Training set - 80%.

Development set - 10%. (Frequently used test data).

Test set - 10%.

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$$= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

Entropy $H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$

$$P(w_1 w_2 \ldots w_N) = P(w_1) P(w_2 | w_1) \ldots P(w_N | w_{N-1})$$

PP of Unigram > PP of Bigram > PP of Trigram

Lower PP, more the information

OOV rate

↳ Out of vocabulary

PP's should be compared across language models with the same vocabularies.

Smoothing / Discounting: Done to avoid getting 0 probs for unseen events.

$$P(w_i) = \frac{c_i + 1}{N + V}$$

Laplace / Add-1 smoothing

$$c_i^* = \frac{(c_i + 1) N}{N + V}$$

$$d_c = \frac{c^*}{c}$$

$$P'_{Laplace}(w_n | w_{n-1}) = \frac{C(w_n, w_{n-1}) + 1}{\sum_w (C(w_n, w) + 1)}$$

Add-k smoothing:

$$P^*_{Add-k}(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n) + k}{C(w_{n-1}) + kV}$$

Held-out estimation:

$$P(w_1 \ldots w_n) = \frac{T_r}{N_r N}$$

Cross-estimation :

$$P_{ho}(w_1 \ldots w_n) = \frac{T_r^{01} + T_t^{10}}{(N_0^0 + N_0^1) N}$$

Back-off vs Interpolation

Katz back-off:

$$P_{BO}(w_n | w_{n-1}^n, w_N) = \begin{cases} P^*(w_n | w_{n-1} \cdot w_N) & \text{if } C(w_n, w_{n-1} \ldots) > 0 \\ \alpha(w_{n-1} w_N) \cdot P_{BO}(w_n | w_{n-1} \ldots w_{N-1}) & \text{otherwise} \end{cases}$$

Good Turing :

$$r^* = \frac{(r+1) E(N_{r+1})}{E(N_r)}$$

$$P(w_1, \ldots w_n) = \frac{r^*}{N}$$

Kneser-Ney smoothing: (Absolute discounting)

$$P_{\text{Absolute discounting}}(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i) - d}{\sum_v C(w_{i-1} v)} + \lambda(w_{i-1}) P(w_i)$$

$$P_{\text{CONTINUATION}}(w) \propto |\{v : C(vw) > 0\}|$$

$$= \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u'w') > 0\}|}$$

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(C(w_{i-1} w_i) - d, 0)}{C(w_{i-1})} + \alpha(w_{i-1}) P_{cont}(w_i)$$

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1} v)} |\{w : C(w_{i-1} w) > 0\}|$$

Witten-Bell smoothing : N → no. of tokens

T → no. of types (not V)

$$Z = \sum_{i : c_i = 0} 1$$

Good turing count for most frequent word (like "the") is undefined

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

$$P_c = \frac{c^*}{N}$$

jii ⇒ Joint if. independent

jii = Ne (p(x)) e/p(y)) for xy

N bins

Typically 3 bins per decade.

Simple Good-Turing: (Monte Carlo examples)

$$\log(N_r) = a + b\log r$$

a, b are fit by linear regression

Closed class ⇒ $1^* > 1$

① $r^* < r$

② $\frac{r^*}{r} \uparrow 1$ as $r \uparrow$

$$r^* = \frac{(r+1)N_{r+1}}{N_r} = \frac{(r+1)A(r+1)^b}{Ar^b}$$

$$= r\left(1+\frac{1}{r}\right)^{b+1}$$

$r^* < r$ if $b+1 < 0 \Rightarrow b < -1$

Low frequencies ⇒ Good Turing estimates

Higher " ⇒ SGT

Prob. of unseen unigrams $= \frac{T}{Z(T+N)}$

" " seen " $= \frac{c_i}{T+N}$

Prob of unseen bigrams $P^*(w_i|w_x) = \frac{T(w_x)}{Z(w_x)(N(w_x)+T(w_x))}$

" " seen " $= \frac{C(w_x, w_i)}{N(w_x)+T(w_x)}$

$Z(w) = V - T(w)$

↳ seen bigram types starting with w.

$N(w)$ = No. of bigram tokens starting with w

POS Tagging

Assign POS to each word in a sentence

Discourse & Coreference
Semantics extraction
Parsing
Chunking
POS tagging
Morphology

↑ Increased complexity

Named Entity Recognition:

Company : Boeing Co.

Location : Wall Street

Person : Alan Mulally

NA : No entity

Local vs contextual constraints

↓

More likely in general

$x(i)$ → Input $y(i)$ → Output

Eg: $x(i)$ → The dog laughs

$y(i)$ → DT NN VB

Conditional models : $P(y|x)$ — learn

Generative models : $P(x, y) = P(y) \cdot P(x|y)$ — learn

↓ Prior

$$P(y|x) = \frac{P(y)\ P(x|y)}{P(x)}$$

↓

Discriminative model

$$f(x) = \underset{y}{\operatorname{argmax}}\ P(y|x)$$

$$= \underset{y}{\operatorname{argmax}}\ \frac{P(y) \cdot P(x|y)}{P(x)}$$

$$= \underset{y}{\operatorname{argmax}}\ P(y) \cdot P(x|y)$$

$P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_n)$

$x_1 \ldots x_n$ ↓ Sentence $y_1 \ldots y_n$ ↓ Tag Sequence

$$\underset{y_1 \ldots y_n}{\operatorname{argmax}}\ P(x_1, x_2, \ldots x_n, y_1, \ldots, y_n)$$

Trigram HMM's:

$V \rightarrow$ sentence $(x_1, x_2, \ldots x_n)$

$S \rightarrow$ Tag sequence $(y_1, \ldots y_n)$

$y_{n+1} =$ STOP

$$P(x_1, \ldots x_n, y_1, \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

$q(y_i | y_{i-2}, y_{i-1})$ ↑ Trigram $e(x_i | y_i)$ ↑ Emission parameters

$$\text{or } \prod_{i=1} e(x_i | y_i) = P(x_1 \ldots x_n | y_1 \ldots y_{n+1})$$

$$q(Vt | DT, JJ) = \lambda_1 \frac{\#(DT, JJ, Vt)}{\#(DT, JJ)} + \lambda_2 \frac{\#(JJ, Vt)}{\#(JJ)} + \lambda_3 \times \frac{\#(Vt)}{\#()}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

$$e(base | Vt) = \frac{\#(base, Vt)}{\#(Vt)}$$

$e(x, y) = 0$ $\forall y$ if x is never seen in data

Frequent words : $\#(words) \geq 5$

Low frequency words : Put into corresponding classes

Eg : Profits : First word

, : Punctuation

$$e(\text{Profits} | NN) = e(\text{First word} | NN)$$

No. of possible sequences = $|S|^n$.

Viterbi Algorithm:

$$r(y_{-1}, y_0, y_1, \ldots y_k) = \prod_{i=1}^{k} q(y_i | y_{i-2}, y_{i-1}) \cdot \prod_{i=1}^{k} e(x_i | y_i)$$

$$e(x_i, x_{i-1} | y_i)$$

$\pi(k, u, v)$ = Max. probability of a tag sequence ending in tags u, v at position k.

Define S_k for $k = -1 \ldots n$ to be set of possible tags at position k.

$$S_{-1} = S_0 = \{*\}$$

$$S_k = S \quad \forall k \in \{1 \ldots n\}$$

-1	0	x_1	x_2	...	x_n
		↓			
		D			
		N			
		V			
		P			

$$\pi(k, u, v) = \max_{(y_{-1}, y_0, \ldots y_k) : y_{k-1} = u, y_k = v} r(y_{-1}, y_0, \ldots y_k)$$

$S = \{D, N, V, P\}$

$\pi(7, P, D)$

$w \; u \; v$

The man saw the dog with the telescope

$$\pi(0, *, *) = 1$$

For any $k \in \{1 \ldots n\}$, for any $u \in S_{k-1}$ and $v \in S_k$.

$$\pi(k, u, v) = \max_{w \in S_{k-2}} (\pi(k-1, w, u) \times q(v | w, u) \times e(x_k | v))$$

Algo:

Set $\pi(0, *, *) = 1$

$S_{-1} = S_0 = \{*\}$; $S_k = S$ for $k = \{1 \ldots n\}$

for $k = 1$ to n

for $u \in S_{k-1}$, $v \in S_k$

$$\pi(k, u, v) = \max_{w \in S_{k-2}} (\pi(k-1, w, u) \; q(v | w, u) \; e(x_k | v))$$

$$bp(k, u, v) = \arg$$

return $\max_{u \in S_{n-1}, v \in S_n} (\pi(n, u, v) \times q(STOP | u, v))$

$$(y_{n-1}, y_n) = \underset{(u,v)}{\operatorname{argmax}} \left(\pi(n, u, v) \times q(STOP \mid u, v) \right)$$

For $k = n-2 \ldots 1$ $\quad y_k = bp(k+2, y_{k+1}, y_{t+2})$

$$O = \{o_1, \ldots o_n\}$$

$P(O|\lambda)$

$$P(O|\lambda) = \sum_n P(O|Q, \lambda) P(Q|\lambda)$$

$$\alpha_t(i) = P(o_1, \ldots o_t, q_t = s_i | \lambda)$$

$$\alpha_1(i) = \pi_i b_i(o_1)$$

Words with frequency < 100 serve as unknown words.

F → ~~adverb~~ function words

R → Adverb

Chaotic systems

POS tagging ⇒ Sequence labeling task

$W = \{w_1, w_2, \ldots, w_n\}$ ⇒ Observation sequence

$T = \{t_1, t_2, \ldots, t_n\}$

$$P(T|W) = \frac{P(T)\,P(W|T)}{P(W)}$$

Markov assumption → Limited Horizon

→ Time invariance:

$$P(X_t = i \mid X_{t-1} = j) = P(X_0 = i \mid X_0 = j)$$

Hidden Markov model (HMM):

Set of states = S

Alpha Output = V

Transition probs: $A = \{a_{ij}\}$.

Emission probs: $B = \{b_j(o_k)\}$

$\lambda = (A, B, \pi)$ → Initial state probs.

① $P(O|\lambda)$ → Likelihood of a sequence (F - B)

② Given O, get best $Q = \{q_1 \ldots q_n\}$ (state sequence) (Viterbi)

③ Adjust λ to best maximize $P(O|\lambda)$ Given O.

(Baum - Welch) (Re-estimate)

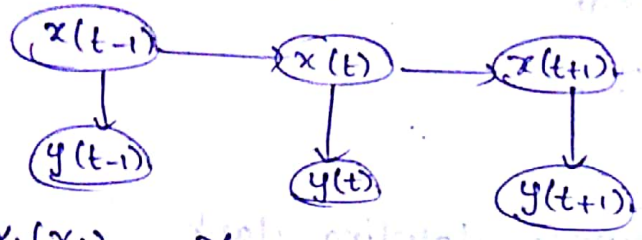
$$P(O|\lambda) = \sum_{Q} P(O|Q;\lambda)\,P(Q|\lambda)$$

↓ in terms of emission → in terms of transition

$2TN^T$

Forward procedure :

Belief state : Prob. of state at a certain time given history of evidence $\alpha_1(i) = \pi_i b_i(o_1)$



$\alpha_t(x_t) = P(x_t, y_{1:t})$ $\quad P(o_1, \dots o_t, q_t = S_i \mid \lambda)$

$= \sum_{x_{t-1}} P(x_t, x_{t-1}, y_{1:t})$

$= \sum_{x_{t-1}} P(y_t \mid x_t, x_{t-1}, y_{1:t-1}) P(x_t \mid x_{t-1}, y_{1:t-1}) P(x_{t-1}$

$= P(y_t \mid x_t) \sum_{x_{t-1}} P(x_t \mid x_{t-1}) \cdot \alpha_{t-1}(x_{t-1})$

Emission $\quad$ Transition

$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$ $\quad \alpha_{t+1}(j) = \sum \alpha_t(i) \times a_{ij} \times b_j(o_{t+1})$

Backward procedure : $\beta_t(i) = P(o_{t+1} \dots o_T \mid q_t = S_i, \lambda)$

$\beta_t(x_t) = P(y_{1:t} \mid x_t) = \dfrac{P(x_t, y_{1:t})}{P(x_t)}$

$\beta_t(x_t) = \sum_{x_{t+1}} \beta_{t+1}(x_{t+1}) \cdot P(x_{t+1} \mid x_t) \cdot P(y_{t+1} \mid x_{t+1})$

$\beta_T(i) = 1$

$\sum_{j=1}^{N} a_{ij} \cdot \beta_{t+1}(j) b_j o(t+1) = \beta_t(i)$ $\quad (N^2 T)$

Viterbi algorithm :

Baum - Welch algo :

$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda)$

$= \dfrac{\alpha_t(i) \cdot a_{ij} b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{P(O \mid \lambda)}$

$P(O \mid \lambda) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$

$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$

$\bar{a}_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_t \gamma_t(i)}$ $\quad \bar{b}_j(k) = \dfrac{\sum_{t \text{ s.t. } o_t = v_k} \gamma_t(j)}{\sum_t \gamma_t(j)}$

$\bar{\pi}_i = \gamma_1(i)$

Logistic regression → estimate parameters of $P(Y|X)$

Naive Bayes → " " " $P(X|Y)$ & $P(Y)$ → Generative

Discriminative

Normal neural networks don't scale well to images.

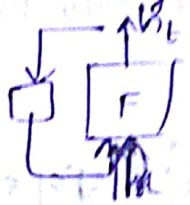
So, use CNN's

3D input 3D output

Recurrent NN → used for sequence of text (Sequential data)

↓

In RNN, output of a hidden layer is fed back into the same layer.

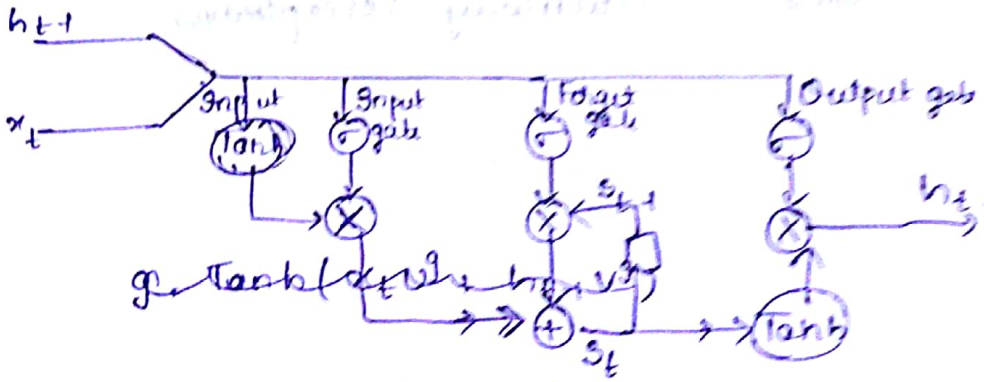


$$h_t = \sigma(Ux_t + Vh_{t-1})$$

$$= \sigma(Ux_t + V(\sigma(Ux_{t-1} + V(\sigma(Ux_{t-2})))))$$

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial out_3} \cdot \frac{\partial out_3}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial U}$$

Vanishing Gradient problem. So, use LSTM.



$$S_t = S_{t-1} \circ f + g \circ i$$

$g = \tanh(x_t U^g + h_{t-1} V^g + b^g)$ Squashed input } Element wise product

$i = \sigma(b^i + x_t U^i + h_{t-1} V^i)$ Input gate

$f = \sigma(b^f + x_t U^f + h_{t-1} V^f)$ $S_{t-1} \circ f$ → Acts as weight for input

→ Acts as weight for S_{t-1}

$$O = \sigma(b^o + x_t U^o + h_{t-1} V^o)$$

$$h_t = \tanh(S_t) \circ O$$

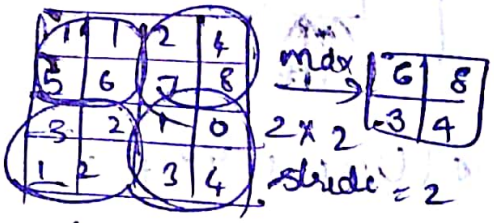
This way, LSTM controls what is "input, remember, output" using gates.

$$\frac{\partial S_t}{\partial S_{t-1}} = f.$$

CNN's : Used for images

Convolution layer : Input matrix ⊛ convolved with filter matrix. Use Relu on the obtained matrix.

Pooling layer : Downsampling.
Max pooling
Avg "
Sum "



Fully connected layer : Flatten matrix into vector & feed into neural network

Feed forward NN's ⇒ Multiway Perceptrons

NP → ProperNoun

NP → Det Nominal

Nominal → Noun | Nominal Noun

S → NP VP

NP → Pronoun | Proper noun | Det Nominal

Nominal → Nominal Noun | Noun

VP → Verb | Verb NP | Verb NP PP | Verb PP

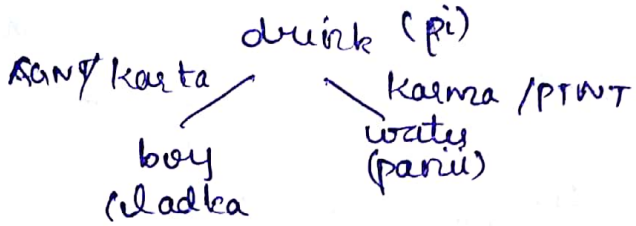
PP → Preposition NP.

S → Aux NP VP

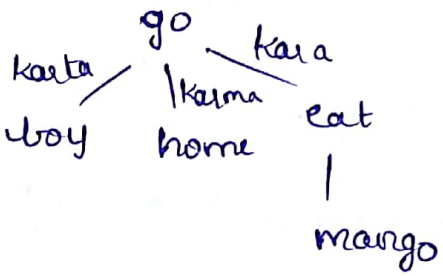
S → wh-NP VP

S → wh-NP Aux NP VP

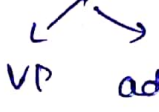
Verb is head & noun is modifier



Temporal precedence (Verb - Verb modification)



NP → det (adjP)*n



VP → V [NP] [NP] PP* [AdvP]

S → AdvP NP VP

AdvP → adv | VP

Viterbi algorithm:

① $\delta_1(i) = \pi_i b_i(O_1)$

$\psi_1(i) = 0$

② $\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t)$

$\psi_t(j) = \operatorname{argmax}_{1 \le i \le N} [\delta_{t+1}(i) a_{ij}]$

③ $P^* = \max_{1 \le i \le N} [\delta_T(i)]$

$q_T^* = \operatorname{argmax}_{1 \le i \le N} [\delta_T(i)]$

④ $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Chunks are non-recursive & non-exhaustive

Constituents are recursive

Algorithms that try to learn $P(y|x)$ directly (like logistic regression) are discriminative

GDA: perceptron, NN etc.

$P(x|y)$ & $P(y)$ ⇒ Generative

$$P(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{d/2}} e^{\frac{-1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

$Cov(x) = \Sigma$

$\Sigma \uparrow$ compressed.

$P(y) = \phi^y (1-\phi)^{1-y}$ ⇒ Bernoulli $P(x|y)$ ⇒ Gaussian

$$l(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma)$$

$$= \log \prod_{i=1}^{m} P(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) P(y^{(i)}; \phi)$$

maximize l

$$\phi = \frac{1}{m} \sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$$

$$P(y=1|x; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1+e^{-\theta^T x}}$$

↕ function of $\phi, \Sigma, \mu_0, \mu_1$

Modeling assumptions Correct → GDA is correct

else LR better

NB:

$$P(x_1, x_2, \ldots x_n | y) = P(x_1|y)\, P(x_2|y, x_1)\, P(x_3|y, x_1, x_2) \ldots$$

$$= P(x_1|y)\, P(x_2|y)\, P(x_3|y) \ldots P(x_n|y)$$

$$\phi_{i|y=1} = P(x_{i=1} | y = 1)$$

$$\phi_{i|y=0} = P(x_i = 1 | y = 0)$$

$$\phi_y = P(y = 1)$$

$$L(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}$$ → Laplace smoothing

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

2^n possible values for x

2 " " y

2^{n+1} parameters

$2^{n+1} - 1$ independent parameters

$2(2^n - 1) \longrightarrow 2n$

Naive Bayes

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i + |V|}$$

$$l(w) = \sum_l Y^l \ln P(Y^l = 1 | X^l, w) + (1 - Y^l) \ln P(Y^l = 0 | X^l, w)$$

① $$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)}$$

$$\sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\left(\frac{(X_i - \mu_{i0})^2}{2\sigma_i^2}\right)}}{\frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\left(\frac{(X_i - \mu_{i1})^2}{2\sigma_i^2}\right)}}$$

$$\sum_i \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

②
$$w \leftarrow \arg\max_w \prod_l P(Y^l | X^l, w)$$

$$w \leftarrow \arg\max_w \sum_l \ln P(Y^l | X^l, w)$$

$$l(w) = \sum_l Y^l \ln \frac{P(Y^l=1|X^l,w)}{P(Y^l=0|X^l,w)} + \ln P(Y^l=0|X^l,w)$$

$$= \sum_l Y^l (w_0 + \sum_i w_i X_i^l) - \ln\left(1 + \exp(w_0 + \sum_i w_i X_i^l)\right)$$

$$\frac{\partial l(w)}{\partial w_i} = \sum_l X_i^l \left(Y^l - \hat{P}(Y^l = 1 | X^l, w)\right)$$

③
$$w \leftarrow \arg\max_w \sum_l \ln P(Y^l | X^l, w) - \frac{\lambda}{2} \|w\|^2$$

$$\sim \sum_l \ln P(Y^l | X^l, w) + \ln P(w)$$

↓ 0 mean GD.

⑥. $w_{ji} \leftarrow w_{ji} + \eta \sum_d x_i^d (\delta(y^d = y_j) - \hat{P}(y^d = y_j | x^d, w)) - \eta \lambda w_{ji}$

a_i s_i w_{ji}^h s_j^h w_{kj} s_k^o

$$E(m) = \frac{1}{2} \sum_{k=1}^{K} [b_k(m) - f_k^o(x_k^o)]^2$$

$$x_k^o = \sum_{j=1}^{J} w_{kj} \cdot s_j^h$$

$$s_j^h = f_j^h(x_j^h)$$

$$x_j^h = \sum_{i=1}^{I} w_{ji}^h s_i$$

$$s_i = x_i = a_i(m)$$

$$\frac{\partial E(m)}{\partial w_{kj}} = \frac{1}{2} \frac{\partial}{\partial w_{kj}} \left[b_k - f_k^o \left(\sum_j w_{kj} s_j^h \right) \right]^2$$

$$= -(b_k - f_k^o) \times f_k^{o'} \cdot s_j^h$$

$$= -\delta_k^o s_j^h$$

$$\frac{\partial E(m)}{\partial w_{ji}} = -\sum_{k=1}^{K} (b_k - f_k^o) \cdot \frac{\partial}{\partial w_{ji}} f_k^o \left(\sum_j w_{kj} s_j^h \right)$$

$$= -\sum_{k=1}^{K} (b_k - f_k^o) \cdot f_k^{o'} \cdot w_{kj} \cdot \frac{\partial s_j^h}{\partial w_{ji}}$$

$$\frac{\partial s_j^h}{\partial w_{ji}} = f_j^{h'} \cdot s_i$$

Short term memory :- Activation dynamics

Long term " - Synaptic "

S → NP VP

NP → det (adj)* noun

VP → VG [NP] [NP] PP*

VG → (Vaux)* V

PP → Prep NP

NP → NP PP

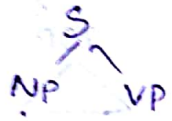
PP → NP age

Dependency tree : AGNT, PATNT, RECIP

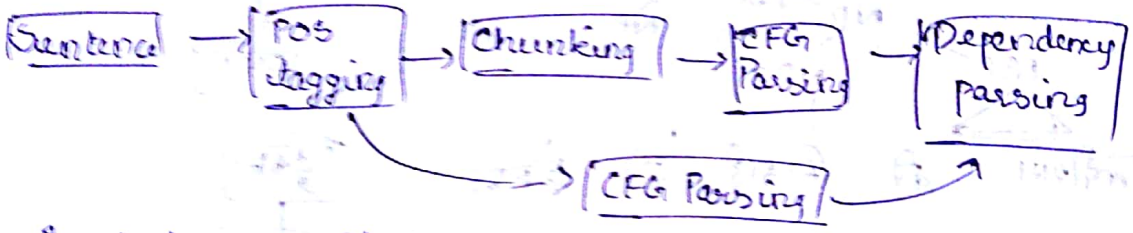
head

modifies

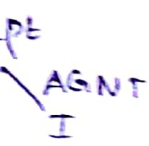
Phrase structure tree (PST) :



are → Vaux



I slept

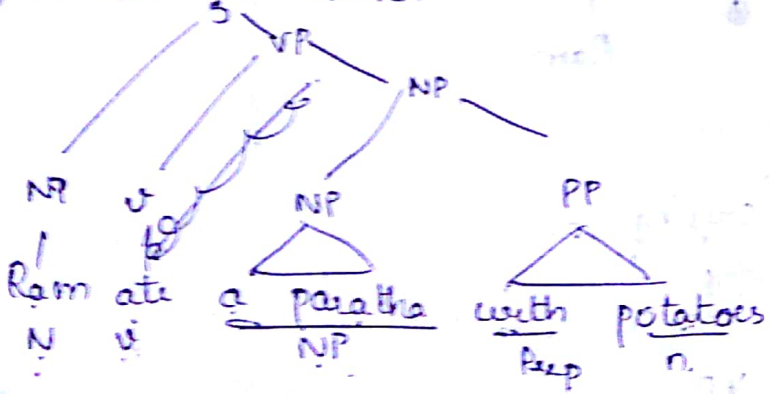


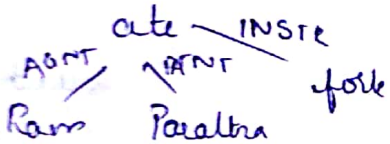
Tense can't be learnt by animals

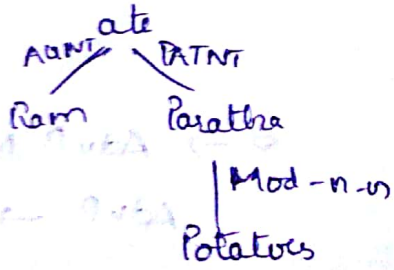
① Statistical based - Rules, grammar, semantics

② Statistical learning features

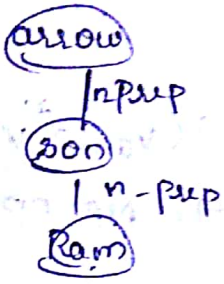
③ Neural net based







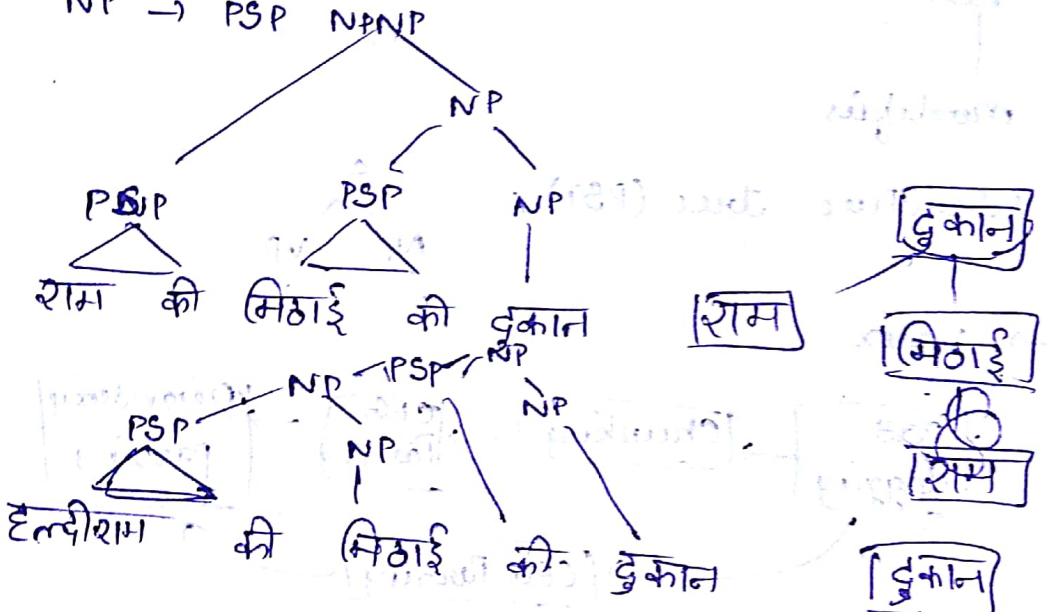
arrow of son of Ram

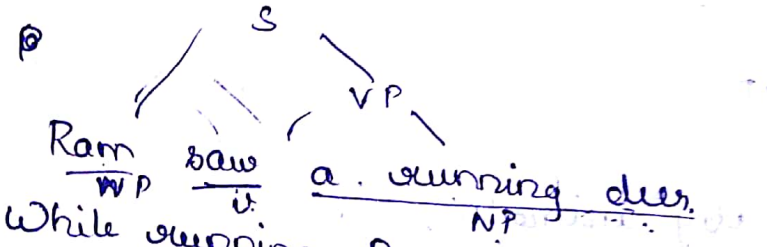


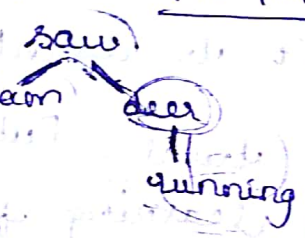
Hindi:

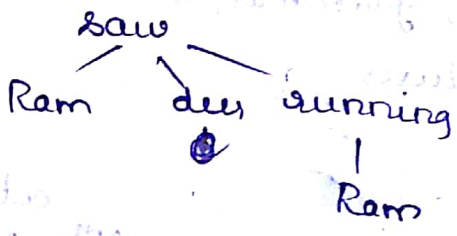
PSP → NP PSP

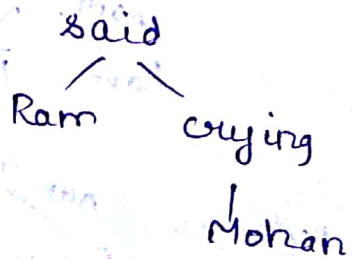
NP → PSP NP NP





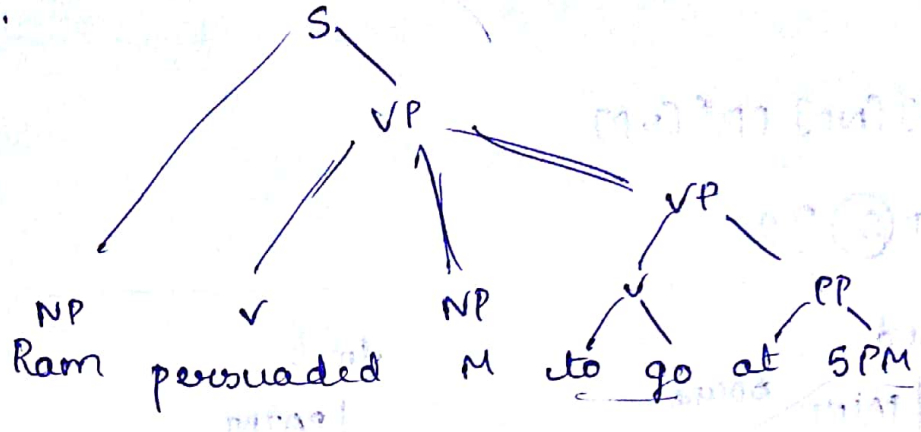


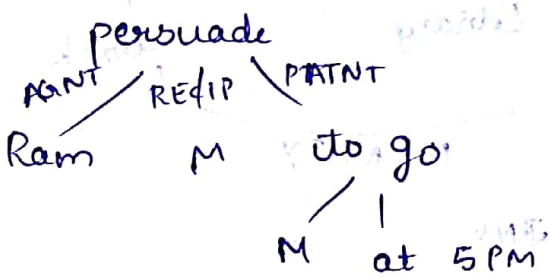




S → AdvP NP VP

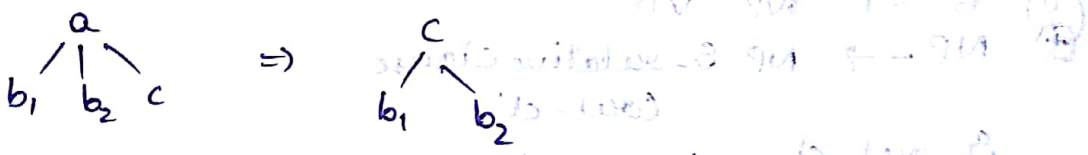
AdvP → adv/VP

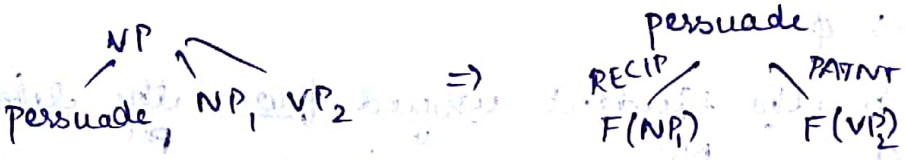




persuade – RECIP control

promise – ~~AGNT~~ "





NP ⇒ PATNT

If VP is present, VP ⇒ PATNT

NP ⇒ RECIP

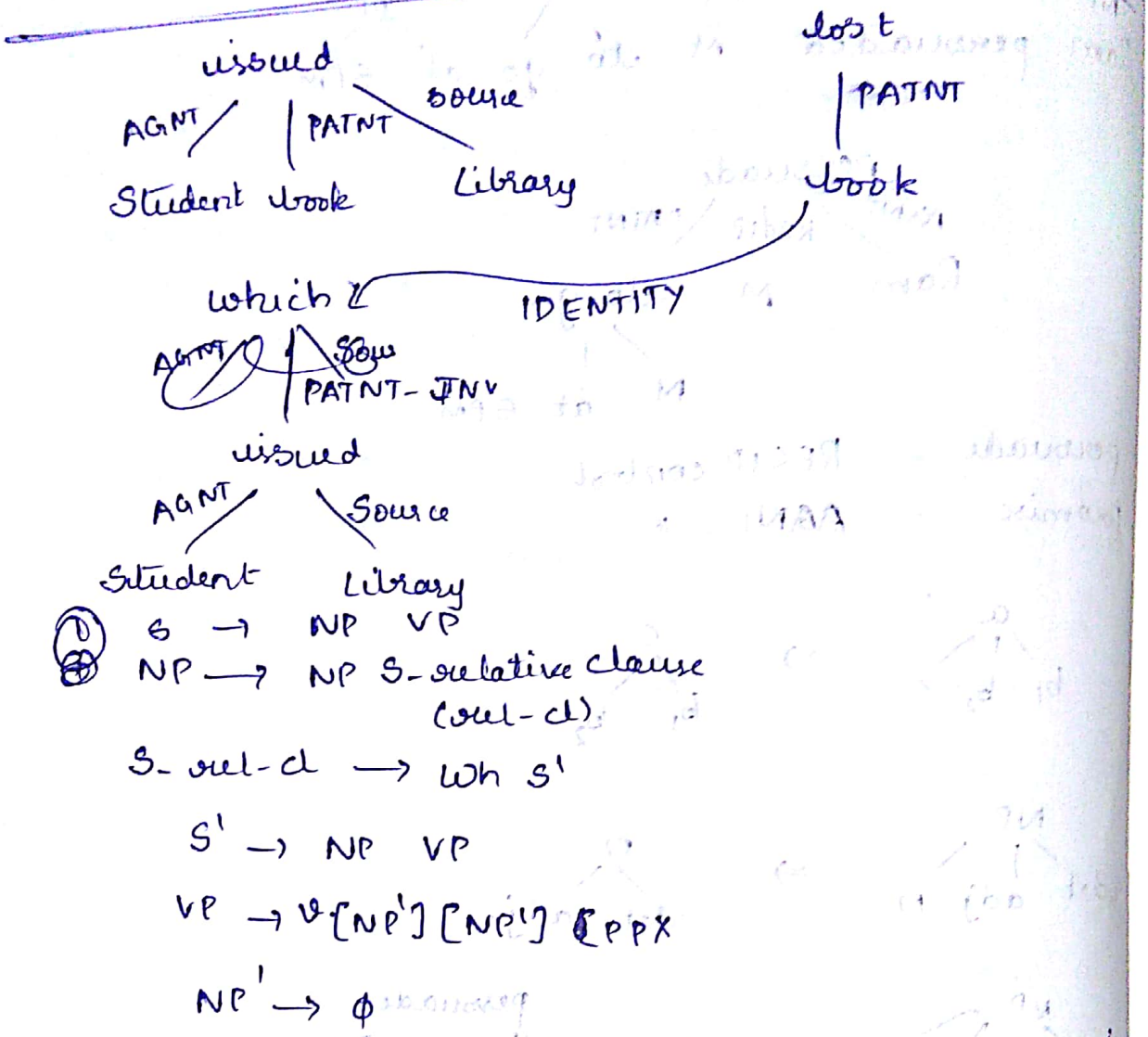
Active – Passive ⇒ MM tree same

PST different

Verb frames:

VP → V [NP] [NP] PP* [VP]

VP → V NP (S) ??



S → NP VP

NP → NP S-relative clause (rel-cl)

S-rel-cl → Wh S'

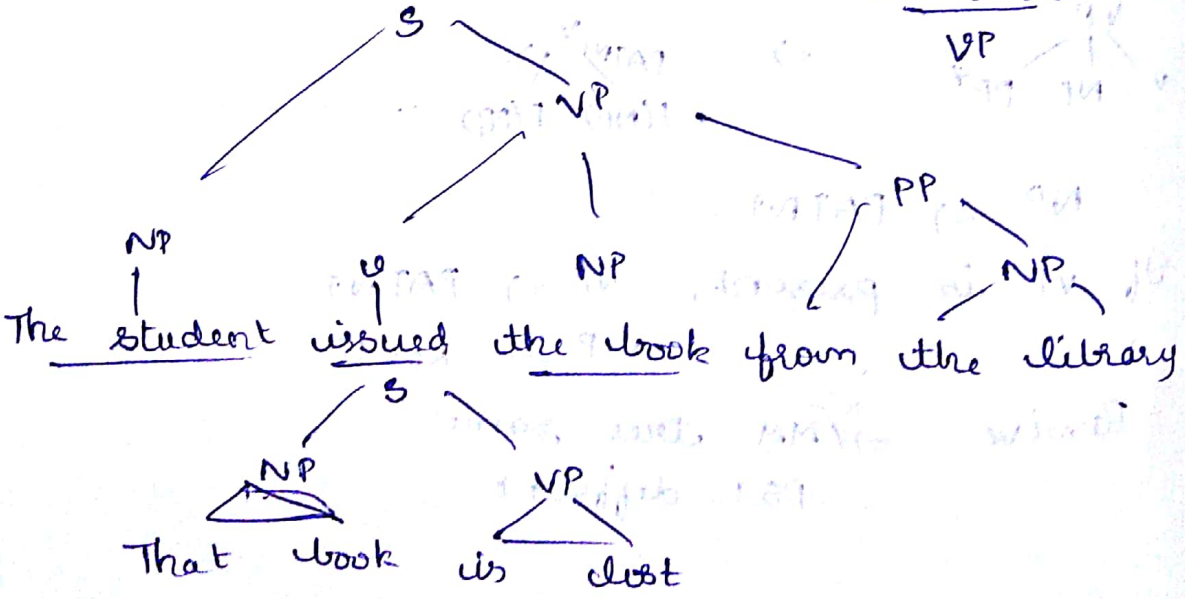
S' → NP VP

VP → V [NP'] [NP'] PPX

NP' → ϕ

The book which the student issued from the library is lost

NP R-cl NP V P/P VP



Object relativization

Subject

S ← [NP] VP

S ← Wh ∀Aux S'

2/11/2018

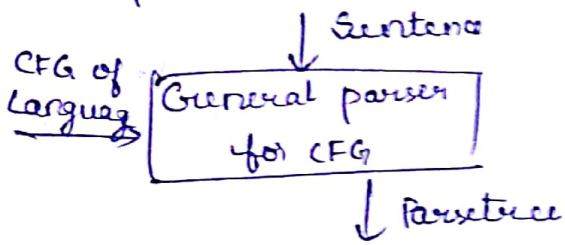
After Mid2 (RS)

Context-free grammar (CFG)

Parser gives the analysis of a sentence given CFG of that language.

After the ~~introduction~~ invention of CFG's, parsers for each language needn't be written seperately.

General parser is built.



The child plucked a flower
det n v det n.

R1 : S → NP VP

R2 : NP → det n

R3 : PP → prep NP

R4a : VP → v

R4b : VP → v NP

R4c : VP → v NP NP

R4d : VP → v NP NP PP*

⋮

Bottom-up Parsing algorithm:

The child plucked a flower
det n v det n

OPEN	Remaining Sentence	Remaining Rules	
–	1	ALL	
det	2	ALL	Shift
det n	3	ALL	Shift
NP	3	R3	Reduce
NP v	4	~~R4a~~ R4b	Shift
NP VP	4	R1 ~~R2~~	Reduce, POP X
S	4	~~R1~~ R2	~~Reduce~~ POP X

NP V det	5	ALL	– Shift
NP V det n	6	ALL	– Shift
NP V NP	6	R3	– Reduce
NP VP	6	R4c	– Reduce
S	6	R2	– Success ✓

Probabilistic CFG :

Rules, Probs

$A \rightarrow \alpha \mid P_1$

$B \rightarrow \beta \mid P_2$

$P(A \rightarrow \alpha \mid A) = P_1$

Introduce probability with lexical background.

LHS + Head of that phrase.

Dependency parser ⇒ NLP

Valency – demand charts

sleep

| AGNT

word (noun)

pluck

| AGNT PATNT

n v n

C- parsers

CFG Parsing

Bottom-up parsers : Generally fast, Less recursion

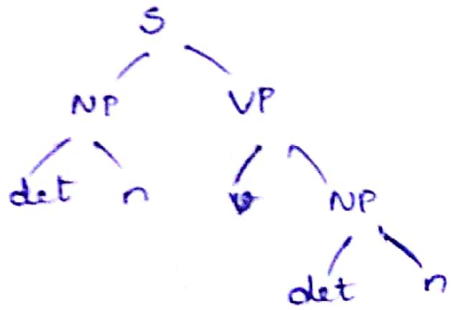
Top-down parsers

The[1] boy[2] plucked[3] the[4] flower[5]

det n v det n

Stack	Open	Remaining sentence	Remaining rules
1	nil	1	ALL
1'	det	2	ALL
1''	det n	3	R3
2	NP	33	ALL
2'	NP v	4	R4b
3	NP VP	4	R2
4 x POP	S	4	ALL
3' x POP	NP VP det	5	ALL
2'''	NP v det	5	ALL
2''''	NP v det n	6	~~ALL~~ R3
3	NP v NP	6	~~R3~~ R4c
4	NP VP	6	~~R4c~~ R2
5	S	6	~~ALL~~ ← Record success

Record :



PCFG

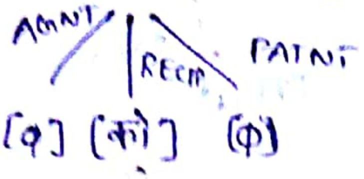
Dependency parsing:

Constraint graph

राम बच्चे को खिलौना देता है । Root

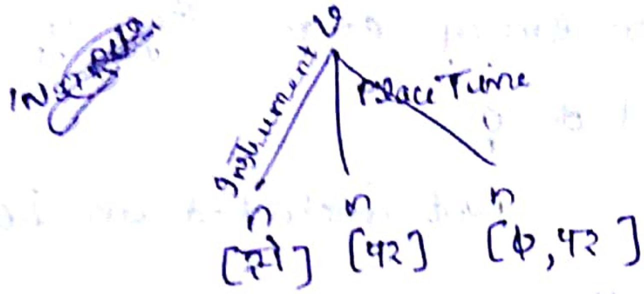
n n postprep n v aux verb

Mandatory दे [ता - है]



Di-transitive

Optional :



Solution graph (parse tree):

1. ~~Not more than~~ one incoming edge.
   Exactly

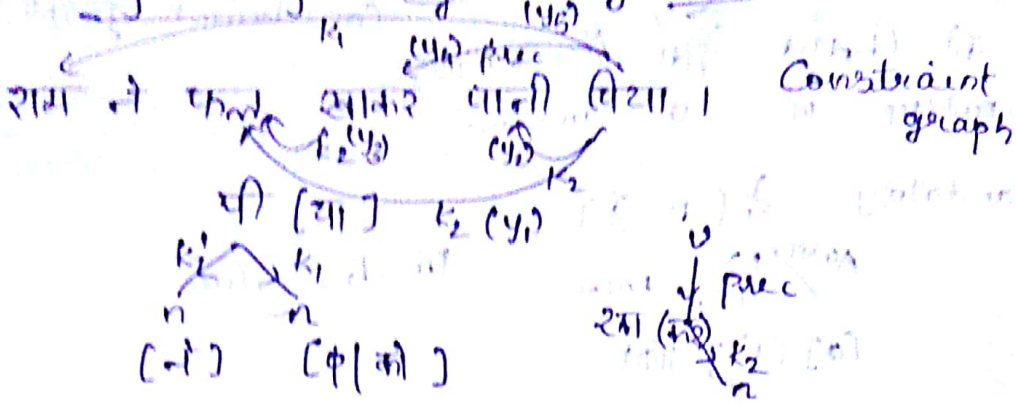
2. For ~~each~~ outgoing edges out of a node, there should be exactly one labelled edge corresponding to mandatory edges.

3. At most one optional edge.

A solution graph is a subgraph of constraint graph satisfying ①, ② & ③

Scanned by CamScanner

Parsing using Integer Programming:



Constraint graph

A variable for every edge. Eg: y_1

$y_1 = 1$ or 0

Not included in solution graph

Constraints: $y_5 = 1$ } Mandatory

$y_1 + y_2 = 1$

$y_3 \leq 1$ } Optional

$y_4 = 1$

$y_5 = 1$ } Incoming edges

$y_1 + y_3 = 1$

पी (या): k_1 mandatory n[को]

k_2 " n[φ|को]

रवा (कर): ~~k_1 mandatory [φ]~~ ×

k_2 optional n[φ|को]

prec mandatory ∨ –

Treebank (Penn treebank) Phrase-structured trees

10,000 sentences → tree

MALT Parser – Nivre

MST Parser (Max. Spanning tree) } Fundamentally different

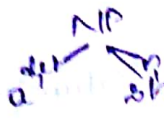
$\sum_j x_{ijk} = 1$

Global maximization

Local optimization (Uses Shift-Reduce algorithm)

Internally uses SVM's

Feature bundle / Feature structure :



Attribute → Value

children — n — [num : plural, root, child]

a : [num singular, specificity Indefinite]

sheep : [root sheep]

NP → det n

NP.num ⇐ n.num
NP.num = det.num
NP.head ⇐ n.root
NP.spec ⇐ det.spec

} Feature constraints

S → NP VP

S.num = VP.num
NP.num = VP.num

} Subject VP agreement

VP → v NP

VP.num = v.num
VP.gender = v.gender
⋮

} ⇒ FS(VP) = FS(v)

v
Feature structure

VP.head = v.root.

Dependency structure:

S → NP VP

S.AGNT = FS(NP)
↳ Feature structure

S.PATNT = VP.PATNT
S.RECIP = VP.RECIP
S.head = VP.head.

VP → v NP
↓
Transitive verb
v.trans = Yes | No | Ditrans

VP.PATNT = FS(NP)
VP.num = VP.num

Lexical Function Grammar (LFG)

Machine translation

HPSG => Head-driven Phrase Structured Grammar

Lecture-2 (NM)

3/8/2018

The task of seperating tokens in a given input sequence of words or any punctuation marks is counted as a token

Morphological
↑
Lexical level

Talib - aan
⇓ ⇓
Student Plural

Type, Token
⇓
Vocabulary word

High $\frac{\text{type}}{\text{token}}$ ratio ⇒ Higher inflection
⇓
Gharasamorcha

Morphological change?

Inflection: Change in the form of a word.

Language allows a lot of suffixes (can be added one after another).

Tokenization challenges:

① Hyphenation : state-of-the-art, hard-working

② Number : 9.8 ↳ Generally we seperate 9 and 8 but here no.

③ Dates

④ Abbreviations

⑤ Punctuations

⑥ URL's

⑦ Sentencifications

Regular expressions

N-grams and Zipf's laws:

Markov assumption : Word is affected only by its "prior local context" (last few words).

Unigram $\frac{c(w_i)}{N}$ $P(w_i | w_{i-1}, w_{i-2}) = \frac{c(w_i, w_{i-1}, w_{i-2})}{c(w_{i-1}, w_{i-2})}$

Bigrams $\frac{c(w_{i-1} w_i)}{c(w_{i-1})}$

Reliability vs Discrimination

Class estimation $<N, V>$ $<V, N>$

Lecture - 3 (NLP)

$P(o_1, o_2, \ldots, o_k) = P(o_k | o_{k-1}, \ldots o_1) P(o_{k-1} | o_{k-2}, \ldots o_1) \ldots P(o_2 | o_1) P(o_1)$

Markov assumption

$\simeq P(o_1) \prod_i P(o_i | o_{i-1})$

$\simeq P(o_1 | o_0) \prod_i P(o_i | o_{i-1})$

$\simeq \prod_i P(o_i | o_{i-1})$

Perplexity & Entropy

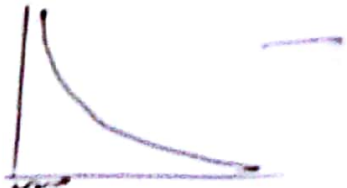
Smoothing & Backoff

Uniform Distribution : Highest entropy

Entropy $H(X) = -\sum_{x=1}^{n} P(x) \log_2 P(x)$

Perplexity $PP(X) = 2^{H(X)}$

Zipf's law : The product of frequency of words & their rank is approximately constant.



Grammatic content

$X = UDV^T$

$n \times k \quad n \times k \quad k \times k \quad k \times k$ → Diagonal

Orthogonal

Singular value decomposition

Softmax function $\frac{e^{x_i}}{\sum_j e^{x_j}}$

$\frac{d\sigma_x}{dx} = \frac{1}{1 - \sigma_x}$

A context predicts a word.

Lecture-4 (NLM)

Project → Spelling correction (18th)

Project → Grammar error detection

N-Grams and Smoothing:

① Laplace smoothing: Initialize the matrix with 1's instead of 0's

$$P_{Lap}(w_1 w_2 \cdots w_{n-1} w_n) = \frac{c(w_1 \cdots w_{n-1} w_n) + 1}{N + B}$$

N = No of seen n-grams

B = No of possible n-grams

② Lidstone's law:

$$P_{Lid}(w_1 w_2 \cdots w_{n-1} w_n) = \frac{c(w_1 \cdots w_{n-1} w_n) + \lambda}{N + B\lambda}$$

③ Held-out estimation:

$$P(w_1 \cdots w_n) = \frac{T_r}{N_r N}$$

$$P(w_1 \cdots w_n) = \frac{r}{N} = \frac{r \neq N_r}{N_r \neq N} = \frac{T_r}{N_r N}$$

④ Cross-over estimation

$$P(w_1 \cdots w_n) = \frac{(T_r^{01} + T_r^{10})}{(N_r^0 + N_r^1) N}$$

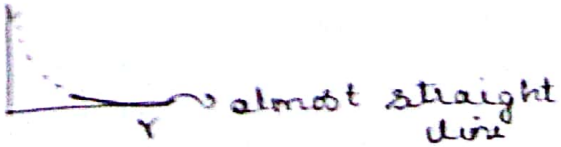
⑤ Good-Turing smoothing

$$r^* = \frac{(r+1)\, E(N_{r+1})}{E(N_r)}$$

$$P(w_1 \cdots w_n) = \frac{r^*}{N}$$

$r-1, r, r+1$ are all similar

$\frac{N_{r+1}}{N_r}$ Tells us a dot (Sudden ↑ or ↓)



Back-off is "using lower order N-grams for estimating higher orders"

Katz back-off

Linear interpolation

Language Modeling : Predict the probability with which a sentence belongs to a language.

25/9/2018

Lecture-5 (Ganesh)

Naive Bayes

Functional words : Highest frequent words (top 100)

Token : No. of words

Type : No. of unique words

$$P(\text{word}|\text{class}) = \frac{\#\text{ Word in class}}{\#\text{ Tokens in class}}$$

$$\log P(\text{Sentence}|\text{class}) = \sum_{i=1}^{N} \log P(w_i|\text{class})$$

$$P(\text{class}|\text{sentence}) = \frac{P(\text{sentence}|\text{class})\,P(\text{class})}{P(\text{sentence})}$$

$$P(\text{senti}|\text{sent}) = P(\text{sent}|\text{senti})\,P(\text{senti})$$

$$= \sum \log(P(w_i|\text{senti})) + \log P(\text{senti})$$

* Tokenization as classification problem

Word boundary, Sentence boundary, char

(, | ;) (. | !)

28/8/2018 Lecture - 6 (MM)

Good Turing : $c^* = \frac{(c+1)\, N_{c+1}}{N_c}$

Church & Gale : jii ⇒ Join if independent

N_c can be zero?. Then?

So, put c-5 to c+5 in a bucket.

So, N_c contains [c-5, ... c, ... c+5]

(Read this paper) Great one.

Simple good turing : $N_0 = ar^b$

Pabsolute discounting $(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1}) P_{cont}(w)$

Kneser - Ney smoothing :

$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$

$= \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$

$P_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{cont}(w)$

$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$

Written - bell smoothing :

$T(w_x)$ = No. of seen n-gram types starting with w_x.

$Z(w_x)$ = No. of unseen n-gram types starting with w_x.

$N(w_x)$ = No. of n-gram tokens starting with w_x

$P^*(w_i|w_x) = \frac{T(w_x) \times 1}{Z(w_x)(N(w_x) + T(w_x))}$ ⇒ Prob. of unseen unigrams

$P^*(w_i|w_x) = \frac{c(w_x, w_i)}{N(w_x) + T(w_x)}$ ⇒ Prob. of seen unigrams

5/8/2018

Lecture - 7

Learn hierarchy of languages (Regular → Context free → Context sensitive ..... → Natural languages)

POS tagging & chunking

Statistical model

① { Distributions
② { Assumptions

Deep learning models (DL)

Representations
Architectures

Not provable

Use statistical & DL together for better results

$$P(T|w) = P(t_1, t_2, \ldots t_n | w_1, w_2, \ldots w_n)$$

$$= \prod_1 P(t_i | w_1, w_2, \ldots w_n)$$ (Assumption: Tags are independent of each other)

$$= \prod_1 P(t_i | w_i)$$ (Assumption: Tag of a word is independent of all other words)

Parameters : A, B, Π

Forward procedure

Backward procedure.

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j)\, a_{ij}\, b_j(o_{t+1})$$

Expectation → First moment

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot \beta_{t+1}(j) \cdot a_{ij}\, b_j(o_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}$$

$\sum_t \xi_t(i,j)$ ⇒ How many times j is seen when i is seen.

$\sum_t \sum_j \xi_t(i,j)$ ⇒ How many times i is seen.

$$\gamma_t(i) = \sum_j \xi_t(i,j)$$

$$\bar{a}_{ij} = \frac{\sum_{t}^{T-1} \xi_t(i,j)}{\sum_{t}^{T-1} \gamma_t(i)}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} \gamma_t(i) \quad \text{s.t. } o_i = v_k}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\bar{\Pi}_i = \gamma_1(i)$$

Sentiment Analysis
N Hotencoding
NB model for SA
Log Reg for SA
Hypothesis

Generative vs Discriminative

Sentiment analysis < +, - / +, -, neutral, sarcasm / +, -, neutral

N-hot encoding : Represent sentence as 1-hot vector of words.

$$P(\text{Senti}|\text{Sentence}) = \frac{P(\text{Sentence}|\text{Senti})\, P(\text{Senti})}{P(\text{Sentence})}$$

$$P(+|x) = h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

8/10/2018

Word2vec:

Continuous bag of words (CBOW):

Skip gram (SG): Not possible in practical. At max 1 word can be predicted

Neural networks don't work with discrete data.

Use Real-valued vectors.

Feed-forward NN's

Recurrent NN's.

Why Softmax? Small changes in val will bring huge difference in e^{val}.

Link it back to viterbi

$h_i = \sigma(W L_{i-1} + U E_{x,i})$ (80% 20%)

→ Brief

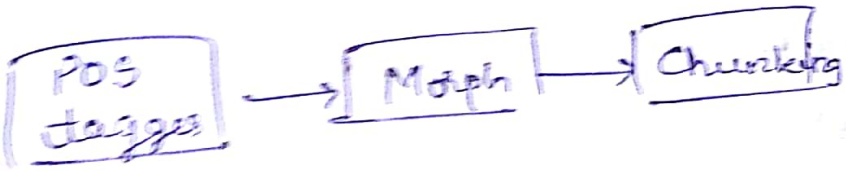
Brevity ⟷ Ambiguity

Natural language: Easy to convey thoughts. Just say half, listener will understand whole meaning. Whereas in programming language, you need to define/say things at the first level.

Rule-based

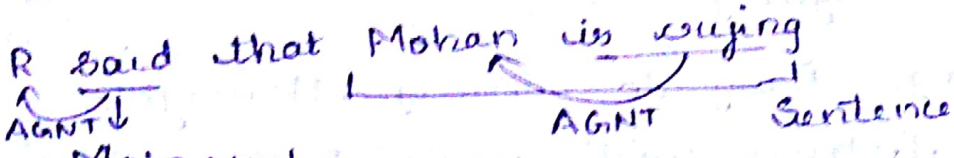
Statistical methods

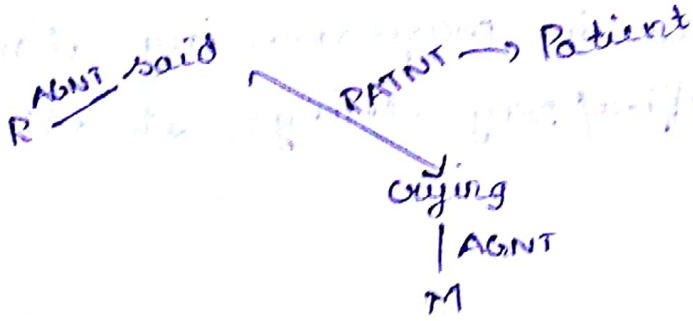
Artificial neural networks



26/10/2018

AGNT ⇒ Agent





Why only one tree for a sentence? Why not 2/3?
Sentence is a one connected structure

Participial verbs modify
→ nouns (adjectival)
→ verbs (adverbial)

NP ⇒ det (adj)* n.

NP ⇒ det (Adj Phrase)* n

Adj Phrase → Adj
→ VP

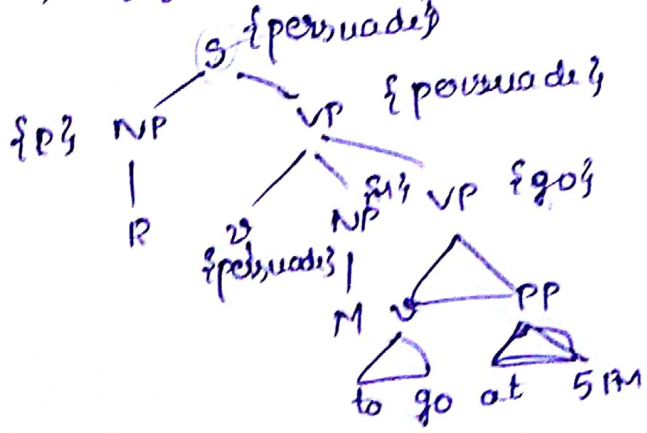
VP ⇒ v [NP] [NP] PP* [~~adv~~] [AdvP]

RB → Boolean

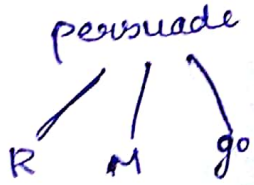
AdvP → adv | VP

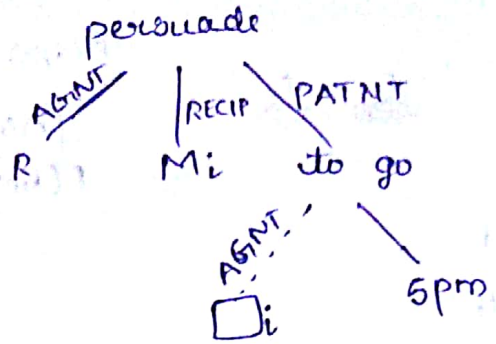
R persuaded M to go at 5PM
R promised M to go

VP → v [NP] [NP] [VP] PP*



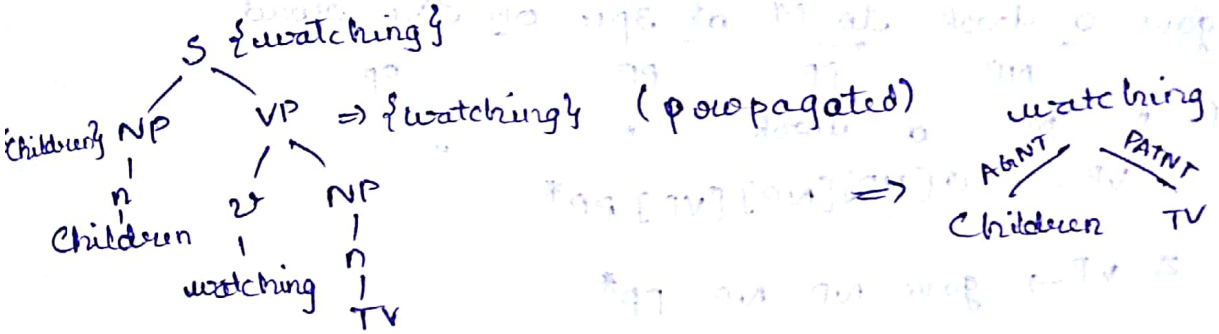
F ⇒

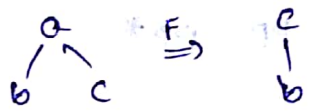


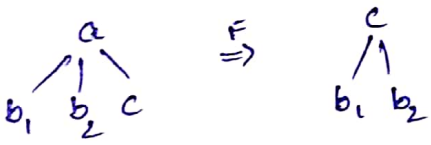


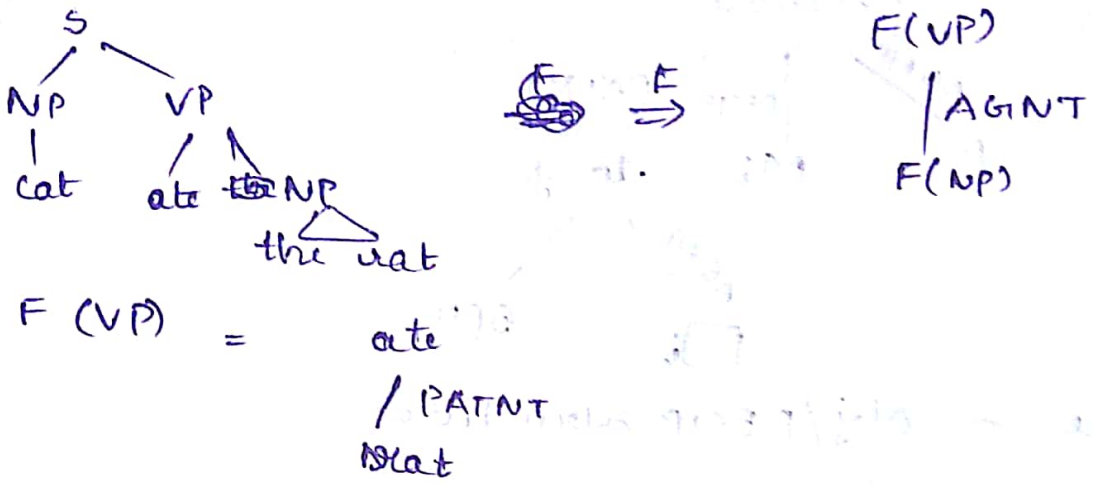
persuade — Obj/RECIP structure

promise — AGNT control









1) R gave a book to M at 3pm on the road
   NP PP PP PP

2) R " M a book " "

VP → V [NP] [NP] [VP] PP*

2) VP → gave NP NP PP*

1) VP → gave NP PP PP*

The child plucked the growing flower

The child plucked the flower growing in the garden

The child plucked the flower walking in the garden

Relative clauses

Verb v