computing relevant properties

E.g. what is sum of weights of all derivations in:

$$\begin{array}{lcll} S & \rightarrow & A\ a\ B & [0.3] \\ S & \rightarrow & B\ b & [0.4] \\ A & \rightarrow & c\ B\ c & [0.5] \\ A & \rightarrow & d & [0.1] \\ B & \rightarrow & e & [0.3] \\ B & \rightarrow & f & [0.4] \end{array}$$

We could enumerate all derivations, and sum their weights, or:

- for B sum 0.3 + 0.4 = **0.7**
- for A sum 0.5 * **0.7** + 0.1 = **0.45**
- for S sum 0.3 * **0.45** * **0.7** + 0.4 * **0.7** = 0.3745

Partition function

Partition function Z maps each nonterminal A to sum of weights of all sub-derivations from A

As computed on previous slide for B, A, S

Less easy if we have recursion

In general, we have system of polynomial equations

E.g.

$$\left.\begin{array}{lcll} S & \rightarrow & A\ a\ S & [0.3] \\ S & \rightarrow & B\ b & [0.4] \\ A & \rightarrow & c\ B\ c & [0.5] \\ A & \rightarrow & d & [0.1] \\ B & \rightarrow & S & [0.3] \\ B & \rightarrow & b & [0.2] \end{array}\right\} \left\{\begin{array}{l} Z(S) = 0.3 \cdot Z(A) \cdot Z(S) + 0.4 \cdot Z(B) \\ Z(A) = 0.5 \cdot Z(B) + 0.1 \\ Z(B) = 0.3 \cdot Z(S) + 0.2 \end{array}\right.$$

Partition function as least fixed-point

Before values of Z are known, there is a system with variables

E.g.

$x_S = 0.3 \cdot x_A \cdot x_S + 0.4 \cdot x_B$
$x_A = 0.5 \cdot x_B + 0.1$
$x_B = 0.3 \cdot x_S$

Write $\vec{x}$ for a vector listing all such variables

The system can be written as $\vec{x} = F(\vec{x})$

The values of Z are the smallest (non-negative) solution to $\vec{x} = F(\vec{x})$, which is $\lim_{k\to\infty} F^k(0,\ldots,0)$

(Use Kleene's fixed-point theorem, considering F is monotone and continuous)

Computation of Z

This suggests computation as:

$\vec{x}^{(0)} = (0,\ldots,0)$
$\vec{x}^{(k+1)} = F(\vec{x}^{(k)})$

Halt at smallest k such that $\vec{x}^{(k+1)} = \vec{x}^{(k)}$

Note that $\vec{x}^{(k)}$ has physical interpretation:

"sum of weights of subderivations of height at most k"

In practice this naive iteration is very slow

Faster computation uses Newton's algorithm (not discussed here)

In any case, partition grammar into maximal sets of mutually recursive nonterminals (Lecture 2) for faster processing

Outer values

$Z(A)$ can be called **inner** value of A

Auxiliary definitions:

$Z(X\alpha) = Z(X) \cdot Z(\alpha)$ and $Z(a) = 1$ for any terminal a

There is also **outer** value of A

Is sum of weights of all derivations of sentential forms wAv, some terminal strings w and v

Again system of equations:

$outer(A) = \delta(A = S) + \sum_{\rho=(B\to\alpha A\beta)} outer(B) \cdot p(\rho) \cdot Z(\alpha\beta)$

where $\delta(A = S)$ is 1 if $A = S$ and 0 otherwise

After filling in Z values, linear system of equations

Example

$$\begin{array}{lcll} S & \to & A\ a\ B & [0.3] \\ S & \to & B\ b & [0.4] \\ A & \to & c\ B\ c & [0.5] \\ A & \to & d & [0.1] \\ B & \to & e & [0.3] \\ B & \to & f & [0.4] \end{array}$$

We saw before $Z(B) = 0.7$, $Z(A) = 0.45$, $Z(S) = 0.3745$

$outer(S) = 1$

$outer(A) = outer(S) \cdot 0.3 \cdot Z(aB) = 0.21$

$$\begin{aligned} outer(B) = & \ outer(S) \cdot 0.3 \cdot Z(Aa) + \\ & \ outer(S) \cdot 0.4 \cdot Z(b) + \\ & \ outer(A) \cdot 0.5 \cdot Z(cc) \ = \ldots \text{ (easy exercise)} \end{aligned}$$

Use of outer values

Assume fixed weighted CFG

Let $\mathcal{D}$ be set of derivations of terminal strings

Let $|d|_A$ be number of times that an A is rewritten in derivation d

Let $|d|_\rho$ be number of times that rule ρ is used in d

Obviously $|d|_A$ is sum of $|d|_\rho$ for different rules ρ with left-hand side A

By distributivity, we obtain for A and $\rho = A \to \alpha$:

$F(A) = \sum_{d \in \mathcal{D}} p(d) \cdot |d|_A = outer(A) \cdot Z(A)$

$F(\rho) = \sum_{d \in \mathcal{D}} p(d) \cdot |d|_\rho = outer(A) \cdot p(\rho) \cdot Z(\alpha)$

If p is probability distribution over $\mathcal{D}$ then F is expected frequency

2 intersection

Intersection of CFGs and FAs

Closure of CFLs under intersection with regular languages

Given CFG $\mathcal{G}$ and FA $\mathcal{M}$, there is CFG $\mathcal{G}_\cap$ with $L(\mathcal{G}_\cap) = L(\mathcal{G}) \cap L(\mathcal{M})$

Constructive proof (Bar-Hillel & Perles & Shamir, 1964)

1. For each $A \to X_1 \cdots X_k$ from $\mathcal{G}$

2. And any sequence of states $s_0, \ldots, s_k$ from $\mathcal{M}$
3. $\mathcal{G}_\cap$ will contain

$$(s_0, A, s_k) \to (s_0, X_1, s_1) \cdots (s_{k-1}, X_k, s_k)$$

1. For each a transition (s, a, s')
2. $\mathcal{G}_\cap$ will contain $(s, a, s') \to a$

Start symbol $S_\cap = (q_0, S, q_f)$

Weighted intersection

Suppose $\mathcal{G}$ is now weighted CFG

Then let

$$(s_0, A, s_k) \to (s_0, X_1, s_1) \cdots (s_{k-1}, X_k, s_k)$$

adopt weight from $A \to X_1 \cdots X_k$

Let

$$(s, a, s') \to a$$

have weight 1

Weighted intersection (cont.)

If $\mathcal{M}$ is deterministic, then:

$$\begin{array}{ll} p_\cap(w) = p(w) & \text{if } w \in L(\mathcal{M}) \\ p_\cap(w) = 0 & \text{otherwise} \end{array}$$

Because:

- A derivation in $\mathcal{G}_\cap$ is an 'enhanced' derivation from $\mathcal{G}$, checking that a string in $\mathcal{M}$ is derived
- Determinism implies that there is at most one 'enhanced' derivation for each original derivation

Hence

$$\sum_{w \in L(\mathcal{M})} p(w) = \sum_v p_\cap(v)$$

Weighted intersection (cont.)

If $\mathcal{M}$ is deterministic and accepts exactly one string w, then:

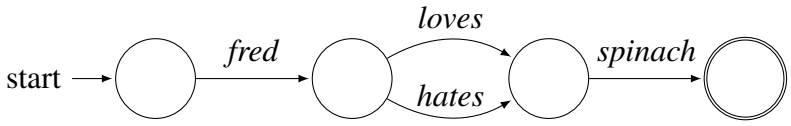
$$p(w) = \sum_v p_\cap(v)$$

So to determine weight of string w by weighted CFG:

- construct linear FA accepting only w,
- compute intersection grammar, and
- compute Z value of its start symbol

Example: PCFG $\mathcal{G}$ and DFA $\mathcal{M}$

S	$\rightarrow$	*N VP*	$[0.6]$
S	$\rightarrow$	*N*	$[0.1]$
S	$\rightarrow$	*S and S*	$[0.3]$
VP	$\rightarrow$	*V N*	$[1.0]$
N	$\rightarrow$	*fred*	$[0.4]$
N	$\rightarrow$	*spinach*	$[0.3]$
N	$\rightarrow$	*haggis*	$[0.3]$
V	$\rightarrow$	*loves*	$[0.7]$
V	$\rightarrow$	*hates*	$[0.3]$



Example: PCFG $\mathcal{G}_\cap$

$$
\begin{array}{lcll}
(0, S, 3) & \rightarrow & (0, N, 1)\ (1, VP, 3) & [0.6] \\
(1, VP, 3) & \rightarrow & (1, V, 2)\ (2, N, 3) & [1.0] \\
(0, N, 1) & \rightarrow & (0, \textit{fred}, 1) & [0.4] \\
(1, V, 2) & \rightarrow & (1, \textit{loves}, 2) & [0.7] \\
(1, V, 2) & \rightarrow & (1, \textit{hates}, 2) & [0.3] \\
(2, N, 3) & \rightarrow & (2, \textit{spinach}, 3) & [0.3]
\end{array}
$$

$$
\begin{array}{lcll}
(0, \textit{fred}, 1) & \rightarrow & \textit{fred} & [1.0] \\
(1, \textit{loves}, 2) & \rightarrow & \textit{loves} & [1.0] \\
(1, \textit{hates}, 2) & \rightarrow & \textit{hates} & [1.0] \\
(2, \textit{spinach}, 3) & \rightarrow & \textit{spinach} & [1.0]
\end{array}
$$

Sum of probabilities of derivations/strings: $0.6 * 0.4 * (0.7 + 0.3) * 0.3$

3 parameter estimation

Parameter estimation

For probabilistic grammars, how to choose probabilities?

Count how often nonterminals occur and how often rules occur, in parsed sample of language use (treebank)

Set $p(A \rightarrow \alpha) = \frac{C(A \rightarrow \alpha)}{C(A)}$

where C is the 'count' in the treebank

Form of **relative frequency estimation**

Obtained PCFG maximises likelihood of treebank

Similarly for probabilistic FAs: ratio of counts of transitions (s, a, s') and states s

Unsupervised training

Suppose only strings $w_1, \ldots, w_n$ are given (no trees)

Then we can improve existing PCFG $\mathcal{G}$:

- Let $\mathcal{G}_i$ be intersection of $\mathcal{G}$ and w_i $(1 \leq i \leq n)$
- For each $\rho_{s_0,\ldots,s_k} = (s_0, A, s_k) \rightarrow (s_0, X_1, s_1) \cdots (s_{k-1}, X_k, s_k)$ in $\mathcal{G}_i$, compute its F-value in $\mathcal{G}_i$ (by $outer$, p and Z) and call this $F_i(\rho_{s_0,\ldots,s_k})$
- Consider $\rho = A \rightarrow X_1 \cdots X_k$ and let $F_i(\rho)$ the sum of $F_i(\rho_{s_0,\ldots,s_k})$ for different choices of $s_0, \ldots, s_k$

- Normalise $E_i(\rho) = F_i(\rho) \;/\; p(w_i)$
- Let $E(\rho) = \sum_i E_i(\rho)$

Inside-outside algorithm

So we now have estimate of number of occurrences of rules in training corpus

And thereby $E(A) = \sum_{A \to \alpha} E(A \to \alpha)$

This means we can revise the probability of a rule: $p'(A \to \alpha) = \frac{E(A \to \alpha)}{E(A)}$

If we replace p in given PCFG by p' then likelihood of training corpus increases

. . . and this can be repeated until convergence (local optimum)

Known as **inside-outside algorithm**

Example of **EM (expectation-maximisation) algorithm**

4 approximation

How to approximate PCFG by PFA?

Suppose PCFG is given

Take underlying CFG

Approximate it by FA (by any of the methods from Lecture 2)

(Make sure FA is determinized)

Now estimate probabilities of PFA to minimize distance to PCFG

To be solved:

- how to estimate PFA
- how to measure distance between probability distributions
- and prove that the estimated PFA is closest

How to estimate PFA?

Again, we could like to use something like relative frequency estimation

But instead of count of transition in some corpus, we have expected frequency of transition in strings generated by PCFG $\mathcal{G}$

Let τ be transition in DFA $\mathcal{M}$ and $w \in L(\mathcal{G}) \cap L(\mathcal{M})$

Then let $|w|_\tau$ be number of occurrences of τ in (unique) computation of FA accepting w

The expected frequency of τ is

$$E(\tau) = \sum_{w \in L(\mathcal{G}) \cap L(\mathcal{M})} p(w) \cdot |w|_\tau$$

How to estimate PFA? (cont.)

Now consider the intersection grammar $\mathcal{G}_\cap$ constructed from the PCFG and the DFA

Because of the way that transitions are part of 'enhanced' derivations, we have for (transition/nonterminal) $\tau = (s, a, s')$:

$$E(\tau) = \sum_d p_\cap(d) \cdot |d|_\tau$$

Based on what we have seen before, we conclude:

$$E(\tau) = outer(\tau) \cdot Z(\tau) = outer(\tau)$$

How to estimate PFA? (cont.)

$E(s) = \sum_{\tau=(s,a,s')} E(\tau)$

We set (by relative frequency estimation):

$$p(\tau) = \frac{E(\tau)}{E(s)}$$

With this p we obtain PFA

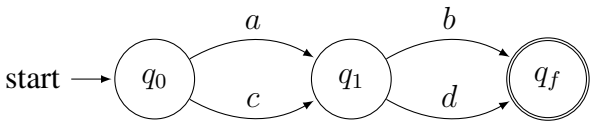
Later we will discuss:

This is best PFA that has given underlying FA, i.e. closest to the PCFG

Example

PCFG $\mathcal{G}$: $S \rightarrow a\ b$ $[1/3]$
$S \rightarrow c\ d$ $[2/3]$

DFA $\mathcal{M}$:



$\mathcal{G}_\cap$: $(q_0, S, q_f) \rightarrow (q_0, a, q_1)\ (q_1, b, q_f)$ [1/3]
$(q_0, S, q_f) \rightarrow (q_0, c, q_1)\ (q_1, d, q_f)$ [2/3]
$(q_0, c, q_1) \rightarrow c$ [1]
$\vdots$

E.g.: $p((q_0, a, q_1)) = \frac{outer((q_0,a,q_1))}{outer((q_0,a,q_1))+outer((q_0,c,q_1))} = \frac{\frac{1}{3}}{\frac{1}{3}+\frac{2}{3}} = \frac{1}{3}$

How to measure distance between probability distributions?

A **probability distribution** (over Σ^*) is a function p satisfying $\sum_w p(w) = 1$

Kullback-Leibler (KL) distance between two probability distributions p and q:

$$D(p\|q) = \sum_w p(w) \cdot \log_2 \frac{p(w)}{q(w)}$$

Also known as **relative entropy**

$D(p\|q) \geq 0$

$D(p\|q) = 0$ iff $p = q$

Generally $D(p\|q) \neq D(q\|p)$ so not a metric

Application of KL distance

We are not worried about $p(w) = 0$ in:

$$D(p\|q) = \sum_w p(w) \cdot \log_2 \frac{p(w)}{q(w)}$$

as by convention $0 \log_2 \frac{0}{q} = 0$, any q

It is a concern however if $q(w) = 0$ and $p(w) > 0$, some w

Let p be according to PCFG $\mathcal{G}$

Let q be according to approximating PFA $\mathcal{M}$

What if $w \in L(\mathcal{G})$ but $w \notin L(\mathcal{M})$?

(Can only happen with subset approximation)

Application of KL distance (cont.)

Remove $w \in L(\mathcal{G}) \setminus L(\mathcal{M})$ from consideration

And normalise to obtain probability distribution again

$$p'(w) = \begin{cases} \frac{p(w)}{Z} & \text{if } w \in L(\mathcal{M}) \\ 0 & \text{otherwise} \end{cases}$$

where $Z = \sum_{w:w \in L(\mathcal{M})} p(w)$

Out of all PFAs for given underlying FA, the one we have constructed has lowest value of $D(p' \| q)$

Proof in:

Nederhof, *Computational Linguistics* 31(2), pp. 173-185, 2005