client

Adapter

configuring

-ve in between the two

tion of syntax

NLP

• Not a finite automata
(no memory)

Bottom up parsing / bott

Parse tree

possible parse tree

the individual parse tree probability

sum them to get the probability of sentence.

Antecedent → Anaphor

coreferring

1 +ve, multiple −ve in between the two

Statistical Parsing:

↳ verification of syntax

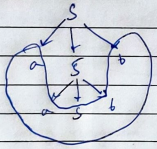
$S \rightarrow aSb$

$S \rightarrow ab$

↗ Not a finite automata (no memory)

$L = \{a^n b^n : n \geq 1\}$

Top down



→ aabb

Bottom up parsing / both

Parse tree

Given a sentence

↓ generate all possible parse tree

↓ calculate individual parse tree probabilities

↓ sum them to get the probability of sentence.

Computers ← communication in everyday language → People

Programming language
→ deterministic, context free
→ parsing in O(n)

Date
Page

Introduction :

Imp slide page : 8

→ Turing test : Test conducts
Human ↙ ↘ Machine

Application:
→ Translation
→ Text to speech, speech to text
→ OCR (optical character handwriting recognition)
→ Sentiment Analysis (public opinion)
→ Chatbots
→ Summarization
→ Information retrieval

* Ambiguity in NLP:

→ Phonetics and phonology → (bank → river, bank → institution)
→ Morphology → (carried = carry + ed) morpheme
→ Lexical analysis → (dog → noun, dog → verb) (dictionary access)
→ Syntactic analysis → (Pos Tagging) (Parsing)
→ Semantic analysis → (word sense disambiguation) (role labelling) (Textual entailment)
→ Pragmatics → model user intention
→ Discourse → processing of sequence of sentences (use reasoning & world knowledge)

Computer vision
Search
logic & knowledge representation
ML
Robotics
Planning
Expert systems
NLP

* Anaphora resolution / co-reference :

John put the carrot on the plate and ate it.

* Information Extraction :

→ Named entity recognition (NER) : identify names of people, places, org, etc from a sentence.

→ Extract relations b/w entities

* Question Answering

* Text summarization

* Sentiment Analysis

* Machine translation

Morphology
↓
Pos tagging
↓
Chunking [NP] [VP] [NP] [VP] [PP]
↓
Parsing
↓
Semantics
↓
Discourse coreference

Increased complexity of processing

ML Algo :
HMM, Max Entropy, CRF, SVM, logistic Reg., RNN

Statistical / corpus based :
ML methods used to acquire knowledge from annotated text corpus

apposition ?

Date
Page

→ **Features**:

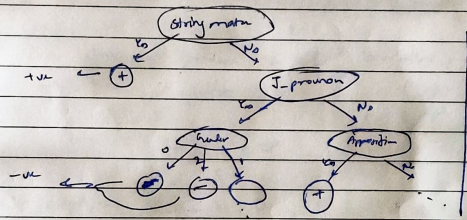
→ string match, alias, number agreement, gender agreement
→ pronoun, def, dem. (NP type)
→ distance (& b/w anaphor & antecedent in terms of sentences) (0, 1, 2, ...)
→ semantic class agreement

Def → definite
Dem → demonstrative
non-pronoun

→ **Model**: Decision Tree:

Person
Male / Female

Object
org / location / ...



Soon et. al

→ **Decoding**: (right to left, consider each antecedent (i) until classifier returns true)

Now the model is already trained, we need to test it on a test.

∀ pair of noun phrases ⟨i, j⟩, check if Prob(⊕ | ⟨i, j⟩) > Prob(⊖ | ⟨i, j⟩)

if true → they coreffer each other
else they don't

High recall: right to left, consider each antecedent & select the highest among one

→ **Evaluation**: Precision, Recall, F-score

→ Finally returns coreference sets (disjoint sets)
(complete coreference chain)

Tree bank (multiple parse trees) —Supervised PCFG training→ English (S → NP VP 0.9, S → VP 0.1, ⋮)

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{Count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

count(α) → all rules in which α is on LHS

Date ____
Page ____

PCFG : Maxm. likelihood Training,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

Coreference Resolution, (anaphora ")

John put the carrot on the plate and ate it
- carrot → antecedent
- it → anaphora (act of carrying back)

Cataphora → when anaphora precede antecedent

Because she ..., Mary was asked ...

Anaphora resolution : i) rule based ii) statistical based / ML based

Constraints :
- → Number agreement
- → gender "
- → person & case
- → syntactical agreement
- → selectional agreement

* ML based :

→ Preprocessing :
- → POS Tagger
- → Noun phrase identification
- → Named Entity recognition
- → Nested noun phrase extraction

} HMM based

Encoding

→ Generate training instances : noun phrase (antecedent)

noun phrase (antecedent)		anaphor	label
Israel	←	the Jewish state	+
US		"	−
Iraq		"	−

b/w anaphor & marked antecedent { correct pair → +ve label; all other pairs of noun phrase & anaphor → −ve

Features (domain independent):

- local context : preceding & succeeding words
- → POS tags (current & surrounding tokens)
- → Word length (shorter word → rarely NEs), Infrequent → most prob NE
- → Prefix / suffix (fixed length strings)
- → Uppercase / Alphabets / Digits
- → Gazetteer → NE list
- word normalization
- previous occurrence (its class label)

Statistical Parsing:

PCFG : Probabilistic Context free grammar

e.g. sentence : Book the flight through Houston

Prob (sentence) = Σ Prob (all possible parse tree derivations)

P (parse tree) = product of probabilities of production

S → NP VP 0.8
S → Aux NP VP 0.1
S → VP 0.1
} $\Sigma = 1$

PCFG tasks,

i) Observation likelihood → Probability (sentence)

ii) Most likely derivation : for all parse tree derivations, select the one with maximum probability : use CKY parser (bottom up)

iii) Maximum likelihood training

* Inside Algorithm: CKY parser with slight modification : (analogous to forward algo of HMM) combine probability of multiple derivations of any constituent using addition instead of max

(Used for observation likelihood)

→ Maxe Entropy

– begin, – end, – unique

~~[illegible]~~ – non (non-entity)

IOB → begin; intermediate; other

Vivian Richards player
B I O

Date ____
Page ____

NERC : Named Entity Recognition & Classification :

Named Entity → Extract proper noun

Classification → classify into ~~[illegible]~~ categories like person, organisation, location, etc. (Also – date, time, money, percent, ~~number~~, measurement)

→ Rule based
→ Statistical based

– global features
– local features

HMM Based NERC : $\hat{T} = \arg\max_T [P(W|T)\ P(T)]$

Bigram model :
prob. of a tag depends only on previous tag

transition ↙ $P(T) = P(t_1) \cdot P(t_2|t_1) \cdot \ldots \cdot P(t_n|t_{n-1})$

emission ↙ $P(W|T) = P(w_1|t_1) \cdot P(w_2|t_2) \ldots P(w_n|t_n)$

Recall
Precision
Fscore
Accuracy

2nd order HMM:
$P(T) = \Sigma P(T_i, T_{i-2} T_{i-1})$
Current tag depends on previous 2 tags
(Trigram)

Ensemble Learning with Genetic Algo

RNN vs CRF 2

→ Initialization, fitness computation, selection, cross over, mutation, termination

Majority ensemble
Weighted ensemble
Stacked ensemble

Biomedical tags:
– rna, – protein, – dna,
– cell type, – others

Slide-203

Pattern – data – disambiguation

DL continued

Date
Page

CNN :

Example: Image → 227 × 227 × 3 → RGB channels
Filters → 11 × 11 × 3 (depth of filter & image assumed to be same)
↳ fx ↳ spatial extent

Padding = 0
Stride = 4
# filters = 96

Output → $W_2 \times H_2 \times D$

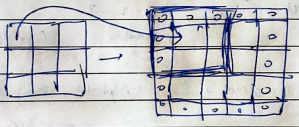
Each filter will produce a 2D convolution / feature map
⇒ D = # filters

(3D → 2D since depth of filter = depth of image)

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$\therefore \text{output} = \frac{227 - 11 + 0}{4} + 1$$

$$= 55 \times 55 \times 96$$



I = 3×3
F = 3×3

Padding = 1

for $W_2 = W_1$ $\quad W_2 = W_1 - F + 2P + 1$

$$\therefore P = \frac{F-1}{2} = \frac{3-1}{2} = 1$$

eg: edge detector kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Hyperparameters:

- Batch normalization after every conv. layer
- Xavier/2 initialization
- SGD + Momentum (0.9)
- learning rate : 0.1
- mini-batch size → 256
- Weight decay → $1e^{-5}$
- No dropout used

Lecture 12 :

GoogLeNet :

→ at each layer, instead of using multiple filters of same dimension, use filters of different dimensions & then concatenate the feature maps (with adequate padding)

→ eg: $(H \times W \times D)$ ← image $(F \times F \times D)$ ← filter

Output → $(H - F + 1)(W - F + 1)$

computations for each pixel = $(F \times F \times D)$

→ To reduce # computations : → Use (1×1) convolutions

$F \times F \times D$ → ~~$(F \times F \times D)$~~ convolutions

Use (1×1) convolutions with D_1 filters :

output → $(F \times F \times D_1)$ where $D_1 < D$

→ At the end before FCN → eg: ~~1024~~ $7 \times 7 \times 1024 = 1024 \times 49$

Use average pooling → $1 \times 1 \times 1024$

(to prevent too much parameters)

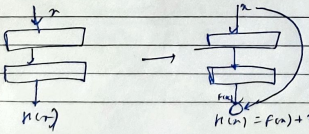
→ 12x less params than AlexNet

2x more computation (multiple layers : conv, max-p, inception)

ResNet :

→ deeper CNN should perform better but performed worse as it couldn't learn identity function well

→ problem lies in initialization values



learning $F(x) = 0$ is easier than $F(x) = x$

Maxpool doesn't change the depth or add new parameters
It's just a max operation
(Max pooling operation is per feature)
(not across the depth)

Date
Page

Alex Net:

Input → Conv1 → Maxpool1 → Conv2 → Maxpool2 → Conv3 → Conv4 → Conv5

→ maxpool3 → FC1 → FC2 → Output
(27×256) (9216) (4096) (1000)

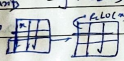
Layers = 5 + 3 = 8
(conv) (FC)

(maxpool not counted as no parameters)

* 27.55 Million params

(Params in FC: 4M + 16M + 4M ≈ 24M)

NOTE: * In every convolution operation, non-linearity (eg ReLU) is also applied to the volume



* Now the pooling operation can be performed.

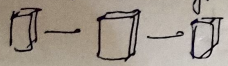
ZF Net: Same # Layer, more parameters — better

Vgg Net: Each filter → 3×3
16 Layers

Input → C C M C C M C C C M C C C M C C C M F F F
conv maxpool fully connected

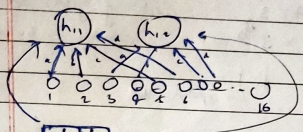
Params in FC Layer ≈ 122M
Non FC → 16M

CNN → multiple convolution filter → multiple layers



Properties:

i) Sparse connectivity:



filter → 2×2
stride = 2

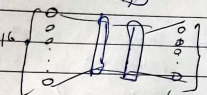
h_{11} only depends on pixel 1, 2, 3, 4

not all the pixels

h_{12} → 3, 4, 7, 8

To increase receptive field, increase depth

Image classification
eg MNIST



ii) Weight sharing:

Using only 1 filter, no. of weights / parameters = 2×2 = 4 (for first hidden layer)

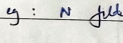
→ underfitting

Sol^n → multiple filters (kernels)

eg: N filters → N×2×2 (N×f×f)

parameters (for 1st hidden layer)

(actually N×f×f×D, no. of filters (depth))

Eg:  (Alphabet classification)

I: (14×14×6) → S=1, F=5, K=16, P=0 → (10×10×16)

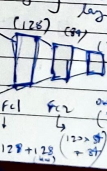
Params = (5×5×6) × 16 = 2400

filter size # filters

(10×10×16) → Maxpool, S=2, F=2, K=16, P=0 → (5×5×16)

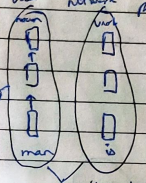
Params = 0

flattening

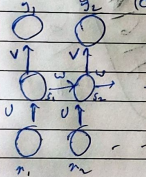


RNN :

- input size is not fixed
- inputs are dependent on each other
- that network performs the same task



↳ Here two networks perform same task (input → words, output → tags)



(Generate output at every time step)

$s_i = \sigma(U x_i + W s_{i-1} + b)$ ← bias

$y_i = O(V s_i + c)$ ← output activation

$y_i = f(x_i, s_i, W, U, V, b, c)$

$x_i \in R^n$ (n-dimensional input)

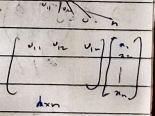
$s_i \in R^d$ (d-dimensional state)

$y_i \in R^k$ (say k classes)

$U \in R^{d \times n}$

$V \in R^{d \times k}$

$W \in R^{d \times d}$

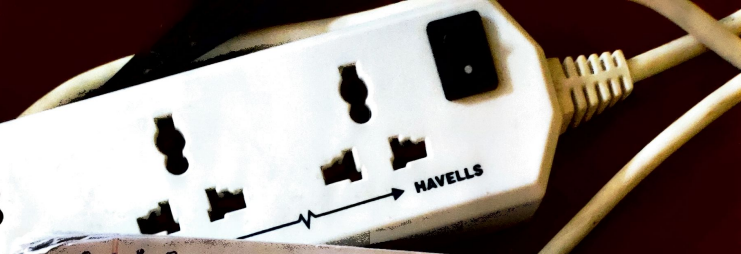


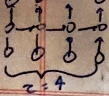
We have N neural networks

Total Loss → $L(\theta) = \sum_{t=1}^{N} L_t(\theta)$

$L_t(\theta) = -\log(y_{tc})$

↳ predicted probability of true character at time step t





$T = 4$

$E_t = \sum_{k=1}^{out} (\hat{y}_{tk} - y_{tk})^2$

$E = \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{out} (\hat{y}_{tk} - y_{tk})^2$

classes of output

total loss (summation of loss from all time steps)

$h_t = U x_t + W \phi(h_{t-1})$

$y_t = V \phi(h_t)$

$\phi \rightarrow$ activation fn

$h \rightarrow$ hidden pre-activation

Now to find the parameters V, U, W

complete $\rightarrow \frac{\partial E}{\partial W}, \frac{\partial E}{\partial V}, \frac{\partial E}{\partial U}$

$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_k} \cdot \frac{\partial h_k}{\partial W}$

next page

$\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \cdot \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{k+1}}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}$

$\because h_i = U x_t + W \phi(h_{i-1})$

$\frac{\partial h_i}{\partial h_{i-1}} = 0 + W^T \text{diag}[\phi'(h_{i-1})]$

$L = ab$

$\frac{\partial L}{\partial z} = a^T \frac{\partial b}{\partial z} + b^T \frac{\partial a}{\partial z}$

$\left(\frac{\partial W}{\partial h_{i-1}} = 0\right)$

If $\frac{\partial h_i}{\partial h_{i-1}} < 1 \quad \forall i \Rightarrow \frac{\partial h_t}{\partial h_k} \rightarrow 0$ (exponential operation) $\left(\frac{\partial h_t}{\partial h_1} \rightarrow 0\right)$

$\frac{\partial h_t}{\partial h_k}$ look back upto a certain no. of steps

Vanishing gradient

Then $\frac{\partial E}{\partial W} \rightarrow 0$ or the weight matrix will stop updating its weights or

$\left(W' = W - \eta \nabla W \Rightarrow W^t = W\right)$

RNN won't learning

If $\frac{\partial h_i}{\partial h_{i-1}} > 1 \quad \forall i$, $\frac{\partial h_t}{\partial h_k} \rightarrow \infty$ ⇒ exploding gradient

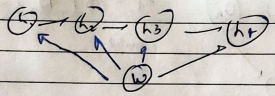
$\frac{\partial h_i}{\partial h_{i-1}}$ — during the gradient of $\frac{\partial L}{\partial W}$

RNN weights updates drastically

$\frac{\partial h_t}{\partial w}$ → vector (d) = tensor (d × d × d) → matrix (d × d)

Let hidden layer has dimension → d

Date _____ Page _____



$h_t = U x_t + W \phi(h_{t-1})$

$\frac{\partial h_t}{\partial w} \neq \phi(h_{t-1})$

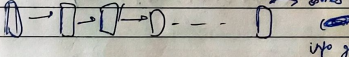
$\phi(h_{t-1})$ also depends on W

$\frac{\partial h_t}{\partial w} = \frac{\partial^+ h_t}{\partial w} + \frac{\partial h_t}{\partial h_3} \cdot \frac{\partial h_3}{\partial w}$ → Now again repeat

explicit (treat all other inputs as constant)

implicit

→ selective read, selective write, selective forget



→ stores all info from the start (overload)

info got morphed beyond recognition

LSTM → solves the problem of vanishing gradient

- $s_t = f_t \odot s_{t-1} + i_t \cdot \tilde{s}_t$

  selective forget, selective read

- $h_t = s_t \odot o_t$

  selective write

read, forget, write

$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$

$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$

$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$

$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$ or tanh

s comm

Γ_f → forget

Γ_i → read

Γ_o → output

In GRU → ~~forget gate~~ → forget gate & input gate are tied

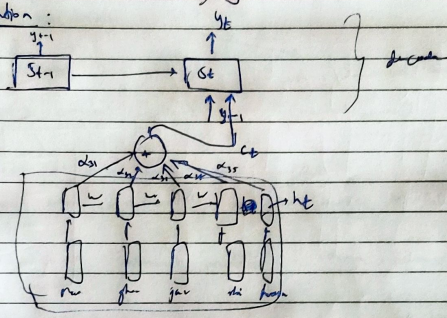
→ Gates depend on s_{t-1} & not h_{t-1}

$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$

$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$

$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$

Attention :



The output "going" depends on input words "jar" and "the" so at $t=3$ α_{t2} and α_{t4} should have more weight

Encoder : $h_t = RNN(h_{t-1}, x_t)$

$s_0 = h_T$

Decoder : $\tilde{\alpha}_{tj} = F(s^{t-1}, h^j)$

(contribution of the j^{th} input word at timestamp t)

$\tilde{\alpha}_{tj} = e_{tj} = V_a^T \tanh(W_a s_{t-1} + U_a h_j)$

scalar

R^k learnable parameters $R^{k\times k}$ $R^{k\times 2k}$

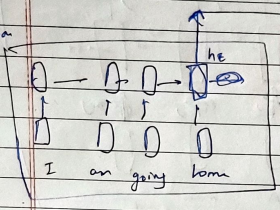
$\alpha_{tj} = \text{softmax}(e_{tj}) = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$

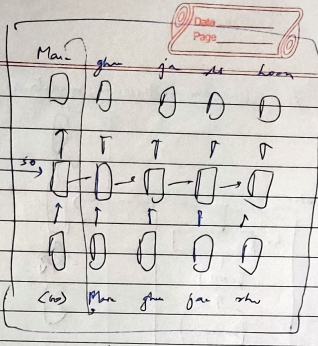
* Attention is itself a FFNN with the parameters V_a, W_a, U_a

$c_t = \sum_{j=1}^{T} \alpha_{tj} \cdot h_j$

$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$

Machine translation :





$h_t = RNN(h_{t-1}, x_t)$

$s_0 = h_t$

$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$

or

$s_t = RNN(s_{t-1}, [h_t, e(\hat{y}_{t-1})])$

↓

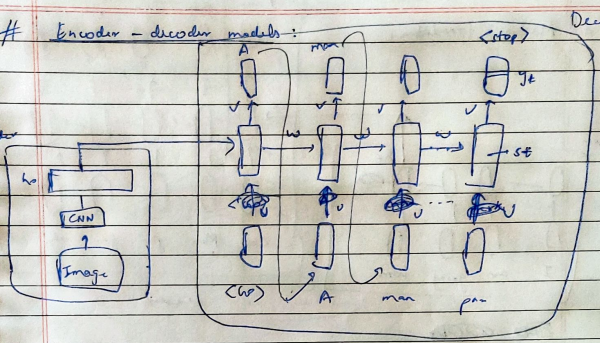
concatenation of output from $(t-1)$ & encoding h_t

i.e h_t is fetched in every neural network

$P(y_t | y^{t-1}, x)$

$= softmax(V s_t + b)$

Encoder - decoder models :



→ At every time stamp, we have to predict

$P(y_t = j \mid y_{t-1}, I)$ → probability distribution of the j^{th} word at time stamp t

$= P(y_t \mid s_t, f(C7(I))$ ($j \in$ English vocab)

(s_t encodes all info. from 0 to $t-1$)

→ Input at time stamp t = output predicted at time stamp $t-1$ (eg one hot vector to convert input to vector)

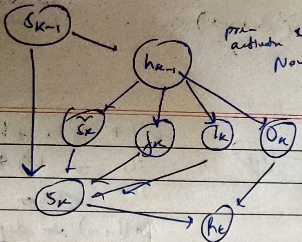
→ Rather than feeding the encoded image at $t=0$, we can feed it to all neural networks

→ Output j^{th} → softmax

loss → Σ cross entropy (across all time stamps)

$s_t = \sigma(W s_{t-1} + U x_t + b)$

$P(y_t = j \mid y_{t-1}) = \text{softmax}(V s_t + c)_j$



pre-activation $h_t = Ux_t + w\,\phi(h_{t-1})$

Now onwards ↓ (here bias is ignored)

$h_t = \phi(W_a[h_{t-1}, x_t] + b_a)$
↓
activated.

→ Starting from s_{k-1} & h_{k-1}, we reached s_k and h_k.

Different Notation :

We have $m_{t-1} = h_{t-1}$ → memory cell

$$\tilde{m}_t = \tanh(W_c[m_{t-1}, x_t] + b_m)$$

Update gate : $\Gamma_u = \sigma(W_u[m_{t-1}, x_t] + b_u)$
(≈ input gate)
(read)

Now $m_t = \Gamma_u \odot \tilde{m}_t + (1 - \Gamma_u)\, m_{t-1}$ ← simplified GRU

Even if $\Gamma_u \to 0$ due to vanishing gradient, m_t won't be 0

Full GRU :

Relevance gate : $\Gamma_r = \sigma(W_r[m_{t-1}, x_t] + b_r)$
(~~write~~ output gate)
(write)

$h_t = \Gamma_r \odot m_t$

LSTM :

$\Gamma_f = \sigma(W_f[h_{t-1}, x_t] + b_f)$
$\Gamma_o = \sigma(W_o[h_{t-1}, x_t) + b_o)$
$\Gamma_u = \sigma(W_u[h_{t-1}, x_t] + b_u)$

$\tilde{m}_t = \tanh(W_m[h_{t-1}, x_t) + b_m)$
$m_t = \Gamma_u \odot \tilde{m}_t + \Gamma_f \odot m_{t-1}$
$h_t = \Gamma_o \odot m_t$

NLP Continued

Date 5th Sept
Page Sun day

Sentiment Analysis :

- Aspect Based (fine grained) (battery, camera, display)
- Sentence / Document based (coarse grained) (overall phone)

→ polarity → +ve, -ve, neutral, conflict

→ Aspect term extraction, and then identify its polarity

→ CNN based sentiment analysis : 2gram, 3gram → etc

→ Hybrid : CNN + optimized feature set (using GA based optimizer)
+ Train SVM using optimized features & trained FFNN

Optimized features → emoticons, punctuations, #

RNN / LSTM based :

→ LSTM

→ Target dependent LSTM

→ Target connection LSTM

source → target
src → target

Out of Vocabulary words → translate into target language & perform tasks in target bilingual embedding

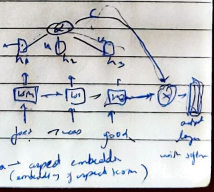
BLUE
using skip gram word2vec model

(source bilingual word embedding (BLUE) share the same vector space as target BLUE)

Attention mechanism for classification →

→ Attention based LSTM

→ ~~Aspect~~ " " " with Aspect Embedding

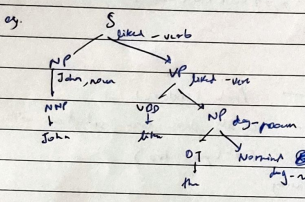


Maximum Likelihood training : sentences → PCFG
— unsupervised

Date
Page

* lexicalization of PCFG :
→ include head word in LHS non-terminal
most central to the phrase
+ POS

eg.



only for non-terminals
non-terminal
A <> → B <> C <>

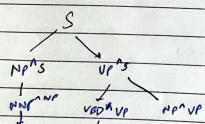
Head for VP → verb
NP → noun
PP → preposition
S → head of VP

* Collin's parsers → slides

* Splitting non-terminals :
→ more contextual info
→ extra parent annotation

Non-terminal → new non-terminal

eg: VP^S → VBD^VP NP^VP



* Evaluating : Correct tree T

total constituents = z

Computed tree P

correct constituents = x
total constituents = y

Recall = x/z
Precision = x/y

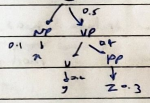
Statistical Parsing !

→ Probabilistic Context free grammar

→ Prob (sentence) = $\sum$ Prob (all parse tree)

→ Prob (Parse tree) = $\prod$ (all transition probs)

= 0.5 × 0.4 × 0.1 × 0.2 × 0.3



PCFG tasks :

i) Observation likelihood → Prob (sentence)

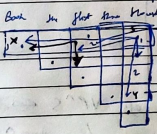
ii) Most likely derivation → of all parse tree, select one with max Prob (parse tree)

iii) Maximum likelihood training.

CNF : $A \to BCD$, $B \to E$, $C \to f$, $D \to g$ →

$A \to BX$ } two non-terminal
$X \to CD$ } prob = 1
$B \to e$
$C \to f$ } terminal
$D \to g$

Probabilistic CKY parser : bottom up parser



$S \to XY$
= Prob (S→XY) . Prob (X) . Prob (Y)

so together they represent the complete sentence

→ terminals eg: Noun : 0.1

Noun → Noun 0.1

Inside Algorithm :

for observation likelihood : Prob (sentence)

→ in CKY parser : combine probability of each cell (instead of max) → in case of most likely derivation

eg: cell S : represents the complete sentence so observation likelihood = $\sum$ Prob (of cell S)

Word Embedding

text → vector (vector space model)

— Non-semantic representation → each word → one hot vector

Problems → no semantics captured, high dimension of vec, sparse

— Semantic word representation :

i) — Co-occurrence matrix

I like deep learning
I like NLP

Window Size = 1 (one word before & after)

	I	like	NLP
I	0	2	0
like	2	0	1
NLP	0	1	0

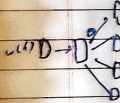
— symmetrical
— very high dimension, sparse

ii) Word2Vec :

→ Skip gram : predict context words given current word

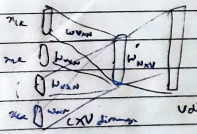
minimize $J = -\log P(w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m} \mid w_c)$

$= -\log \prod_{j=0,\, j \neq m}^{2m} P(w_{c-m+j} \mid w_c)$ (window size = m)



→ Continuous bag of words : predict current word based on other context words

minimize $J = -\log P(w_c \mid w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$



Word Embedding:

text → vector (vector space model)

→ Non-semantic representation → each word → one hot vector

Problems → no semantics captured, high dimension of vector, sparse

→ Semantic word representation:

i) → Co-occurrence matrix

I like deep learning

I like NLP

Window size = 1 (one word before & after)

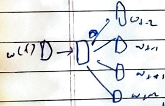
	I	like	NLP
I	0	2	0
like	2	0	1
NLP	0	1	0

→ symmetrical

→ very high dimension, sparse

ii) Word2vec:

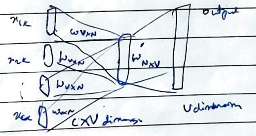
→ skip gram: predict context words given current word



minimize $J = -\log P(w_{c-m}, \ldots w_{c-1}, w_c, w_{c+1} \ldots w_{c+m} | w_c)$ ↳ loss fn (window size = m)

→ Continuous bag of words: predict current word based on other context words

minimize $J = -\log P(w_c | w_{c-m}, \ldots w_{c-1}, w_{c+1}, \ldots w_{c+m})$



→ Parse reranking : to select parse from N-best parse

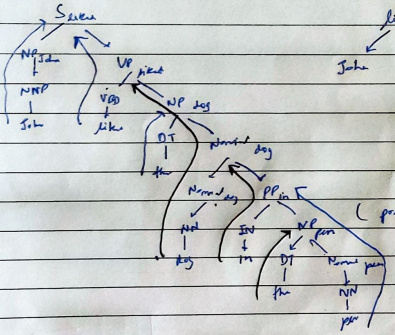
→ Human Parsing:

→ Garden path sentences : The old man the sea.
The complex house married students

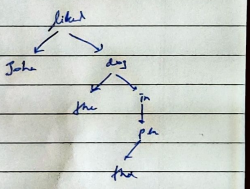
→ Center embedding → nested expressions

→ Dependency grammar:

(from parse tree)



Dependency graph



(preorder traversal : left, root, right)

Lexical Knowledge NN:

- Homonym: same sound / spelling but diff. meaning (eg bat, ball)
- Polysemy: single word, multiple related meanings. eg: bank.
- → Metonymy: one entity used to stand for another. eg: crown for king
- → Verb arguments frame: Verb { agent: _, direct obj: _, indirect obj: _ }

Wordnet: sense disambiguation | main instrument: relational semantics

- Synonymy / Antonymy
- Hypernymy / Hyponymy (superclass, subclass) (parent, many)
- Lexical entailment: V_1 logically entails V_2: If V1 then V2 or if !V_2 then !V_1
- Temporal inclusion: V1 temporally include verb V_2 Eg: If V2 then V1 must happen
- troponymy: verb in specific form of the more general form
  eg: V_1 = ~~walk~~ stroll / ~~walk~~ march / toddle (specific)
  V_2 = walk (broader concept)
- meronymy: one word represents a part or member of another word.
  eg: car (holonym, whole obj) , wheel (meronym, part of) | eg 2: head, body
- Gradation: series of gradual change
  eg: childhood, youth, old age