1 Math

Semiring

$(\mathcal{A},\oplus,\otimes,\bar{0},\bar{1})$

- $(\mathcal{A},\oplus,\bar{0})$ is a commutative monoid: $a\oplus\bar{0}=a$, $a\oplus(b\oplus c)=(a\oplus b)\oplus c$
- $(\mathcal{A},\otimes,\bar{1})$ is a monoid: $a\otimes\bar{1}=a$
- $\otimes$ distributes over $\oplus$: $(a\oplus b)\otimes c=(a\otimes c)\oplus(b\otimes c)$
- $\bar{0}$ is an annihilator for $\otimes$: $a\otimes\bar{0}=\bar{0}$

Commutative SR: $a\otimes b=b\otimes a$ *Idempotent SR*: $a\oplus a=a$

- For idempotent SRs, $\bigoplus_{k=0}^{K} M^k=(I+M)^K$

Closed SR: *Kleene star* *:

- $x^*=\bigoplus_{n=0}^{\infty}x^{\otimes n}=\bar{1}\oplus x\oplus x^{\otimes 2}\oplus x^{\otimes 3}\oplus\cdots$
- Properties: $x^*=\bar{1}\oplus x\otimes x^*=\bar{1}\oplus x^*\otimes x$

$\bar{0}$-closed SR: $\bar{1}\oplus a=\bar{1}$ (incl. tropical and arctic SR)

- $x^*=\bigoplus_{n=0}^{N-1}x^{\otimes n}$ since cycles in a path of length $\geq N$ do not contribute
- Idempotent

Common SRs:

- *Boolean SR*: $(\{0,1\},\vee,\wedge,0,1)$
- *Inside SR*: $(\mathbb{R}\cup\{\infty\},+,\times,0,1)$
- *Log-sum-exp SR*: $(\mathbb{R}\cup\{-\infty\},\oplus_{\log},+,-\infty,0)$ where $a\oplus_{\log}b=\log(e^a+e^b)$
- *Viterbi SR*: $([0,1],\max,\times,0,1)$
- *Arctic SR*: $(\mathbb{R}\cup\{-\infty\},\max,+,-\infty,0)$

Probability Distributions

Normal distribution — $X\sim\mathcal{N}(\mu,\sigma^2)$, for univariate PDF: $\frac{1}{\sigma\sqrt{2\pi}}exp(\frac{-(x-\mu)^2}{2\sigma^2})=\frac{1}{\sigma\sqrt{2\pi}}exp(-x^2\frac{1}{2\sigma^2}+2x\frac{\mu}{2\sigma^2}-\frac{\mu^2}{2\sigma^2})$, for multivariate PDF: $\frac{1}{2\pi^{n/2}}\frac{1}{|\Sigma|^{1/2}}exp(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu))$

Exponential families —

$p(x|\theta)=\frac{1}{Z(\theta)}h(x)\exp(\theta\cdot\Phi(x))$

Gaussian:

- $\frac{1}{Z(\theta)}=\frac{e^{-\frac{\mu^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$
- $h(x)=1$
- $\theta=\begin{bmatrix}\mu/\sigma^2\\-1/2\sigma^2\end{bmatrix}$
- $\Phi(x)=\begin{bmatrix}x\\x^2\end{bmatrix}$

Beta:

- $Z(\theta)=B(\theta_1+1,\theta_2+1)$
- $h(x)=1$
- $\theta=\begin{bmatrix}(\alpha-1)\\(\beta-1)\end{bmatrix}$
- $\Phi(x)=\begin{bmatrix}\ln(x)\\\ln(1-x)\end{bmatrix}$

Binomial:

- $Z(\theta)=e^{n\ln(1+e^{\theta})}$
- $h(x)=\binom{n}{x}$
- $\theta=[\ln(\frac{\pi}{1-\pi})]$
- $\Phi(x)=[x]$

Hypothesis Testing

- Errors:
  - *TP*: Chose $H_0|H_0$
  - *FN, type I*: Chose $H_A|H_0$
  - *TN*: Chose $H_A|H_A$
  - *FP, type II*: Chose $H_0|H_A$
- *Significance level*: $\alpha=p(\text{type I error})=p(\bar{x}\geq c|H_0)$, *Beta*: $\beta=p(\text{type II error})$
- *Critical value*: Critical value c resp. z-score associated with α
- *P-value*: Probability, given H_0 that we observe a value as or more extreme as the observed value z_n: $p=P(|z|\geq z_n)$, smallest significance level, at which we can reject H_0 given the sample observed
- *Confidence level*: $1-\alpha$, *Power*: $1-\beta=p(\bar{x}\geq c|H_1)$
- *Test statistic*: $z_n|H_0=\frac{\bar{x}-\mu_0}{\sigma/\sqrt{n}}$

Other

Geometric series — $S=\sum_{i=0}^{\infty}a_i r^i=\frac{a_1}{1-r}$ for $r<1$

Quotient derivative rule — $\frac{\partial(\frac{f}{g})}{\partial x}=\frac{g\times\frac{\partial f}{\partial x}-f\times\frac{\partial g}{\partial x}}{g^2}$

2 ML Paradigms

MLE estimator

- Maximizes *log likelihood*: $\hat{\theta}=\arg\max_\theta(L)=\prod_{i=1}^n p(y_i|x_i,\theta)=\sum_{i=1}^n log(p(y_i|x_i,\theta))$
- For binary classification, *log loss = binary cross entropy*

NLP-specific model taxonomy

- Locally normalized:
  - $p(y_i|\mathbf{x})=\frac{\exp(\text{score}(\mathbf{x},y_i))}{\sum_{y'}\exp(\text{score}(\mathbf{x},y'))}$
  - $p(\mathbf{y}|\mathbf{x})=\prod_{i=1}^{|\mathbf{y}|}p(y_i|\mathbf{x})$
- Globally normalized:
  - $p(\mathbf{y}|\mathbf{x})=\frac{\exp(\text{score}(\mathbf{x},\mathbf{y}))}{\sum_{\mathbf{y}'\in\mathcal{Y}}\exp(\text{score}(\mathbf{x},\mathbf{y}'))}$

3 Estimating Distributions

Gaussian

MLE —

- Likelihood: $L=(\frac{1}{\sigma})^n\prod_{i=1}^n exp(-\frac{1}{2\sigma^2}(x^{(i)}-\mu)^2)=\frac{1}{\sigma^n}exp(-\frac{1}{2\sigma^2}\sum_{i=1}^n(x^{(i)}-\mu)^2)$
- Log likelihood: $LL=-nlog(\sigma)-\sum_{i=1}^n(\frac{1}{2\sigma^2}(x^{(i)}-\mu)^2)$
- $\mu_{MLE}=\frac{1}{n}\sum_{i=1}^n x^{(i)}$
- $\sigma^2_{MLE}=\frac{1}{n}\sum_{i=1}^n(x^{(i)}-\mu)^2$

Bayesianism —

- Let σ^2 be known, Posterior $p(\mu|x,\mu_0,\sigma_0^2)\propto$ Likelihood $p(x|\mu,\sigma^2)\times$ Prior $p(\mu|\mu_0,\sigma_0^2)$
- Expanding posterior, we see (based on the form of the Gaussian): $(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2})\mu^2=\frac{1}{2\sigma_n^2}\mu^2$ and $(\frac{\sum_{i=1}^n x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2})\mu=\frac{\mu_n}{\sigma_n^2}\mu$. Then: $\mu_n=\frac{n\bar{x}\sigma_0^2+\mu_0\sigma^2}{n\sigma_0^2+\sigma^2}$ and $\sigma_n^2=\frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2}$
- $p(\mu|x,\mu_0,\sigma_0^2)\sim\mathcal{N}(\mu_n,\sigma_n^2)$
- Conjugate prior

4 Linear Regression

Formulation

$y^{(i)}=\beta\cdot x^{(i)}$ resp. $y=X\beta$ where $X\in n\times m$

Possible transformation: Φ with rows $\phi(x^{(i)})^\top$

Optimization

Find parameters β

Objective function — Ordinary least squares estimator (OLSE): Minimize MSE: $LO=(y-X\beta)^\top(y-X\beta)$

Optimization —

- $\nabla_\beta LO=\frac{1}{2}\nabla_\beta(\beta^\top X^\top X\beta-2y^\top X\beta)=X^\top X\beta-X^\top y=0$
- $\Rightarrow\beta=(X^\top X)^{-1}X^\top y$

Ridge (ℓ_2) Regression

Find parameters β subject to $\|\beta\|^2-t\leq 0$

Objective function — Lagrangian: $LO=(y-X\beta)^\top(y-X\beta)+\lambda(\|\beta\|^2-t)$

Optimization — $\beta=(X^\top X+\lambda I)^{-1}X^\top y$

Characteristics —

- Strictly convex with global minimum, unique solution (linearly independent columns), and analytic solution (always invertible)
- Corresponds to Bayesian MAP estimator, when λ is chosen as $\frac{\sigma^2}{\tau^2}$: Posterior $p(\beta|X,y)\propto$Likelihood $p(y|X,\beta)\sim\mathcal{N}(X\beta,\sigma^2 I_n)\times$ Prior $p(\beta)\sim\mathcal{N}(0,\tau^2 I_m)$

4.1 Lasso (ℓ_1) Regression

Subject to $|\beta|-t\leq 0$

Objective function — Lagrangian: $LO=\ldots+\lambda(|\beta|-t)$

Characteristics — Corresponds to Bayesian MAP estimator, when λ is chosen as $\frac{\sigma^2}{b}$

5 Log Linear Models

Formulation

$p(y|x,\theta)=\frac{\exp(\theta\cdot f(x,y))}{\sum_{y'}\exp(\theta\cdot f(x,y'))}$

Optimization

Find parameters θ

Objective function —

- Minimize log loss: $\theta=\arg\min_\theta-\sum_{i=1}^n\log(p(y^{(i)}|x^{(i)},\theta))=-\sum_{i=1}^n\theta\cdot f(x^{(i)},y^{(i)})-\log(\sum_{y'}\exp(\theta\cdot f(x^{(i)},y')))$

Optimization — First-order derivative:

- Gradient: $\frac{\partial}{\partial\theta}\log\text{loss}=-\sum_{i=1}^n(f(x^{(i)},y^{(i)})-\frac{\sum_{y'}\exp(\theta\cdot f(x^{(i)},y'))\times f(x^{(i)},y')}{\sum_{y'}\exp(\theta\cdot f(x^{(i)},y'))})=-\sum_{i=1}^n(f(x^{(i)},y^{(i)})-\sum_{y'}p(y'|x^{(i)},\theta)\times f(x^{(i)},y'))=-\sum_{i=1}^n f(x^{(i)},y^{(i)})+\sum_{i=1}^n\mathbb{E}(f(x^{(i)},y'))$
- If we set gradient to 0, we have *expectation matching* (observed features = expected features)

Second-order derivative:

- $\frac{\partial^2}{\partial\theta\partial\theta^\top}\log\text{loss}=\frac{\partial}{\partial\theta^\top}\nabla\log\text{loss}=\sum_{i=1}^n\mathbb{E}[f(x^{(i)},y')f(x^{(i)},y')^\top]-\sum_{i=1}^n(\mathbb{E}[f(x^{(i)},y')]\mathbb{E}[f(x^{(i)},y')])=\sum_{i=1}^n Cov(f(x^{(i)},y'))$

6 Logistic

Formulation

- *Sigmoid*: $\sigma(z)=\frac{1}{1+e^{-z}}$
- $P(y=1|x)=\frac{1}{1+e^{-\beta\cdot x}}=\frac{e^{\beta\cdot x}}{1+e^{\beta\cdot x}}$, $P(y=0|x)=\frac{1}{1+e^{\beta\cdot x}}=\frac{e^{-\beta\cdot x}}{1+e^{-\beta\cdot x}}$
- *Odds*: $\frac{P(y=1|x)}{P(y=0|x)}=e^{\beta\cdot x}$, *Log odds*: $ln(\frac{P(y=1|x)}{P(y=0|x)})=\beta\cdot x$
- Geometrically, $z=\beta\cdot x$ defines a linear separating hyperplane: When $z>0$ resp. log-odds >0, then odds >1 resp. $P(y=1|x)>P(y=0|x)$, then predict $y=1$, otherwise $y=0$

Optimization

Find parameters β

Objective function —

- Likelihood: $L=\prod_{i=1}^n\sigma(z^{(i)})^{y^{(i)}}(1-\sigma(z^{(i)}))^{1-y^{(i)}}$
- Log-likelihood: $LL=\sum_{i=1}^n[y^{(i)}\log\sigma(z^{(i)})+(1-y^{(i)})\log(1-\sigma(z^{(i)}))]=\sum_{i=1}^n[y^{(i)}\log\frac{1}{1+e^{-z^{(i)}}}+(1-y^{(i)})\log\frac{e^{-z^{(i)}}}{1+e^{-z^{(i)}}}]=\sum_{i=1}^n[y^{(i)}z^{(i)}-\log(1+e^{z^{(i)}})]$

Optimization —

- $\frac{\partial-LL}{\partial\beta}=-\sum_{i=1}^n\frac{\partial}{\partial\beta}[y^{(i)}\log\sigma(z^{(i)})+(1-y^{(i)})\log(1-\sigma(z^{(i)}))]=\sum_{i=1}^n[\sigma(z^{(i)})-y^{(i)}]x^{(i)}$
- If we set gradient to 0, we have *expectation matching*

7 Multinomial Logistic

Formulation

Softmax: $P(y=k|x)=\frac{e^{f_i(x)/T}}{\sum_{j=1}^k e^{f_j(x)/T}}=\frac{e^{\beta_k\cdot x/T}}{\sum_{j=1}^k e^{\beta_j\cdot x/T}}$

Proof:

- Take class k as reference class
- Based on log-odds: $\sum_{i=1}^{k-1}(\frac{P_y(y=i|x)}{P_y(y=k|x)})=\frac{1-P_y(y=k|x)}{P_y(y=k|x)}=\sum_{i=1}^{k-1}\exp(f_i(x)-f_k(x))\Rightarrow P_y(y=i|x)=1-\frac{1}{1+\sum_{i=1}^{k-1}\exp(f_i(x)-f_k(x))}=\frac{\exp(f_i(x))}{\sum_{j=1}^k\exp(f_j(x))}$

Softmax (T=1) vs. argmax (T=0)

Optimization

Find parameters $\beta_1,\ldots,\beta_k$

Objective function —

- Likelihood: $L=\prod_{i=1}^n\prod_{\ell=1}^k(\frac{e^{\beta_\ell\cdot x^{(i)}}}{\sum_{j=1}^k e^{\beta_j\cdot x^{(i)}}})^{\delta\{y^{(i)}=\ell\}}$
- Log likelihood: $LL=\sum_{i=1}^n\sum_{\ell=1}^k\delta\{y^{(i)}=\ell\}[\beta_\ell\cdot x^{(i)}-\log(\sum_{j=1}^k e^{\beta_j\cdot x^{(i)}})]$

Optimization —

- $\frac{\partial-LL}{\partial\beta_k}=-\sum_{i=1}^n\delta\{y^{(i)}=k\}x^{(i)}-P(y=k|x^{(i)})x^{(i)}$
- If we set gradient to 0, we have expectation matching
- For reference: Softmax derivative $\frac{\partial P(y=\ell|x)}{\partial\beta_k}$: If $\ell=k\Rightarrow P(y=\ell|x)(1-P(y=\ell|x))x$, if $\ell\neq k\Rightarrow-P(y=\ell|x)P(y=k|x)x$, can be shown via quotient rule

8 Neural Networks

Formulation

Formulation — Model architecture: Inputs: n words > Embedding $e(w_i)$ > Concatenation: $e(x)=\frac{1}{n}\sum_{w_i}e(w_i)$ > Hidden layer: $h^{(k)}=\sigma(w^{(k)}h^{(k-1)})$ > Activation: Softmax: $p(y|x)=\frac{\exp(h_y^{(K)})}{\sum_{y'}\exp(h_{y'}^{(K)})}$

Neuron (j) in layer $[k]$: $h^{(j)[k]}=\varphi(h^{(i)[k-1]}\cdot\beta^{(j)[k]})$

Activation functions —

Sigmoid:

- Convex
- $[0,1]$
- $\varphi(z)=\sigma(z)=\frac{1}{1+e^{-z}}=\frac{e^z}{e^z+1}$
- $\varphi'(z)=\frac{e^{-z}}{(1+e^{-z})^2}$ with maximum at 0.25

Hyperbolic tangent:

- Not convex
- $[-1,1]$
- $\varphi(z)=tanh(z)=\frac{e^z-e^{-z}}{e^z+e^{-z}}=\frac{1-e^{-2z}}{1+e^{-2z}}$
- $\varphi'(z)=1-tanh(z)^2$

ReLU:

- Piecewise convex
- $\varphi(z)=max(0,z)$
- $\varphi'(z)=1$ if $z>0$;0 otherwise

Optimization

- Forward pass to calculate loss
- Backpropagation to calculate gradient: $\frac{\partial L}{\partial B^{[k]}}=\frac{\partial L}{\partial H^{[l]}}\frac{\partial H^{[l]}}{\partial B^{[k]}}=C$
  - When $l>k+1$, i.e. when going several layers back: $\frac{\partial L}{\partial B^{[k]}}=\frac{\partial L}{\partial H^{[l]}}\frac{\partial H^{[l]}}{\partial S^{[l-1]}}\frac{\partial S^{[l-1]}}{\partial H^{[l-1]}}\frac{\partial H^{[l-1]}}{\partial B^{[k]}}$
  - When $l=k+1$, i.e. when going one layer back: $\frac{\partial L}{\partial B^{[k]}}=\frac{\partial L}{\partial H^{[k+1]}}\frac{\partial H^{[k+1]}}{\partial S^{[k]}}\frac{\partial S^{[k]}}{\partial B^{[k]}}$
- Gradient descent to find best weights

9 Backpropagation

Point of Departure

Computation graph $\mathcal{G}$ — With n input nodes and $|E|=M$ edges, we have $|V|=M+n$ total nodes

Note: For exponentiated terms, e.g. $x^{a+b}=exp(log(x^{a+b}))=exp((a+b)\times log(x))$

Bauer's Formula —

- $P(j,i)$ is set of all paths from node $j\to i$
- $\frac{\partial z_i}{\partial z_j}=\sum_{p\in P(j,i)}\prod_{(k,l)\in p}\frac{\partial z_l}{\partial z_k}$
- Challenge: Runtime $O(|P(j,i)|)=O(2^{|E|})$

Forward Pass

Randomly initialize values of input nodes and then evaluate function

- Runtime $O(|E|)$, linear in edges
- Space $O(|V|)$, linear in vertices

Backpropagation

1. Perform forward pass
2. For $i=M$: $\frac{\partial f}{\partial z_i}=1$
3. For $i=M-1,\ldots,1$: $\frac{\partial f}{\partial z_i}=\sum_{j:i\in\text{Pa}(i)}\frac{\partial f}{\partial z_j}\frac{\partial z_j}{\partial z_i}$
4. Return $[\frac{\partial f}{\partial z_1},\frac{\partial f}{\partial z_2},\ldots,\frac{\partial f}{\partial z_M}]$

Properties:

- Partial derivatives on multiple paths are memoized
- For first-order derivative runtime and space same as forward pass, for k^{th} order derivative runtime $O(|E|n^{k-1})$

Requirements:

- Weights need to be initialized to different values (not 0 or same constant)
- At least one activation must be non-linear so that there is a non-zero gradient

10 RNNs

Formulation

Model architecture: Input > Hidden layer resp. *memory cell*: *Cell state* h_t, *cell output* y_t > Output

Output of neuron in layer n: $Y_t=\phi(X_t W_x+H_{t-1}W_y+b)V$

Optimization

Gradient:

- $\nabla_{W_h}L\propto\sum_{k=1}^t(\prod_{i=k+1}^t\frac{\partial h_i}{\partial h_{i-1}})\frac{\partial h_k}{\partial W_k}$
- $\frac{\partial h_{i+k}}{\partial h_i}=\prod_{j=0}^{k-1}\frac{\partial h_{i+k-j}}{\partial h_{i+k-j-1}}$

11 Attention

Approach

$K=EW^k$ resp. $k_i=e_i W^k$

- E $(n\times h)$
- W_k $(h\times d_k)$
- K $(n\times d_k)$
- k_i $(1\times d_k)$
- e_i $(1\times h)$

$V=EW^v$ resp. $v_i=e_i W^v$ where

- E $(n\times h)$
- W_v $(h\times d_v)$
- V $(n\times d_v)$
- v_i $(1\times d_v)$
- e_i $(1\times h)$

$Q=EW^q$ resp. $q_i=e_i W^q$

- E $(m\times h)$
- W_q $(h\times d_k)$
- Q $(m\times d_k)$
- q_i $(1\times d_k)$
- e_i $(1\times h)$

- *Similarity matrix*: $A=\sigma(\frac{QK^\top}{\sqrt{d_k}})$ in $(m\times n)$ resp. $\alpha_t=\sigma(\frac{q_t K^\top}{\sqrt{d_k}})$ resp. $\alpha_{ti}=\frac{exp(q_t\cdot k_i)}{\sum_{i'}exp(q_t\cdot k_{i'})}$
- *Attention-weighted embedding matrix*: $Z=AV$ in $(m\times d_v)$ resp. $z_t=\alpha_t V=\sum_i\alpha_{ti}v_i$
- In cross-attention: Q is decoder input with m, V,K are encoder outputs with n
- In self-attention: Q,V,K are all either encoder or decoder inputs with n or m
- In *multi-head attention*: MHAZ = Concat($Z_{\text{head}_1},\ldots,Z_{\text{head}_h}$)$W_O+b_O$ where
  - Concat$(\ldots)$ in $(m\times(n\times n_{heads}))$
  - W_O in $((n_{heads}\times n)\times d_v)$
  - b_O in $1\times d_v$

Further proofs

- Self-attention ($A=ZW_qW_k^\top Z^\top$) without positional encodings is permutation equivariant (Attention$\Pi Z=\Pi$Attention$Z$)
- Self-attention with learned Q and without positional encodings is permutation invariant (AttentionΠZ=Attention$Z$)

12 RNNs

Formulation

Model architecture:

- *Encoder*: $h_n^{(e)}=f(W_1^{(e)}h_{n-1}^{(e)}+W_2 w_n)$
- *Decoder*: $h_m^{(d)}=f(W_3^{(d)}h_{m-1}^{(d)}+W_2^{(d)}w'_{m-1}+W_1^{(d)}z_m)$ where
  - $z_m=\sum_{n=1}^N\alpha_{m,n}h_n^{(e)}$
  - $h_n^{(e)}=V$
  - $\alpha_{m,n}=$ softmax$(h_{m-1}^{(d)}\times[h_1^{(e)},\ldots,h_N^{(e)}])$ with $h_{m-1}^{(d)}=Q$ and $[h_1^{(e)},\ldots,h_N^{(e)}]^\top=K$

Runtime analysis:

- Encoder: $O(l_e Nd^2)$ from hidden states
- Decoder: $O(l_d Md^2+l_d dNM)$ from hidden states + cross-attention

13 Transformers

Formulation

Model architecture:

- *Encoder*:
  - $H_0^{(e)}=X+P\in(N\times d_{model})$
  - Calculate multi-head attention based on $H^{(e)}_{(l-1)}$
  - Addition and normalization: $H_l^{(e)}$ Layer Norm(MHA(Q,K,V)+$H^{(e)}_{(l-1)}$)
  - FFN($H_l^{(e)}$)= ReLU($H_l^{(e)}W_1+b_1$)W_2+b_2 where
    - $W_1\in\mathbb{R}^{(d\times r)}$
    - $b_1\in\mathbb{R}^{(1\times r)}$
    - $W_2\in\mathbb{R}^{(r\times d)}$
    - $b_2\in\mathbb{R}^{(1\times d)}$
  - Addition and normalization
- *Decoder*:
  - Target sequence inputs are fed into decoder with one time step lag:
    - $H_0^{(d)}=Y+P\in(M\times d_{model})$
    - Calculate masked self-attention based on $H^{(d)}_{(l-1)}$ where mask covers tokens in positions $m\geq t$
    - Addition and normalization on $H_l^{(d)}$
  - Encoder outputs are fed into decoder with cross-attention:
    - Calculate cross-attention with $H_l^{(d)}$ for Q and $H_{(N)}^{(e)}$ for K,V
    - Addition and normalization
  - FFN($H_l^{(d)}$)
  - Addition and normalization
- Linear layer applied to $h_M^{(d)}$
- Softmax layer applied to select token with highest probability
- Challenge: Output y_t is conditioned on entire history (non-Markovian) with runtime of $O(|\Sigma|^n)$
- Solution:
  - *Greedy decoding*
  - *Beam search*: Keep k-highest-probability tokens in memory (beam size) at each step, requires normalization at each step, e.g. if we have k=2, trigrams and vocabulary $\{a,b\}$. Let first probabilities be given by BOSBOSa=0.6,BOSBOSb=0.4, we keep both in memory.
    We then check local probabilities for BOSaa=0.7,... and calculate total probabilities BOSaa=0.7×0.6,... Thereof, we keep highest 2 options, i.e. BOSaa,BOSba.
    We then check local probabilities for aaa=0.7,baa=0.1,... and so on.

14 Skip Grams

Description

Predict context word given center word, generate word embeddings

Formulation

- $\mathcal{V}$ with words w, for which we consider T preceding and following words
- Set $\mathcal{D}$ of $O(T\times|\mathcal{V}|)$ pairs: $\{(w_i,w_t)\}$ where $w_i\in w$ is center, $w_t\in w'$ is context word

Optimization

Objective function —

- Bilinear softmax function
- Likelihood: $\prod_{(w_i,w_t)\in\mathcal{D}}p(w_t|w_i)=\prod_{(w_i,w_t)\in\mathcal{D}}\frac{\exp(e_{\text{wrd}}(w_i)\cdot e_{\text{ctx}}(w_t))}{\sum_{w'\in\mathcal{V}}\exp(e_{\text{wrd}}(w_i)\cdot e_{\text{ctx}}(w'))}$
- Challenge: Denominator has $2|\mathcal{V}|$ parameters
- Solution: Use *negative sampling*: For each pair (w_i,w_t), we randomly sample with replacement a set C^- from $\mathcal{V}$ and compute sigmoid, instead of softmax

Optimization — Use embeddings for w, throw away embeddings for w'

15 Language Models

Formulation

Prefix tree

- String y
- Challenge: *Globally normalized model*: Normalize over all $y\in\Sigma^*$, since Σ^* is infinite, Z is infinite
- Solution: *Locally normalized model*: Assume next word in string is only conditioned on preceding context, normalize over all words in vocabulary
- *Tight model*: Locally normalized model, sums to 1, and has finite length paths, enforced by $p(\text{EOS}|\cdots)>\delta>0$

16 N Grams

Formulation

- String w of length N
- Naively: $p(w)=p(w_1)\times p(w_2|w_1)\times p(w_3|w_1,w_2)\times\cdots\times p(w_N|w_{<N})\times p(\text{EOS}|w)$
- *Markov*: Only the last $n-1$ words are considered to predict next word
- *N-gram*:
  - Vocabulary with $|\mathcal{V}|$ words has $|\mathcal{V}|^n$ (distinct) N-grams
  - Sentence with M words has $M-n+1$ (potentially overlapping) N-grams
  - N-gram counts are given by all distinct N-grams and how often they appear in the sentence
  - $p(w_t|w_{t-n+1},\ldots,w_{t-1})$
- Then: $p(w)=(\prod_{t=n}^N p(w_t|w_{t-n+1},\ldots,w_{t-1}))\times p(\text{EOS}|w_{N-n+1},\ldots,w_N)$
- We have
  - $|\mathcal{V}|^{n-1}+|\mathcal{V}|^{n-2}+\ldots+|\mathcal{V}|+1=\sum_{i=0}^{n-1}|\mathcal{V}|^i$ conditional probabilities for w_t
  - $|\mathcal{V}|-1$ free parameters for each conditional probability

Optimization

Relative frequency counts —

- $p(w_t|\ldots)=\frac{\text{count}(w_{t-n+1},\ldots,w_{t-1},w_t)}{\text{count}(w_{t-n+1},\ldots,w_{t-1})}$
- OOV:
  - Given a sentence with $M+1$ words and vocabulary $\mathcal{V}$, probability that a given bigram does not appear in any of the M spots: $(1-\frac{1}{|\mathcal{V}|^2})^M$
  - Solution: *Smoothing*: $p(w_t|w_{t-n+1},\ldots,w_{t-1})=\frac{\text{count}(w_{t-n+1},\ldots,w_{t-1},w_t)+\lambda}{\text{count}(w_{t-n+1},\ldots,w_{t-1})+|\mathcal{V}|\lambda}$

Neural nets —

$p(w_t|\ldots)=\frac{\exp(v(w_t)\cdot h_t)}{\sum_{w'\in\mathcal{V}}\exp(v(w')\cdot h_t)}$ where $v(w)$ are word vectors, h_t are N-gram context vectors

Evaluation

Perplexity: perplexity$(w_1,\ldots,w_N)=\sqrt[N]{\prod_{i=1}^N\frac{1}{p(w_i|w_{i-n+1},\ldots,w_{i-1})}}=p(w_1,\ldots,w_N)^{-\frac{1}{N}}$

17 POS Tagging

Formulation

Trellis

Point of departure —

- $p(t|w)=\frac{\exp(\text{score}(t,w))}{\sum_{t'}\exp(\text{score}(t',w))}$
- Challenge: Z in $O(|T|^N)$ time
- Solution: Scoring function that is additively decomposable over tag bigrams: $\text{score}(t,w)=\sum_{n=1}^N\text{score}(\langle t_{n-1},t_n\rangle,w)$
- Then: $p(t|w)=\frac{\exp(\sum_{n=1}^N\text{score}(\langle t_{n-1},t_n\rangle,w))}{\sum_{t'}\exp(\sum_{n=1}^N\text{score}(\langle t'_{n-1},t'_n\rangle,w))}$

Backward resp. Viterbi algorithm —

1. For t_{N-1} (all tags):
   - $\beta(w,t_{N-1})$ resp. $v(w,t_{N-1})\leftarrow\exp(\text{score}(\langle t_{N-1},\text{EOS}\rangle,w))$
   - $b(t_{N-1})\leftarrow$EOS since t_N can only be EOS
2. For $n\in N-2,\ldots,1$: For $t_n\in T$ (all tags):
   - $\beta(w,t_n)$ resp. $v(w,t_n)\leftarrow\oplus_{t_{n+1}}[\exp(\text{score}(\langle t_n,t_{n+1}\rangle,w))\otimes v(w,t_{n+1})]$
   - $b(t_n)\leftarrow argmax_{t_{n+1}}[\exp(\text{score}(\langle t_n,t_{n+1}\rangle,w))\otimes v(w,t_{n+1})]$
3. Finally:
   - $\beta(w,t_0)$ resp. $v(w,t_0)\leftarrow\oplus_{t_1}[\exp(\text{score}(\langle t_0,t_1\rangle,w))\otimes v(w,t_1)]$
   - $b(t_0)\leftarrow argmax_{t_1}[\exp(\text{score}(\langle t_0,t_1\rangle,w))\otimes v(w,t_1)]$
4. For $n=1,\ldots,N$: Recover the best sequence using backpointers, by always plugging in next t $t_n\leftarrow b(t_{n-1})$, starting with $t_1=b(\text{BOS}),t_2=b(t_1),\ldots$
5. Return $t_{1:N}$ and $\beta(w,t_0)$ resp. $v(w,t_0)$

- $\beta(w,t_n)$ are *backward variables* that contain the sum of the scores of all paths starting at EOS and ending at tag t_n
- $v(w,t_n)$ are *Viterbi variables* that contain the score of t^* starting at EOS and ending with tag t_n
- $b(t_n)$ are *backpointers* that point to the t_{n+1} tag in t^*
- Runtime, potentially repeated for the number of epochs and samples:
  - For backpointers: $O(N)$
  - For tag N-grams: $O(N|T|^n)$
- Alternatively, the same can be achieved with a *forward Viterbi algorithm*
  - Starting from t_1 and going forward towards EOS t_N
  - Instead of looking at t_{n+1} we look at t_{n-1}
  - Backpointers point to the t_{n-1} tag in t^*, i.e. $t_{n-1}\leftarrow b(t_n)$, starting with $t_{N-1}\leftarrow b(\text{EOS}),t_{N-2}\leftarrow b(t_{N-1}),\ldots$

Extension of backward algorithm in the expectation SR:

- Instead of $\omega=\exp(\text{score}(\langle t_{n-1},t_n\rangle,w))$, we do: $\omega=\langle\omega,-\omega\log(\omega)\rangle$
- Running the backward algorithm with these weights and in the expectation SR, we compute $\langle Z,H_u\rangle$ where $H_u=-\sum_t\exp(\text{score}(t,w))\times\text{score}(t,w)$ is the unnormalized entropy
- We can then compute the normalized entropy $H_n=-\sum_t p(t)\log(p(t))$ as $Z^{-1}H_u+\log(Z)$

Generalized algorithm —

- Backward algorithm: Computes Z, uses inside SR
- Viterbi algorithm: Computes the score of t^* and recovers the sequence itself, uses Viterbi SR
- Challenge with this original formulation: Multiplying probabilities for long sequences may cause numbers to go to 0

- Solution: Log-sum-exp SR (backward algorithm) resp. arctic SR (Viterbi algorithm). Note: The log-sum-exp SR returns $log(Z)$ instead of Z

Scoring functions — Hidden Markov Model:
score$(\langle t_{n-1},t_n\rangle,w)=$ transition$(t_{n-1},t_n)+$emission$(t_n,w_n)=$ transition probability (tag-tag pairs) + emission probability (word-tag pairs) except for $t_N=$ EOS, where emission probability = 0

18 Syntax — Syntactic Description

- *Constituent*: Coherent unit resp. nodes in tree
- *Terminals*: Words, *Non-terminals*: Abstractions over words (e.g. NP, VP, etc.)
- *Grammar*: Production rules
- Can be represented as *parse tree*

Formulation *Probabilistic CFGs*

$(\mathcal{N},\mathcal{S},\Sigma,\mathcal{R},\mathcal{P})$ —
- Non-terminals $\mathcal{N}=\{N_1,N_2,\dots\}$
- Start non-terminal S
- Terminals $\Sigma=\{a_1,a_2,\dots\}$
- Production rules $r: N\to\alpha$, where N is non-terminal and $\alpha\in(\mathcal{N}\cup\Sigma)^*$
- Probabilities $\mathcal{P}$ for each rule, locally normalized over each transition: $\sum_k p(N\to\alpha_k)=1$ where $N\to\alpha_1,\dots,N\to\alpha_k$ are expansions of node N
- String s of length M

Probability of a tree: $p(t)=\prod_{r\in t}p(r)=p(S\to S_1S_2)^{M-1}\times p(S\to X)^M\times p(X\to\sigma)^M$ where:
- $p(S\to S_1S_2)^{M-1}$: repeats the rule $S\to SS$ $M-1$ times to obtain M starting nodes
- X: non-terminal, σ: terminal

Probability of a string: $p(s)=\sum_{t\in\mathcal{T}(s)}p(t)$

Chomsky Normal Form (CNF) —
- Production rules: $N_1\to N_2N_3$ are non-terminal productions, $N\to a$ are terminal productions, $S\to\varepsilon$
- Prohibits cyclic rules
- Transform CFG to CNF:
  - Remove $A\to\varepsilon$, e.g. if CFG contains $S\to AB|\varepsilon$, $B\to b|\varepsilon$, change to $S\to AB|A|\varepsilon$, $B\to b$
  - Remove $A\to B$, e.g. if CFG contains $S\to B$, $B\to b$, change to $S\to b$, $B\to b$
  - Convert long productions $A\to B_1\dots B_k$ with $k>2$ and mixed rules $A\to aB$ with intermediate non-terminals

Weighted CFGs —
- A more general formulation of PCFGs, globally normalized
- $p(t)=\frac{\prod_{r\in t}\exp(\text{score}(r))}{\sum_{t'\in\mathcal{T}}\prod_{r'\in t'}\exp(\text{score}(r'))}$
- Challenge: Z is infinitely large (Σ^*)
- Solution: $p(t|s)=\frac{\prod_{r\in t}\exp(\text{score}(r))}{\sum_{t'\in\mathcal{T}(s)}\prod_{r'\in t'}\exp(\text{score}(r'))}$
- Challenge: Z is still potentially infinitely large (cycles)
- Solution: Revert to CNF. Then, $|Z|$ is the number of rooted binary trees, i.e. Catalan number C_{M-1}
- Then: $p(t|s)=\frac{\prod_{N_i\to N_jN_k\in t}\exp(\text{score}(N_i\to N_jN_k))}{\sum_{t'\in\mathcal{T}(s)}\prod_{N_i\to N_jN_k\in t'}\exp(\text{score}(N_i\to N_jN_k))}\times\frac{\prod_{N_l\to a\in t}\exp(\text{score}(N_l\to a))}{\prod_{N_l\to a\in t'}\exp(\text{score}(N_l\to a))}$ consisting of non-terminal productions × terminal productions

Span —
- *Admissible* if $X\to w[i:j]$
- For tree where nodes expand along right diagonal of tree only, only 1 admissible span per span size: $[M-\text{span size}+1, M+1)$

Optimization *CKY Algorithm* —

- Can: Compute Z (inside SR or log-sum-exp SR), find best parse and its probability (Viterbi or arctic SR), determine if a given string is admissible by the grammar (Boolean SR)
- Requires grammar in CNF
- Chart$[i,j,X]$ is probability that non-terminal X generates the subtree resp. substring $s_i,\dots,s_j$

1. For $m=1,\dots,M$: Terminal productions: For $X\to s_m$: Chart$[m,m+1,X]\oplus=\exp(\text{score}(X\to s_m))$
2. For span $=2,\dots,M$: For $i=1,\dots,M-\text{span}+1$: $k\leftarrow i+\text{span}-1$ For $j=i+1,\dots,k-1$: Non-terminal productions: For $X\to YZ$: Chart$[i,k,X]\oplus=\exp(\text{score}(X\to YZ))\otimes$Chart$[i,j,Y]\otimes$Chart$[j,k,Z]$
3. Return Chart$[1,M+1,S]$

Runtime: $O(M^3|\mathcal{R}|)$ (for tree where nodes expand along right diagonal of tree only, $i=M-\text{span size}+1, j=i+1$, then $O(M|\mathcal{R}|)$)

Pumping lemma

Shows that languages that require strict equality between counts of symbols cannot be generated by a CFG: $a^kb^kc^k=uxyzv$ (u,y,v are fixed, x and z are pumpable) must fulfill $|xyz|\le k$, which is a contradiction

19 Syntax — Dependency

Formulation String w

Spanning tree:
- N nodes + 1 root node (words in sentence $|w|$ or, if there are dependency parsing rules available, items in rules)
- $N-1$ edges, of which 1 edge is fixed (root)
- Directed spanning trees: $N^{(N-1)}$
- Directed, labeled spanning trees: $N^{(N-1)}xx$ where x is given by the number of the destination nodes (e.g. N) raised to the power of the number of edges that can vary (e.g. $N-2$)

Probability of spanning tree —
- $p(t|w)=\frac{\exp(\text{score}(t,w))}{\sum_{t'}\exp(\text{score}(t',w))}$
- Challenge: Naively: $(N-1)^{N-2}$ spanning trees with one root
- Solution: *Edge factored assumption* $p(t|w)=\frac{\prod_{(i\to j)\in t}\exp(\text{score}(i,j,w))\exp(\text{score}(r,w))}{\sum_{t'}\prod_{(i\to j)\in t'}\exp(\text{score}(i,j,w))\exp(\text{score}(r,w))}$ where $(i\to j)$ is an edge, r is the root

Matrix-Tree Theorem — Counts number of directed spanning trees in $O(N^3)$ time:
- *Adjacency matrix*: One entry for each node i to node j, if they were connected via an edge $i\to j$ $A_{ij}=\exp(\text{score}(i,j,w))$ with 0 on diagonal and non-null on off-diagonal
- *Root vector*: One entry for each node j, if it were the root node $\rho_j=\exp(\text{score}(j,w))$
- Construct *Laplacian matrix*, accounting for constraint that there is only one root node:

$$L_{ij}=\begin{cases}\rho_j & \text{if } i=1\\ \sum_{i'=1,i'\neq j}^{n}A_{i'j} & \text{if } i=j\\ -A_{ij} & \text{otherwise}\end{cases}\quad\text{i.e.}\begin{cases}\text{- }1^{st}\text{ row of L contains root scores}\\ \text{- diagonal of L (except } L_{11}\text{) contains sum within each column of } A \text{ (except } A_{i,j}\text{)}\\ \text{- off-diagonal of L (except } 1^{st}\text{ row) contains elements of A × with } -1\end{cases}$$

- According to matrix tree theorem: $|L|=\det(L)=Z=$ number of trees in graph

Optimization *CLE algorithm* in $O(N^2)$ time:

1. Greedy algorithm selects best incoming edge for each node, except the root
2. This can cause cycles, which we contract
3. Break cycle: For each enter edge, break cycle by removing edges that are also incoming at the node where the enter edge is incoming
4. Re-weight: To enter edge, add weights of remaining edges that are strictly on (not in) the cycle
5. If there are multiple edges emanating from the root: For each root edge, calculate cost of deleting this edge: Cost = Weight of root edge - weight of next-best incoming edge to target node (treating cycles as single nodes)
6. Preliminarily remove edge with lowest cost, but keep target node intact
7. Preliminarily repeat steps 5,6 as needed
8. Re-run greedy algorithm in contracted form
9. If this leads to a cycle: Undo removal, contract (treating cycles as single nodes), then re-expand, Otherwise: Re-expand

20 Semantic Parsing

Lambda Calculus

Basic components:
- Logical constants: Objects vs. relations
- Variables: Undetermined logical constants, free vs. bound
  - Objects $x,y,z,\dots$ in $\lambda x.f(x)$
  - Relations $P,Q,R,\dots$ in $\lambda P.P(\dots)$
- Relation has *arity* which is number of objects it relates, e.g. $P(x,y)$ has arity 2
- *Literals*: Formed by applying relations to objects, e.g. likes(ALEX, y), $P(x,y)$

Abstraction:
- Let M,N be terms and x be a variable
- λx is an *abstraction*
- $\lambda x.M$ is a function with input x and *scope of abstraction* M. It replaces every free occurrence of x in M with whatever the function is applied to
- $\lambda x.MN$ is a function applied to N
- Output of $\lambda x.MN$: $M[x:=N]$
- Note:
  - In the output of $\lambda x.MN$, only $M[x:=N]$ remains (for relations $P(x)$, the relation P is exchanged, but the arguments x remain), and $\lambda x.$ and N disappear
  - If M doesn't contain variable, it is returned as is, e.g. $\lambda x.yN$ returns y
  - Work from outside in
  - Determine what is applied to what based on what is the argument in the CCG expression $X|Y$

Application: $M,N\to(MN)$ Parentheses are left-associative, e.g. $(\lambda xy.x\lambda xy.y\lambda xy.y)=(((\lambda xy.x)\lambda xy.y)\lambda xy.y)$

Alpha conversion — Renaming
- We can rename a variable in an abstraction together with all its occurrences in the scope of the abstraction, which are bound to the same abstraction
- Note: Only variables are renamed, not functions $\lambda x.M$
- If the renamed variable remains bound to the same abstraction and the remaining variables remain free resp. bound as before, the renaming is valid

Beta reduction — Application We can apply one lambda term to another if the free variables in N remain free in $M[x:=N]$

LIG resp. CCG

LIG $\langle\mathcal{N},\mathcal{S},\mathcal{I},\Sigma,\mathcal{R}\rangle$ —
- $\mathcal{N}$: Non-terminals $N_1,N_2,N_3,\dots$
- $\mathcal{S}$: Start non-terminal
- $\mathcal{I}$: Indices $f,g,h,\dots$
- Σ: Alphabet of terminals $a_1,a_2,a_3,\dots$
- $\mathcal{R}$: Production rules: $N[\sigma]\to\alpha M[\sigma]\beta$, $N[\sigma]\to\alpha M[f\sigma]\beta$, $N[f\sigma]\to\alpha M[\sigma]\beta$ where N and M are non-terminals, α and β are sequences of terminals and non-terminals, σ is associated with the non-terminal on the LHS and passed to exactly one non-terminal on RHS, base symbol σ is usually combined with a marker f, where markers encode a count and can be popped from (rule 3) or pushed to (rule 2) the stack

CCG $\langle\mathcal{V}_T,\mathcal{V}_N,S,f,\mathcal{R}\rangle$ —
- $\mathcal{V}_N$: Non-terminals
- S: Start non-terminal
- $\mathcal{V}_T$: Terminals
- $\mathcal{R}$: Production rules
- $f: \mathcal{V}_T\to C(\mathcal{V}_N)$: *Lexicon*: Maps $\mathcal{V}_T\cup\{\varepsilon\}$ to $C(\mathcal{V}_N)$
- $C(\mathcal{V}_N)$: *Categories*, including *atomic* and *complex*

Categories:
- *Atomic categories*: Terminals
- *Complex categories*:
  - Built from atomic categories via operators
  - Function with pattern: $X|Y\quad Y\to X$, where $X|Y$ is function, with $|$ being an operator, Y being argument, and X being output, and applying function to Y yields X
  - Have *arity* which is number of arguments it relates, e.g. $X/Y\backslash Z$ has arity 2
- $X=A|_mX_m\cdots|_1X_1$ where
  - A: Atomic category, output of X
  - $X_1,\dots,X_m$: Arbitrary categories, arguments of X
  - m = Arity of X

Operators: Backward and forward slash
- Note: Operators are read from outside in

Rules: Specify how categories can be combined into other categories
- *Function application*:
  - *Forward* ($>$): X: $X/Y\quad Y\to X$
  - *Backward* ($<$): $Y\quad X\backslash Y\to X$
  - CCGs that only have application rules, have power of CFG
- *Function composition*: *Forward* (($B>$: $(X/Y)\quad(Y/Z)\to(X/Z)$, etc.
- *Higher-order rules*: *Forward* ($>^n$): $X/Y\ Y|_nY_n\dots|_1Y_1\to X|_nY_n\dots|_1Y_1$, etc.
- *Type raising*: *Forward* ($T_>$): $X\to T/(T\backslash X)$, etc.
- $2^{|\mathcal{V}_N|}$ forward and $2^{|\mathcal{V}_N|}$ backward rules, but infinitely many rule instances

Lexicon: Associates terminals with categories
- E.g. Atomic: Harry:=NP
- E.g. Complex: walks:=$(S\backslash NP)$

CCG parsing: $\frac{A_1\quad\dots\quad A_k}{B}$ where B is a consequence of $A_1,\dots,A_k$ CKY-style parsing algorithm:
- *Axioms*: $[X,i,i+1]$ where $w_{i+1}=X$ is a lexicon entry
- *Inference rules*: $\frac{[X/Y,i,j]\quad[Y\beta,j,k]}{[X\beta,i,k]}\quad X/Y\quad Y\beta\to X\beta$ resp. $\frac{[Y\beta,i,j]\quad[X\backslash Y,j,k]}{[X\beta,i,k]}\quad Y\beta\quad X\backslash Y\to X\beta$

Polynomial time algorithm:
- Arity bounded by *grammar constant* C_g: ar$(X)\le C_g$
- Challenge: Categories with arity $>C_g$ can no longer be derived
- Solution: Decompose longer derivations into smaller pieces: *Derivation contexts* c
- New inference rules:
  - 1) If the composition of two categories $(X|Y\quad Y\beta)$ would result in a category with arity ar$(X\beta)>C_g$: Context item: $\frac{[X/Y,i,j]\quad[Y\beta,j,k]}{[/Y,\beta,i,j,j,k]}$ where β are all non-terminals and edge operators, except for Y
  - 2.1) If the further composition of a context item with another category $(X|Y\quad Y\beta)$ would result in a category with arity ar$(X\beta)\le C_g$: Context item can be recombined with the original derivation: $\frac{[X/Y,i',j']\quad[/Y,\beta,i,i',j',j]}{[X\beta,i,j]}$
  - 2.2.1) If the further composition of a context item with another category $(\beta|Z\quad Z_\gamma)$ would result in a category with arity ar$(Y\beta\gamma)\le C_g$: Context item can be extended normally: $\frac{[|Y,\beta/Z,i,i',j',j]\quad[Z\gamma,j',k]}{[|Y,\beta\gamma,i,i',j',k]}$ where β_γ refers to β without $/Z$ (can be ε)
  - 2.2.2) Otherwise: A new context is derived from the previous context: $\frac{[|Y,\beta/Z,i,i',j',j]\quad[Z\gamma,j,k]}{[/Z,\gamma,i,i,j,k]}$ item
  - 3.1) When the arity is small enough, the new context item can be recombined with the previous context: $\frac{[|Y,\beta/Z,i'',i',j',j'']\quad[/Z,\varepsilon,i,i'',j'',j]}{[|Y,\beta,i,i',j',j]}$
- $C_g\ge\max\{\ell,a+n\}$: At least as large as maximal Y (determined by largest arity a in lexicon) and β (determined by the maximum degree n of composition rules)

CCG can be paired with Lambda calculus

SK-Calculus

- Alternative to Lambda calculus
- *Variables*: $x,y,z,\dots$
- *Primitive functions resp. combinators*:
  - S: $Sxyz=(xz(yz))=((xz)(yz))$
  - K: Constant function: $Kxy=((Kx)y)=x$
  - I: Identity function: $Ix=x$
  - SKK and I are equivalent: $S(KK)x=SKKx=Kx(Kx)=x$
- Parentheses are left-associative, e.g. $(Kxyz)=(((Kx)y)z)$

Transforming Lambda Calculus into SK-Calculus

1. $T(x)=x$ for every variable x
2. $T[(E_1E_2)]=(T[E_1]T[E_2])$ for all lambda terms E_1,E_2
3. $T[\lambda x.E]=(KT[E])$ for every lambda term E where x is either bound or absent within the term
4. $T[\lambda x.x]=(SKK)=I$
5. $T[\lambda x.\lambda y.E]=T[\lambda x.T[\lambda y.E]]$ for every lambda term E where x is free within the term
6. $T[\lambda x.(E_1E_2)]=(ST[\lambda x.E_1]T[\lambda x.E_2])$ for all lambda terms E_1,E_2 where x is free within at least one of the two terms

21 WFSA

FSA $\mathcal{A}=(\Sigma,Q,I,F,\delta)$

- Alphabet Σ with $a,b,c,\dots$
- States Q, initial states I, final states F
- Transitions $\delta\subseteq Q\times(\Sigma\cup\{\varepsilon\})\times Q$ resp. $q\xrightarrow{a\in\Sigma\cup\{\varepsilon\}}q'$
- Sequentially reads individual symbols of an input string s and transitions from state q to state q' upon reading a symbol a iff $(q,a,q')\in\delta$
- If, after reading the last symbol, ends up in a state $q_f\in F$, automaton *accepts* the string

WFSA $\mathcal{A}=(\Sigma,Q,I,F,\delta,\lambda,\rho)$

- Transitions weighted with SR
- $I=\{q\in Q|\lambda(q)\neq 0\}\subseteq Q$, $F=\{q\in Q|\rho(q)\neq 0\}\subseteq Q$
- $\delta\subseteq Q\times(\Sigma\cup\{\varepsilon\})\times\mathbb{K}\times Q$ resp. $q\xrightarrow{a\in\Sigma\cup\{\varepsilon\}/w}q'$
- $\lambda: Q\to\mathbb{K}$ resp. $\rho: Q\to\mathbb{K}$ are initial resp. final weighting functions, $\bar{0}$ for non-initial resp. non-final q

(W)FSA Terminology

Paths: π
- Element of δ^*: $q_1\xrightarrow{a_1/w_1}q_2,q_2\dots q_N$
- p$(\pi)=q_1$ is the beginning and q$(\pi)=q_N$ the ending state of the path
- *Length* of is the number of transitions, *yield* the concatenation of symbols
- *Inner path weight*: $w_I(\pi)=\bigotimes_{n=1}^N w_n$
- *Path weight*: $w(\pi)=\lambda(\text{p}(\pi))\otimes w_I(\pi)\otimes\rho(\text{q}(\pi))$
- A path is *accepting* iff $w(\pi)\neq 0$

Transitions:
- Outgoing from q: $E_{\mathcal{A}}(q)=\{a,t,w|(q,a,w,t)\in\delta\}$
- Incoming to q: $E_{\mathcal{A}}^{-1}(q)=\{a,s,w|(s,w,a,q)\in\delta\}$

States:
- *Accessible* iff $q\in I$ or path from I to q with $w(\pi)>0$
- *Co-accessible* iff $q\in F$ or path from q to F with $w(\pi)>0$

(W)FSA is *unambiguous* iff for string there is maximum one accepting path

WFST $\mathcal{T}=(\Sigma,\Omega,Q,I,F,\delta,\lambda,\rho)$

- Σ is the input alphabet, Ω is the output alphabet
- $\delta\subseteq Q\times(\Sigma\cup\{\varepsilon\})\times(\Omega\cup\{\varepsilon\})\times\mathbb{K}\times Q$ resp. $q\xrightarrow{a\in\Sigma\cup\{\varepsilon\}:b\in\Omega\cup\{\varepsilon\}/w}q'$

WFST Compositions

- $\mathcal{T}_1\circ\mathcal{T}_2$ from $\mathcal{T}_1=(\Sigma,\Omega,Q_1,I_1,F_1,\delta_1,\lambda_1,\rho_1)$ and $\mathcal{T}_2=(\Omega,\Gamma,Q_2,I_2,F_2,\delta_2,\lambda_2,\rho_2)$: $\mathcal{T}=(\Sigma,\Gamma,Q,I,F,\delta,\lambda,\rho)$ such that: $\mathcal{T}(x,y)=\bigoplus_{z\in\Omega^*}\mathcal{T}_1(x,z)\otimes\mathcal{T}_2(z,y)$ where $\mathcal{T}(i,j)$ is weight assigned to mapping from input i to an output j

Naive algorithm:
1. Initial state $(q_1^i,q_2^i)\in I_1\times I_2$ and final state $(q_1^f,q_2^f)\in F_1\times F_2$
2. For $q_1,q_2\in Q_1\times Q_2$: For $(q_1\xrightarrow{a:b/w_1}q_1'),(q_2\xrightarrow{c:d/w_2}q_2')\in E_{\mathcal{T}_1}(q_1)\times E_{\mathcal{T}_2}(q_2)$: If $b=c$, (q_1,q_2) and (q_1',q_2') are accessible and co-accessible (reachable from (q_1^i,q_2^i), can reach (q_1^f,q_2^f)) in $\mathcal{T}$: Add new (q_1,q_2), (q_1',q_2') and $(q_1,q_2)\xrightarrow{a:d/w_1\otimes w_2}(q_1',q_2')$ to $\mathcal{T}$
3. For (q_1^i,q_2^i): $\lambda_{\mathcal{T}}=\lambda_1(q_1^i)\otimes\lambda_2(q_2^i)$
4. For (q_1^f,q_2^f): $\rho_{\mathcal{T}}=\rho_1(q_1^f)\otimes\rho_2(q_2^f)$
5. Return $\mathcal{T}$

- Challenge: Runtime $O(|Q_1||Q_2|)$
- Solution: *Accessible algorithm*: Construct all possible pairs of initial states and then expand outwards, adding accessible states

Pathsum and Algorithms

Pathsum: $Z(\mathcal{A})=\bigoplus_{\pi\in\Pi(\mathcal{A})}w(\pi)$ Backward algorithm:
Computes finite $Z(\mathcal{A})$ in acyclic WFSA in $O(|\delta|)$ time:
1. For $q\in$Rev-Top$(\mathcal{A})$ (starts with final state):
2. If $q\in F$: $\beta(q)=\rho(q)$ Else: $\beta(q)=\bigoplus_{q\xrightarrow{a/w}q'}w\otimes\beta(q')$
3. Return: $\bigoplus_{q_i\in I}\lambda(q^i)\otimes\beta(q^i)$

Forward: Exchange I and F and use $q'\xrightarrow{a/w}q$

Lehmann's: Computes infinite $Z(\mathcal{A})$ in cyclic WFSA:
First WFSA as a matrix:
- *Adjacency matrix* for all $\sigma\in\Sigma\cup\{\varepsilon\}$: $\boldsymbol{W}^{(a)}\in\mathbb{R}^{(|Q|\times|Q|)}$: From-states q_n in rows, entries as weights w for $q_n\xrightarrow{a/w}q_m$ if it exists, else $\bar{0}$
- Can be collapsed: $\boldsymbol{W}=\bigoplus_{a\in\Sigma\cup\{\varepsilon\}}\boldsymbol{W}^{(a)}$
- Pathsum for paths of length exactly l: $\boldsymbol{W}^l=\boldsymbol{W}\otimes\boldsymbol{W}\otimes\dots$

Compute $Z(\mathcal{A})$ via Kleene closure of matrix $\boldsymbol{R}$ over closed SR in $O(|Q|^3)$ time:
1. $\boldsymbol{R}\in\mathbb{R}^{(|Q|\times|Q|)}$ with entries as $\oplus$-sum of the weights for $q_i\to q_k$: $\boldsymbol{R}_{ik}=\bigoplus_{\pi\in\Pi(q_i,q_k)}w_I(\pi)$ where cyclical terms (either single nodes or node combinations) are denoted with $*$
2. $Z(\mathcal{A})=\bigoplus_{i,k=1}^{|Q|}\lambda(q_i)\otimes\boldsymbol{R}_{ik}\otimes\rho(q_k)$ where SR retains only pairs (i,k) where i is initial and k is final state

Compute $\boldsymbol{R}$:
1. $\boldsymbol{R}^{(0)}\leftarrow\boldsymbol{W}$
2. For $j\leftarrow 1$ up to $|Q|$: For $i\leftarrow 1$ up to $|Q|$: For $k\leftarrow 1$ up to $|Q|$: $\boldsymbol{R}_{ik}^{\le j}\leftarrow\boldsymbol{R}_{ik}^{\le j-1}\oplus(\boldsymbol{R}_{ij}^{\le j-1}\otimes(\boldsymbol{R}_{jj}^{\le j-1})^*\otimes\boldsymbol{R}_{jk}^{\le j-1})$
3. Return: $\boldsymbol{I}\oplus\boldsymbol{R}^{(|Q|)}$

Transliteration

Aim and approach —
Aim: Compute $p(\boldsymbol{y}|\boldsymbol{x})=\frac{\exp(\text{score}(\boldsymbol{x},\boldsymbol{y}))}{Z}$ where score$(\boldsymbol{x},\boldsymbol{y})$ is the pathsum $\log\sum_{\pi\in\Pi(\boldsymbol{y})}w(\pi)$

Approach:
- $\boldsymbol{x}\in\Sigma^*$ as $\mathcal{T}_x$
- $\boldsymbol{y}\in\Omega^*$ as $\mathcal{T}_y$
- $\mathcal{T}$ maps Σ to Ω
- $\mathcal{T}_x\circ\mathcal{T}$: Each path is one alignment of $\boldsymbol{x}$ to Ω^* (pink)
- $\mathcal{T}_x\circ\mathcal{T}\circ\mathcal{T}_y$: Each path is one alignment of $\boldsymbol{x}$ to $\boldsymbol{y}$
- $p(\boldsymbol{y}|\boldsymbol{x})$ is probability of all paths in $\mathcal{T}_x\circ\mathcal{T}$ that align $\boldsymbol{x}$ to $\boldsymbol{y}$: $=\frac{\text{pathsum of }\mathcal{T}_x\circ\mathcal{T}\circ\mathcal{T}_y}{\text{pathsum of }\mathcal{T}_x\circ\mathcal{T}}$
- Training: For $p(\boldsymbol{y}|\boldsymbol{x})$ use Lehmann's with the log-sum-exp SR
- Inference: For highest scoring $\boldsymbol{y}$ of $\mathcal{T}_x\circ\mathcal{T}$ use Dijkstra's or Floyd-Warshall with the arctic SR
- Conditioning on $\boldsymbol{x}$ in $\mathcal{T}_x\circ\mathcal{T}$ makes WFST a WFSA

Training — $\mathcal{A}_x$ is the WFSA with:
- Vector $\boldsymbol{\lambda}$ with $\lambda_n=\lambda(q_n)$
- Vector $\boldsymbol{\rho}$ with $\rho_n=\rho(q_n)$
- Adjacency matrix $\boldsymbol{W}^{(\omega)}$ for all $\omega\in\Omega\cup\{\varepsilon\}$ with entries as weights for $q_n\xrightarrow{\omega}q_m$
- Can be collapsed: $\boldsymbol{W}=\sum_{\omega\in\Omega\cup\{\varepsilon\}}\boldsymbol{W}^{(\omega)}$

Computation of Z for instance:
- $Z(\boldsymbol{x})=\sum_{\boldsymbol{y}'\in\Omega^*}\exp(\text{score}(\boldsymbol{x},\boldsymbol{y}'))$
$=\sum_{\boldsymbol{y}'\in\Omega^*}\exp\left(\log\sum_{\pi\in\Pi(\boldsymbol{y}')}w(\pi)\right)$
$=\sum_{\boldsymbol{y}'\in\Sigma^*}\sum_{\pi\in\Pi(\boldsymbol{y}')}\lambda(\text{p}(\pi))\otimes w_I(\pi)\otimes\rho(\text{q}(\pi))$
$=\sum_{\boldsymbol{y}'\in\Sigma^*}\sum_{\pi\in\Pi(\boldsymbol{y}')}\lambda(\text{p}(\pi))\otimes\prod_{n=1}^{|\pi|}w_n(\pi)\otimes\rho(\text{q}(\pi))$ which is pathsum
$=\sum_{q_n,q_m\in Q}\left(\sum_{\pi\in\Pi(q_n,q_m)}w(\pi)\right)$
$=\sum_{q_n,q_m\in Q}\lambda(q_n)\otimes(\boldsymbol{W}^*)_{nm}\otimes\rho(q_m)$
$=\boldsymbol{\lambda}^T\boldsymbol{W}^*\boldsymbol{\rho}$ with $\boldsymbol{W}^*$ via Lehmann's

Lehmann's applications

Floyd-Warshall algorithm:
- Aim: Find shortest distances between all pairs of nodes in a graph
- Corresponds to Lehmann over tropical SR
- Requirement: No negative cycles in the graph, since if there were, we could always loop over them to lower the pathsum
- Then we can drop $(\boldsymbol{R}_{jj}^{\le j-1})^*$ in Lehmann

Gauss-Jordan algorithm:
- Aim: For $\boldsymbol{M}^{D\times D}$, find $\boldsymbol{M}^{-1}$
- Given $(\boldsymbol{I}-\boldsymbol{M})^*=\boldsymbol{M}^{-1}$, run Lehmann on $(\boldsymbol{I}-\boldsymbol{M})$

(W)FSA applications

N-gram models — As WFSA:
- $\Sigma=V\cup\{\langle\text{BOS}\rangle,\langle\text{EOS}\rangle\}$
- $Q=\bigcup_{n=0}^{N}\{\{\langle\text{BOS}\rangle\}^{N-n}\times V^{n-1}\times(V\cup\{\langle\text{EOS}\rangle\})\}$ represent $N-1$ grams
- $I=\{\langle\text{BOS},\dots,\text{BOS}\rangle\}$ (N times) is a subset of Q at $n=0$
- $F=Q$
- $\delta=\{y_{-N}\cdots y_{-1},y_0,p(y_0|y_{-N+1},\dots,y_{-1}),y_{-N+1}\cdots y_0$ for $(y_{-N+1}\cdots y_{-1})\in\bigcup_{n=0}^{N-1}\{\langle\text{BOS}\rangle\}^{N-1-n}\times V^n\}$ for for $N\ge 2$
- $\delta(q_{ij},y_n,1,q_{ki})=\begin{cases}\log(p(y_n=k|y_{n-1}=i,y_{n-2}=j,\dots,y_{n-N+1}) & \text{if } y_n=k\\ -\infty & \text{else}\end{cases}$
- $\lambda=\langle\text{BOS}\rangle\dots\langle\text{BOS}\rangle\to\bar{1}$ (N times, full padding)
- $\lambda(q_{ij})=\log(p(y_n=i|y_{n-1}=\text{BOS},\dots))$
- $\rho=y_{n-N+1}\cdots y_1\ \langle\text{EOS}\rangle\to\bar{1}$
- $\rho(q_{ij})=\log(p(y_n=\text{EOS}|y_{n-1}=i,\dots))$
- $p(\boldsymbol{y})=\lambda(q_I)+\sum_{n=1}^{L}\delta(q_{y_{n-N},\dots,y_{n-1}},y_n,q_{y_{n-N+1},\dots})+\rho(q_F)$
- $|V|^{N-2}+|V|^{N-1}+1$ start states + intermediate states + end state

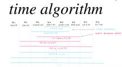
CRFs —
$p(\boldsymbol{y}|\boldsymbol{x})=\frac{\prod_{n=2}^{M}\exp(\boldsymbol{w}\cdot\boldsymbol{f}(y_n,y_{n-1},\boldsymbol{x},n))}{\sum_{\boldsymbol{y}'}\prod_{n=2}^{M}\exp(\boldsymbol{w}\cdot\boldsymbol{f}(y_n',y_{n-1}',\boldsymbol{x},n))}$
as WFSA:
- $\Sigma=\{\varepsilon\}$
- $Q=I=F=V^M$
- $\delta=\{y_n,\varepsilon,\exp(\boldsymbol{w}\cdot\boldsymbol{f}(y_n,y_m,\boldsymbol{x})),y_m\}$
- $\lambda=y_n\to\exp(\boldsymbol{w}\cdot\boldsymbol{f}(y_n,\langle\text{BOS}\rangle,\boldsymbol{x}))$, analog for ρ

String edit FSA — Edit distance $\le d$ for string of length N has $(d+1)(N+1)$ states

Images

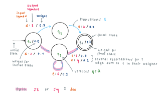
Syntax — Syntactic (gray) → dependency parsing

Semantics — Polynomial time algorithm

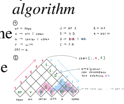


Syntactic — Parse tree

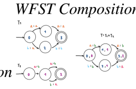
WFSA



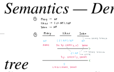
Syntactic — CKY algorithm



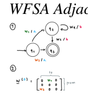
WFST Composition



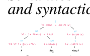
Semantics — Derivation tree



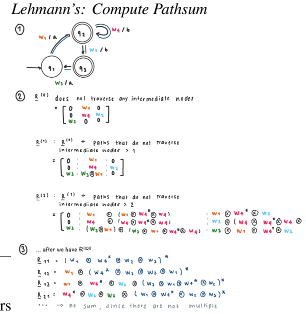
WFSA Adjacency Matrix



Semantics — Semantic and syntactic parse



Lehmann's: Compute Pathsum



POS — Backward algorithm