CS772: Deep Learning for Natural Language Processing (DL-NLP)

Course Summary

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

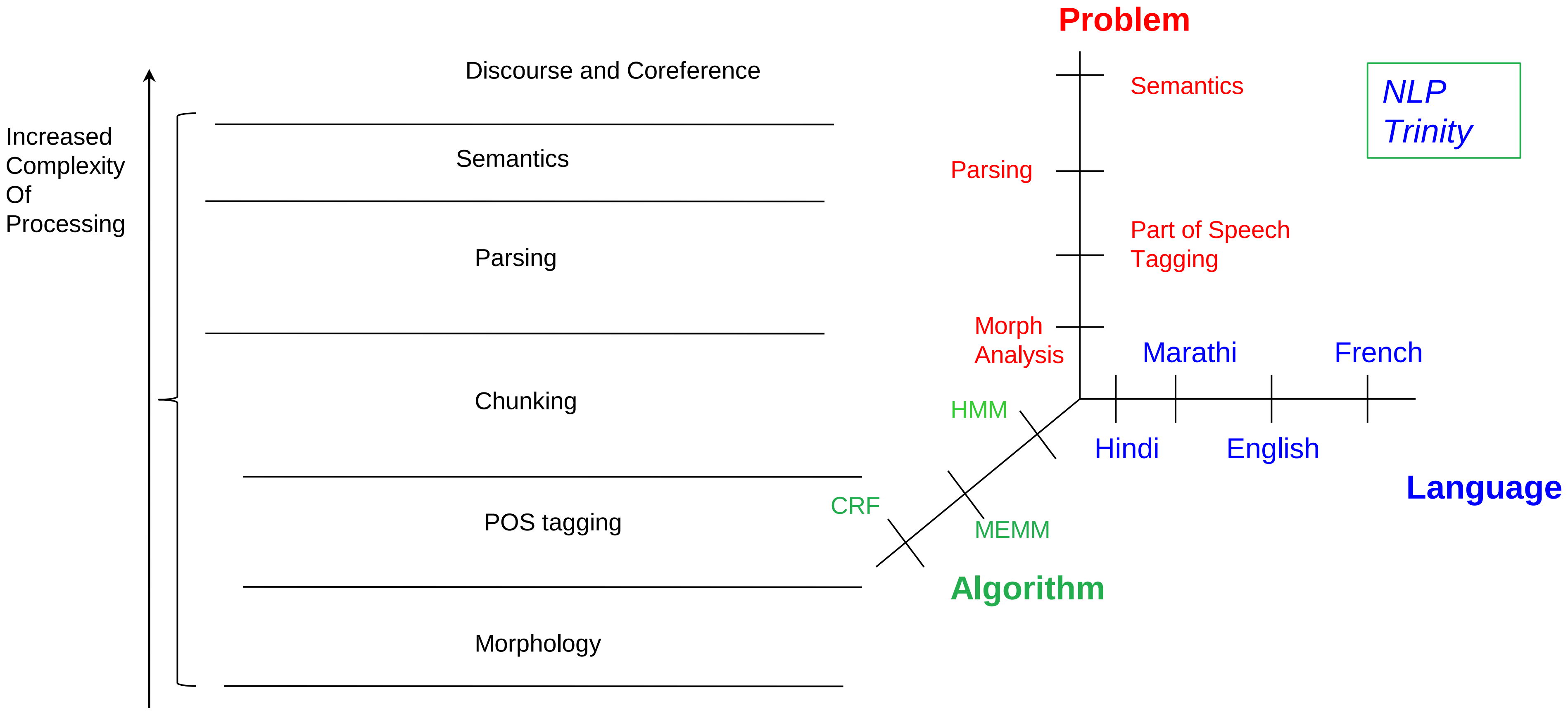
Week 14 of 10th April, 2023

Week1

Natural Language Processing

Art, science and technique of making computers understand and generate language

NLP is layered Processing, Multidimensional too



Main Challenge: **AMBIGUITY**

Example

- (from a TV serial) "*You met the boy; how did you find him?*"

Example (cntd.)

- "*You met the boy; how did you find him; did you like him?*"

Example (cntd.)

- "*You met the boy; how did you find him; through some reference?*"

Topics to be covered

- Single Neuron, perceptron and sigmoid; application to NLP; text classification
- Multilayered FFNN, Backpropagation; Softmax Application to NLP; Multiclass NLP problems
- Recurrent Neural Net (RNN); Application to NLP- seq2seq
- Recursive Neural Net; Application to NLP Parsing
- Convolutional Neural Nets; Multimodal NLP
- Transformers; Application to MT, QA, NLG

Major Topics covered in CS626, last sem

- NLP and Ambiguity
- POS Tagging
- Named Entity Recognition
- Word Sense Disambiguation
- Wordnet and Lexical Resources
- Alignment and EM Algorithm
- Machine Translation and MT Evaluation
- Conversational AI and Pragmatics

Evaluation Scheme (tentative)

- 40%: Reading, Thinking, Comprehending
  - Quizzes (20%) (4 nos.)
  - Endsem (20%)
- 60%: Doing things, Hands on
  - Assignments (20%)
  - Course Project (40%)

Quizzes and Endsem

- ONE/TWO subjective questions- only one page

- Rest MCQs on Moodle

Assignments and Project

- Continuous evaluation
- Meeting every two weeks to monitor progress
- Credit for thorough literature survey for the project work

Demoes

https://www.cfilt.iitb.ac.in/ssmt/speech2speech

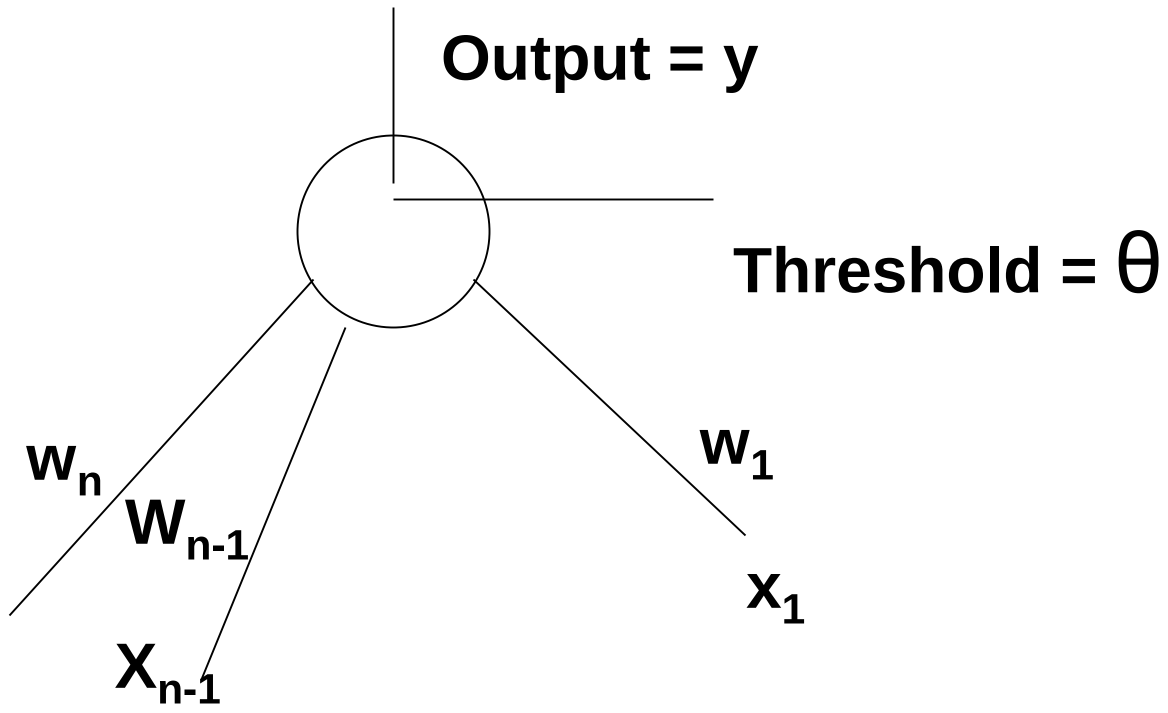
https://www.cfilt.iitb.ac.in/mtsystem/translate

https://chat.openai.com/chat#

Week2

The Perceptron Model

- A perceptron is a computing element with input lines having associated weights and the cell having a threshold value. The perceptron model is motivated by the biological neuron.



Statement of Convergence of PTA

- Statement:

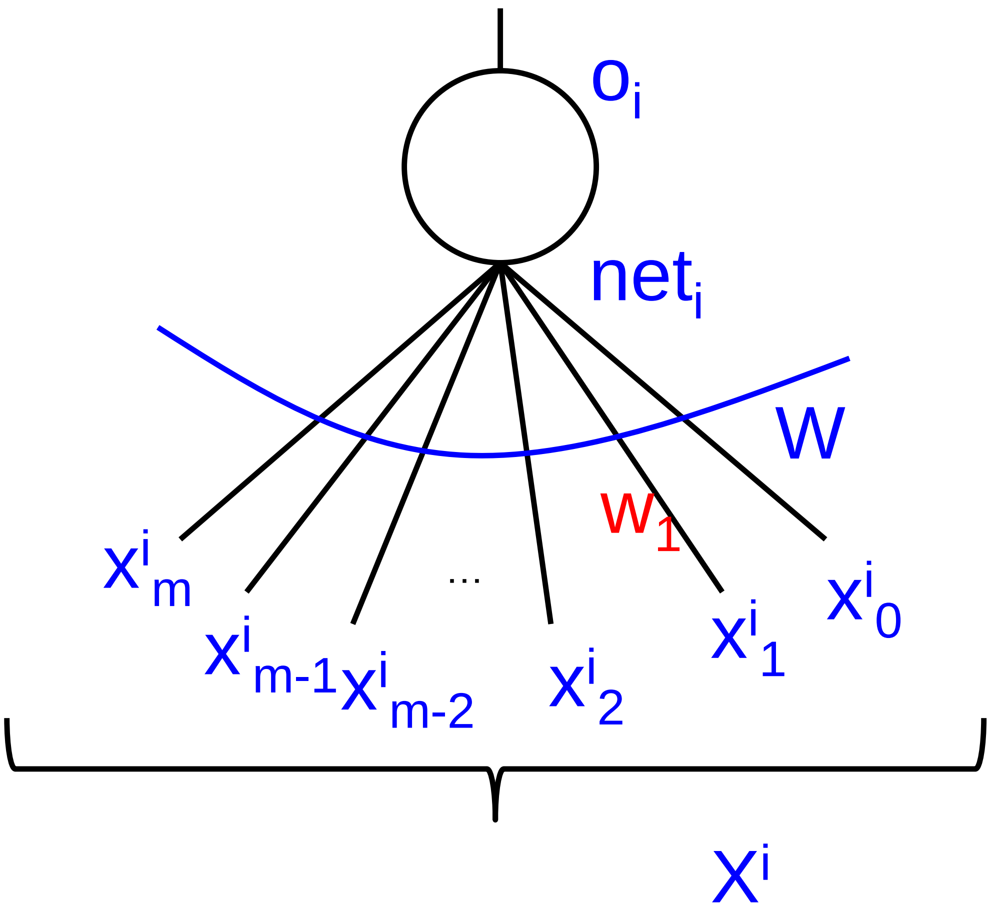
  Whatever be the initial choice of weights and whatever be the vector chosen for testing, PTA converges if the vectors are from a linearly separable function.

To note

- *F1:* $|G(W_n)|$ is bounded
- **IF**
- *F2*: n tends to infinity
- **THEN**
- *F3*: $|G(W_n)|$ is unbounded

Sigmoid

Sigmoid neuron



$$o^i = \frac{1}{1 + e^{-net^i}}$$

$$net_i = W.X^i = \sum_{j=0}^{m} w_j x_j^i$$

Sigmoid function: can saturate

- Brain saving itself from itself, in case of extreme agitation, emotion etc.

Definition: Sigmoid or Logit function

$$y = \frac{1}{1+e^{-x}}$$

$$y = \frac{1}{1+e^{-kx}}$$

$$\frac{dy}{dx} = y(1-y)$$

$$\frac{dy}{dx} = ky(1-y)$$

If k tends to infinity, sigmoid tends to the step function

Decision making under sigmoid

- Output of sigmod is between 0-1

- Look upon this value as probability of Class-1 (C_1)

- *1-sigmoid(x)* is the probability of Class-2 (C_2)
- Decide C_1, if $P(C_1) > P(C_2)$, else C_2

multiclass: SOFTMAX

- 2-class → multi-class (C classes)
- Sigmoid → softmax
- i^{th} input, c^{th} class (small c), c varies over classes
- In softmax, decide for that class which has the highest probability

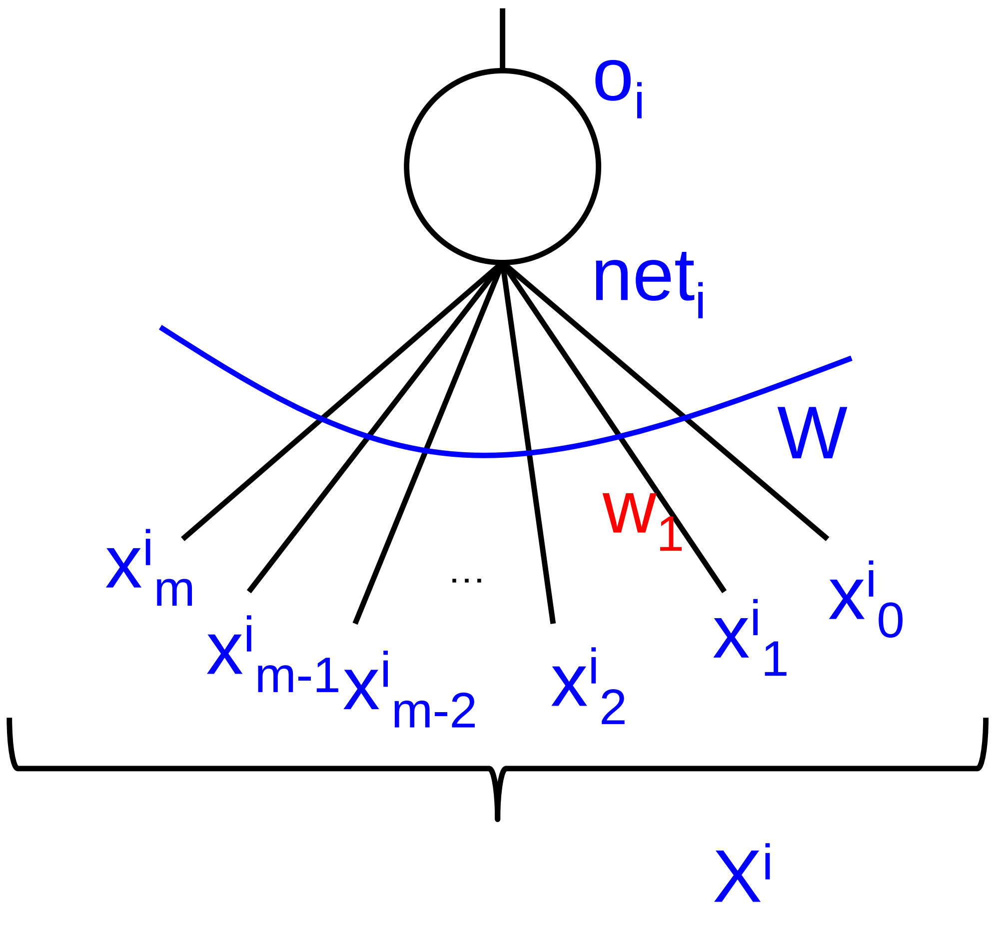
Mathematical form

$$\sigma(\bar{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}}$$

- σ is the **softmax** function
- Z is the input vector of size K
- The RHS gives the i^{th} component of the output vector
- Input to softmax and output of softmax are of the same dimension

Week3

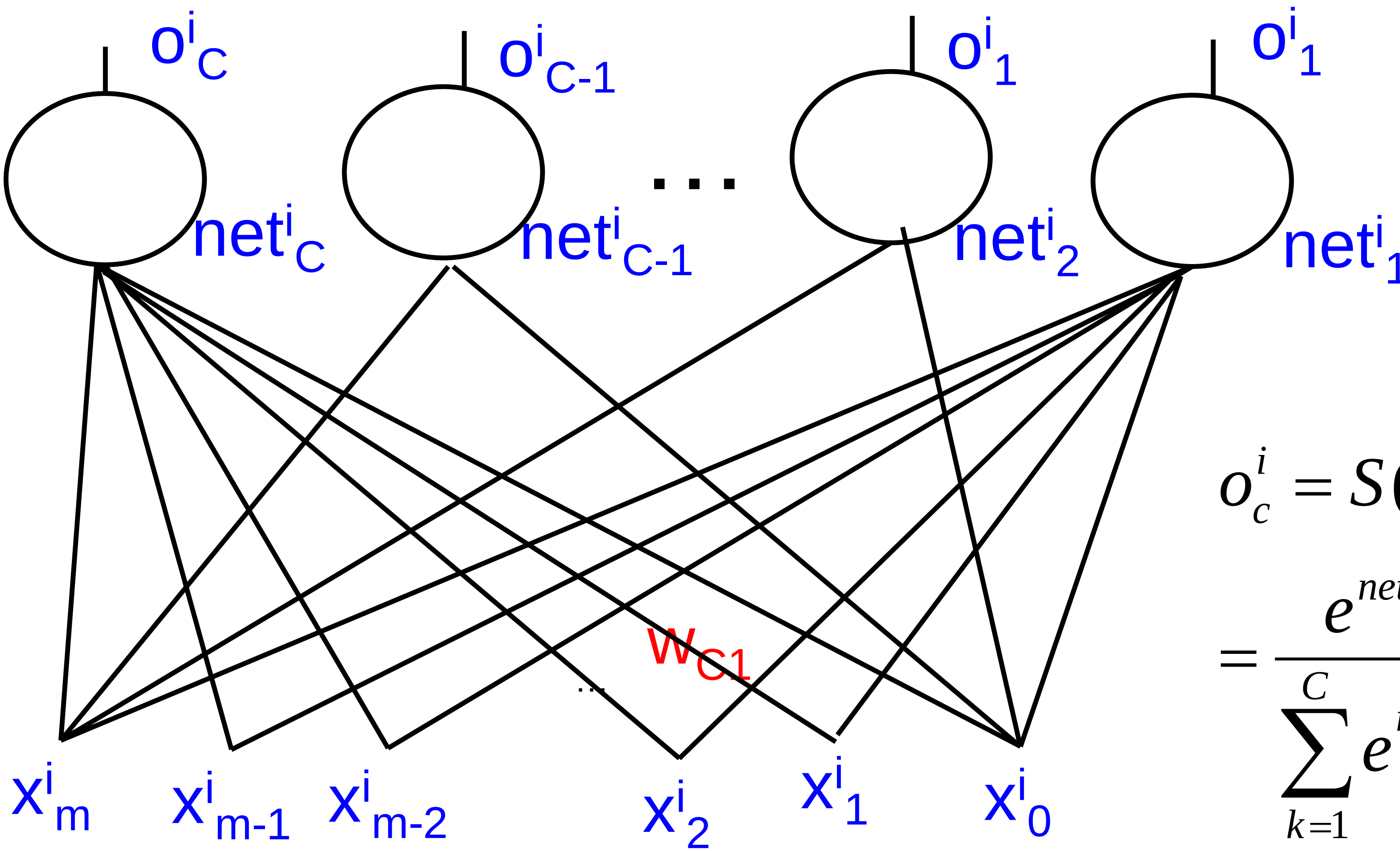
Sigmoid neuron



$$o^i = \frac{1}{1+e^{-net^i}}$$

$$net_i = W.X^i = \sum_{j=0}^{m} w_j x_j^i$$

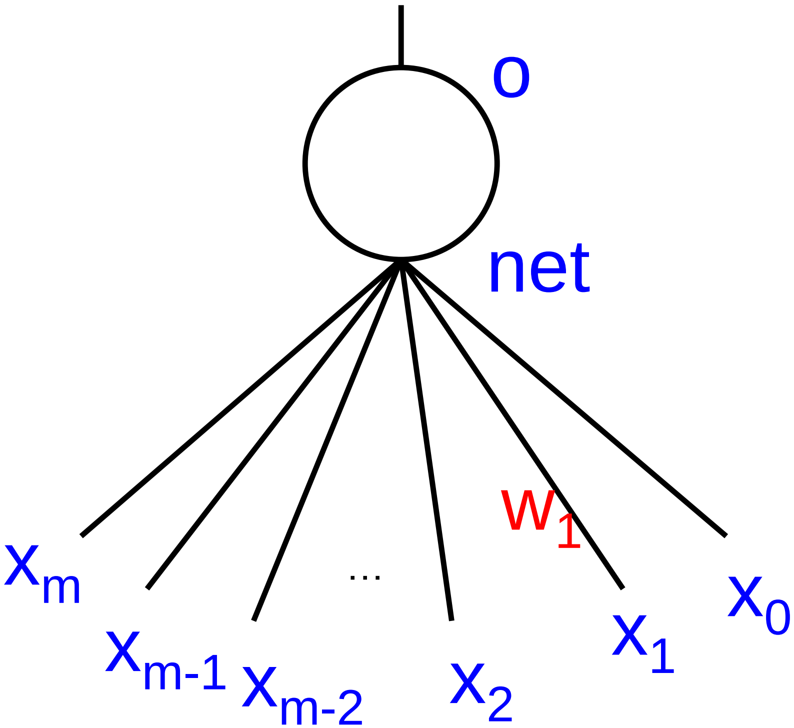
Softmax Neuron



$$o_c^i = S(NET^i)_c = \frac{e^{net_c^i}}{\sum_{k=1}^{C} e^{net_k^i}}$$

Output for class c (small c), c:1 to C

Single sigmoid neuron- weight change rule



$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

$$E = -t \log o - (1-t)\log(1-o)$$

$$\Rightarrow \frac{\partial E}{\partial o} = -\frac{t}{o} + \frac{1-t}{1-o} = -\frac{t-o}{o(1-o)}$$

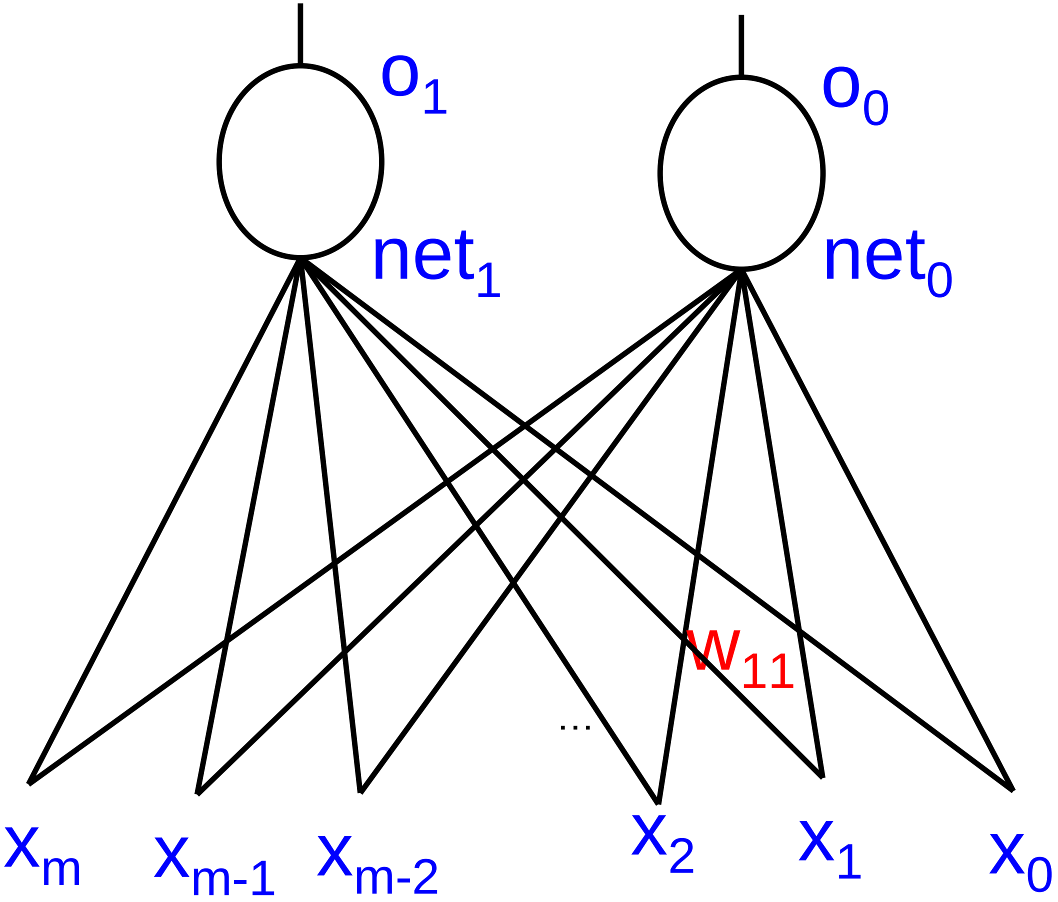
$$o = \frac{1}{1+e^{-net}} \ (sigmoid) \Rightarrow \frac{\partial o}{\partial net} = o(1-o)$$

$$net = \sum_{j=0}^{m} w_j x_j \ \Rightarrow \frac{\partial net}{\partial w_1} = x_1$$

$$\Rightarrow \Delta w_1 = \eta \frac{\partial E}{\partial w_1} = \eta(t-o)x_1$$

$$\Delta w_1 = \eta(t\text{-}o)x_1$$

Multiple neurons in the output layer: softmax+*cross entropy* loss (1/2): illustrated with 2 neurons and single training data point



$$O = < o_1, o_0 >$$

$$NET = < net_1, net_0 >$$

$$o_1 = \frac{e^{net_1}}{e^{net_1} + e^{net_0}}, \; o_0 = \frac{e^{net_0}}{e^{net_1} + e^{net_0}}$$

$$\frac{\partial O}{\partial NET} = \begin{bmatrix} \frac{\partial o_0}{\partial net_0} & \frac{\partial o_1}{\partial net_0} \\ \frac{\partial o_0}{\partial net_1} & \frac{\partial o_1}{\partial net_1} \end{bmatrix}$$

$$= \begin{bmatrix} o_0(1-o_0) & -o_0 o_1 \\ -o_1 o_0 & o_1(1-o_1) \end{bmatrix}$$

Softmax and Cross Entropy (2/2)

$$E = -t_1 \log o_1 - t_0 \log o_0$$

$$o_1 = \frac{e^{net_1}}{e^{net_1} + e^{net_0}}, \; o_0 = \frac{e^{net_0}}{e^{net_1} + e^{net_0}}$$

$$\frac{\partial E}{\partial w_{11}} = -\frac{t_1}{o_1}\frac{\partial o_1}{\partial w_{11}} - -\frac{t_0}{o_0}\frac{\partial o_0}{\partial w_{11}}$$

$$\frac{\partial o_1}{\partial w_{11}} = \frac{\partial o_1}{\partial net_1} . \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_1}{\partial net_0} . \frac{\partial net_0}{\partial w_{11}} = o_1(1 - o_1)x_1 + 0$$

$$\frac{\partial o_0}{\partial w_{11}} = \frac{\partial o_0}{\partial net_1} . \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_0}{\partial net_0} . \frac{\partial net_0}{\partial w_{11}} = -o_1 o_0 x_1 + 0$$

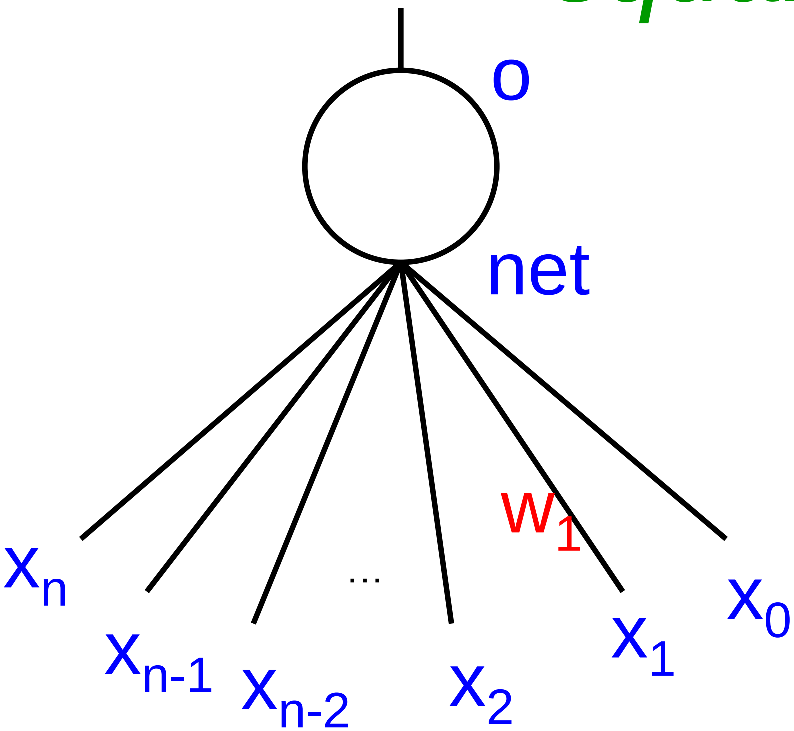
$$\Rightarrow \frac{\partial E}{\partial w_{11}} = -t_1(1 - o_1)x_1 + t_0 o_1 x_1 = -t_1(1 - o_1)x_1 + (1 - t_1)o_1 x_1$$

$$= [-t_1 + t_1 o_1 + o_1 - t_1 o_1]x_1 = -(t_1 - o_1)x_1$$

$$\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}} = \eta(t_1 - o_1)x_1$$

Weight change rule with TSS

Single neuron: *sigmoid+total sum square* (tss) loss



Lets consider wlg w_1. Change is weight $\Delta w_1 = -\eta \delta L / \delta w_1$

η= learning rate,

L=loss= $\frac{1}{2}(t-o)^2$,

t=target, o=observed output

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

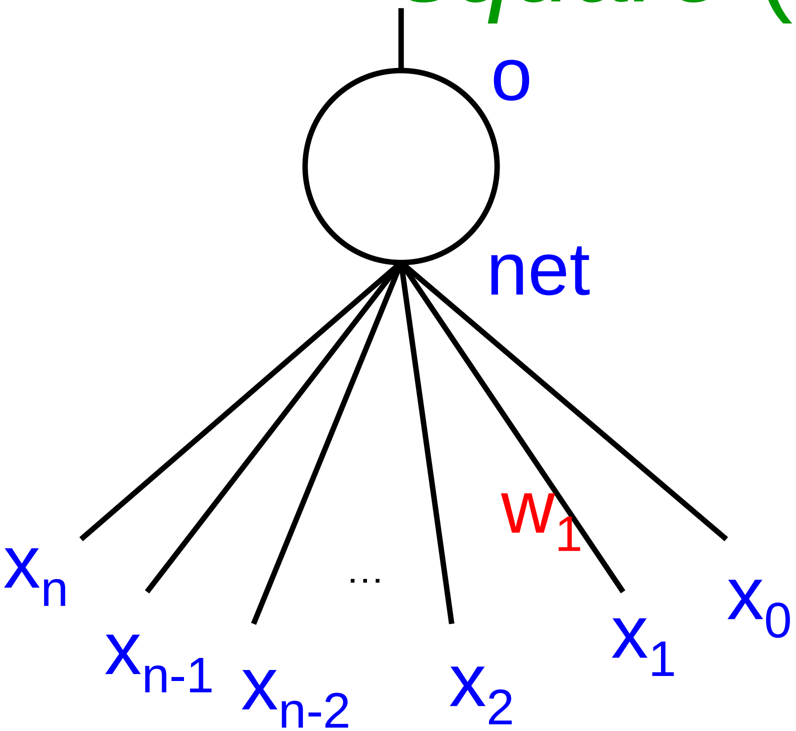
$$L = \frac{1}{2}(t-o)^2 \;\Rightarrow\; \frac{\partial L}{\partial o} = -(t-o) \quad (1)$$

$$o = \frac{1}{1+e^{-net}} \,(sigmoid) \Rightarrow \frac{\partial o}{\partial net} = o(1-o) \;(2)$$

$$net = \sum_{i=0}^{n} w_i x_i \;\Rightarrow\; \frac{\partial net}{\partial w_1} = x_1 \quad (3)$$

$$\Rightarrow \Delta w_1 = \eta(t-o)o(1-o)x_1$$

Single neuron: *sigmoid+total sum square* (tss) loss (cntd)



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

$$L = \frac{1}{2}(t-o)^2 \quad \Rightarrow \frac{\partial L}{\partial o} = (t-o) \;\; (1)$$

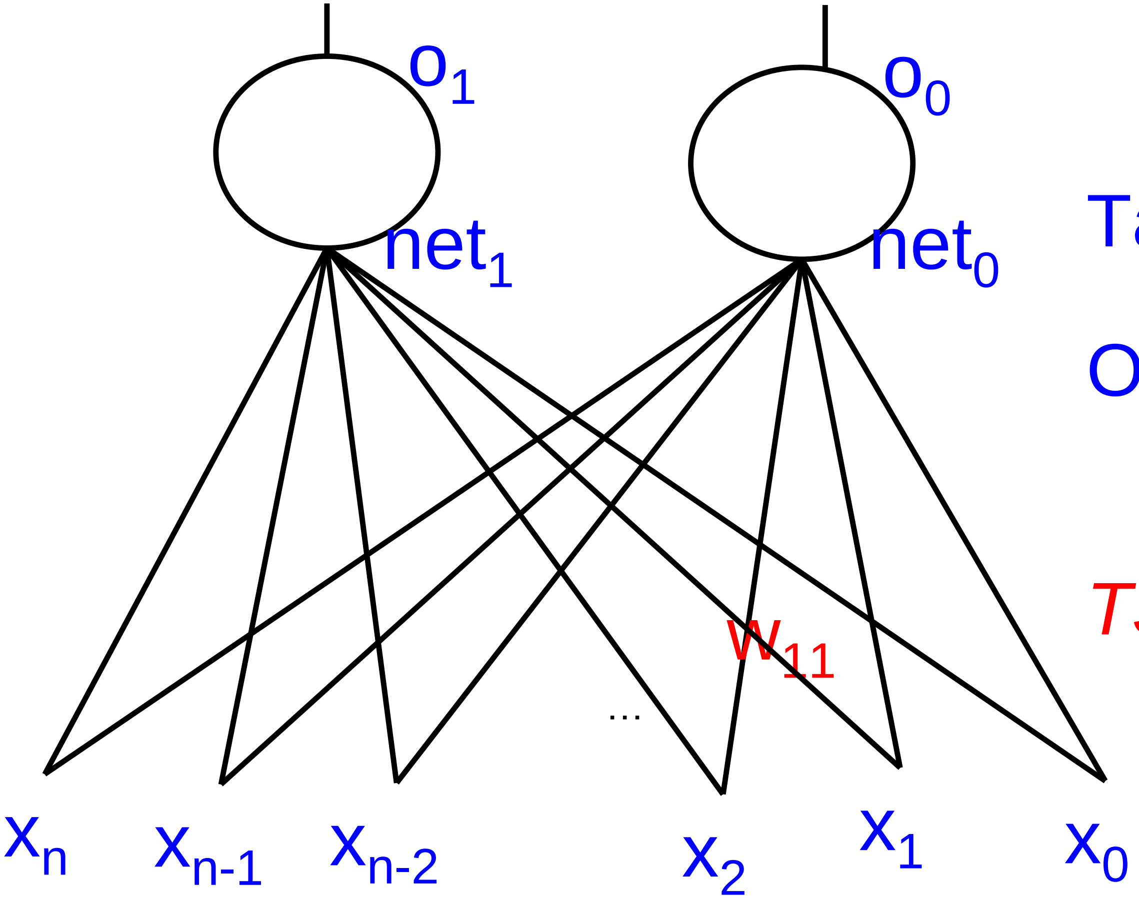
$$o = \frac{1}{1+e^{-net}} \; (sigmoid) \Rightarrow \frac{\partial o}{\partial net} = o(1-o) \; (2)$$

$$net = \sum_{i=0}^{n} w_i x_i \quad \Rightarrow \frac{\partial net}{\partial w_1} = x_i \;\; (3)$$

$$\Rightarrow \Delta w_1 = \eta(t-o)o(1-o)x_i$$

$$\Delta w_1 = \eta(t\text{-}o)o(1\text{-}o)x_1$$

Multiple neurons in the output layer: *sigmoid+total sum square* (tss) loss



Target vector: $<t_1, t_0>$

Observed vector: $<o_1, o_0>$

TSS Loss$= \frac{1}{2}[(t_1-o_1)^2+(t_0-o_0)^2]$

$$\Delta w_{11} = \eta(t_1-o_1)o_1(1-o_1)x_1$$

General Backpropagation Rule

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \text{ for hidden layers}$$

Week4

Deriving the word vector: setting

$$W^s : w_0^s, w_1^s, w_2^s, \ldots w_i^s, \ldots w_m^s$$

W^S: word sequence in the s^{th} Sentence

$$V_{w_i} : [v_0^i, v_1^i, v_2^i, \ldots v_k^i, \ldots v_d^i]$$

V_{wi}: word vector of w_i

$$J = P(w_j \mid w_i)$$

$$L = -P(w_j \mid w_i)$$

$$P(w_j \mid w_i) = \frac{e^{V_{w_i} . V_{w_j}}}{\sum_{j'=1}^{|V|} e^{V_{w_i} . V_{w_{j'}}}}$$

$$LL = -V_{w_i} . V_{w_j} + \ln\left(\sum_{j'=1}^{|V|} e^{V_{w_i} . V_{w_{j'}}} \right)$$

Deriving the word vector: Optimization (1/2)

$$V_{w_i} : [v_0^i, v_1^i, v_2^i, \ldots v_k^i, \ldots v_d^i] = [u_0, u_1, u_2, \ldots u_k, \ldots u_d]$$

$$V_{w_j} : [v_0^j, v_1^j, v_2^j, \ldots v_k^j, \ldots v_d^j] = [v_0, v_1, v_2, \ldots v_k, \ldots v_d]$$

$$V_{w_{j'}} : [v'_0, v'_1, v'_2, \ldots v'_k, \ldots v'_d]$$

$$V_{w_i} . V_{w_j} = \sum_{k=0}^{d} u_k v_k$$

$$\frac{\partial LL}{\partial u_k} = -v_k + \frac{\frac{\partial}{\partial u_k}\left(\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}} \right)}{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}}}$$

Deriving the word vector: Optimization

$$= -v_k + \frac{\sum_{j'=1}^{|V|} \frac{\partial}{\partial u_k}\left(e^{\sum_{k=0}^{d} u_k v_k^{'}} \right)}{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}}} = -v_k + \frac{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}} \frac{\partial}{\partial u_k}\left(\sum_{k=0}^{d} u_k v_k^{'} \right)}{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}}}$$

$$= -v_k + \frac{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}} v_k^{'}}{\sum_{j'=1}^{|V|} e^{\sum_{k=0}^{d} u_k v_k^{'}}} = -v_k + \sum_{j'=1}^{|V|} P(w_{j'} \mid w_i).v_{k'} = -v_k + E(v_{k'})$$

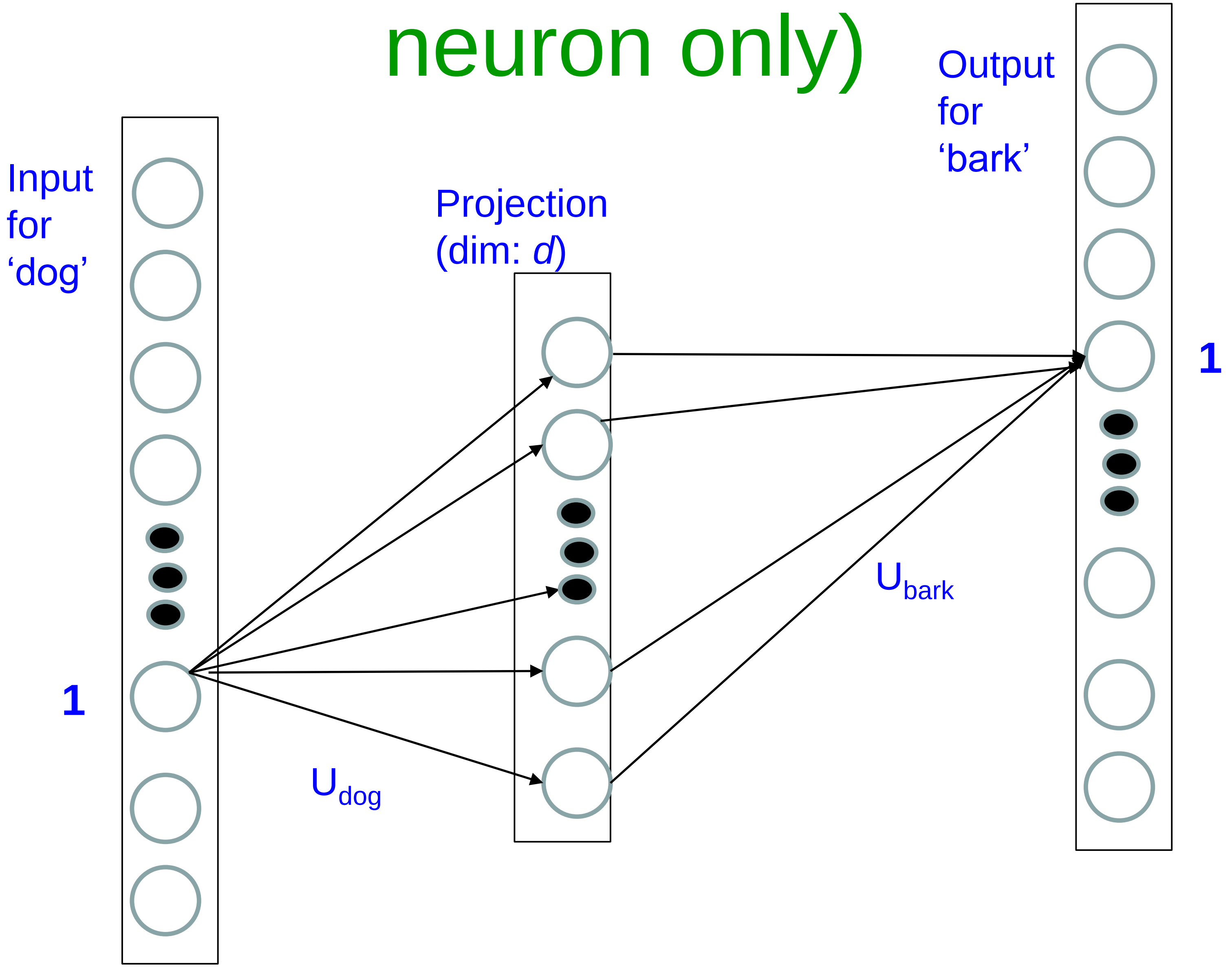
Deriving the word vector, Gradient Descent: Δu_k

$$\Delta u_k = -\eta \frac{\partial LL}{\partial u_k} = \eta[v_k - E(v_{k'})]$$

Example

- We want, say, *P('bark'|'dog')*
- Take the weight vector **FROM** 'dog' neuron **TO** projection layer (call this U_{dog})
- Take the weight vector **TO** 'bark' neuron **FROM** projection layer (call this U_{bark})
- When initialized, U_{dog} and U_{bark} give the initial estimates of word vectors of 'dog' and 'bark'
- The weights and therefore the word vectors get fixed by back propagation

Input to Projection (shown for one neuron only)



Modelling *P(context word|input word)* *(2/2)*

- To model the probability, first compute dot product of u_{dog} and v_{bark}
- Exponentiate the dot product
- Take softmax over all dot products over the whole vocabulary

$$P('bark'|'dog') = \frac{\exp(U_{dog}^{T} U_{bark})}{\sum_{R \varepsilon Vocabulary} \exp(U_{dog}^{T} U_{R})}$$

P('bark'|'dog') (1/2)

$$P('bark'|'dog') = \frac{\exp(U_{dog}^{T} U_{bark})}{\sum_{R \varepsilon Vocabulary} \exp(U_{dog}^{T} U_{R})}$$

$$\log(P('bark'|'dog')) = U_{dog}^{T} U_{bark} - \log(\sum_{R \varepsilon Vocabulary} \exp(U_{dog}^{T} U_{R}))$$

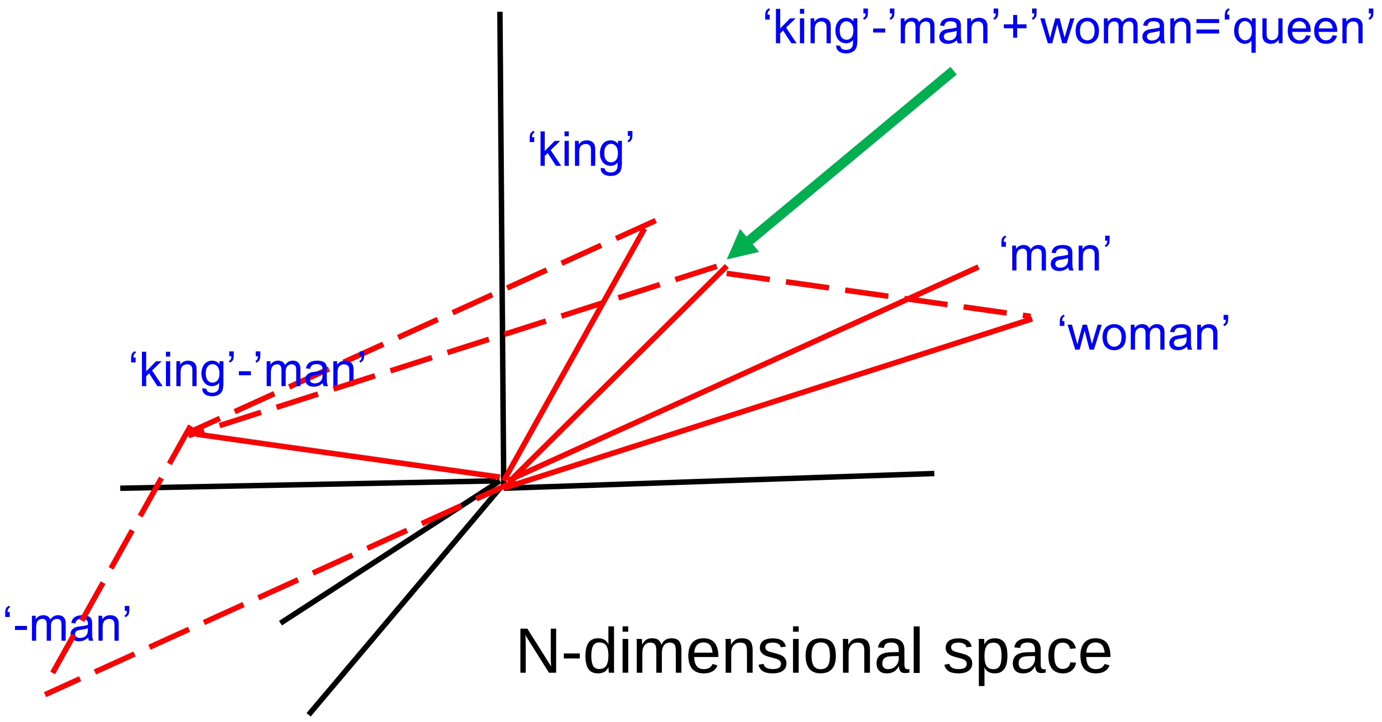
Word2vec architectures

Mikolov 2013

Classic work

- Caught the attention of the world by equations like

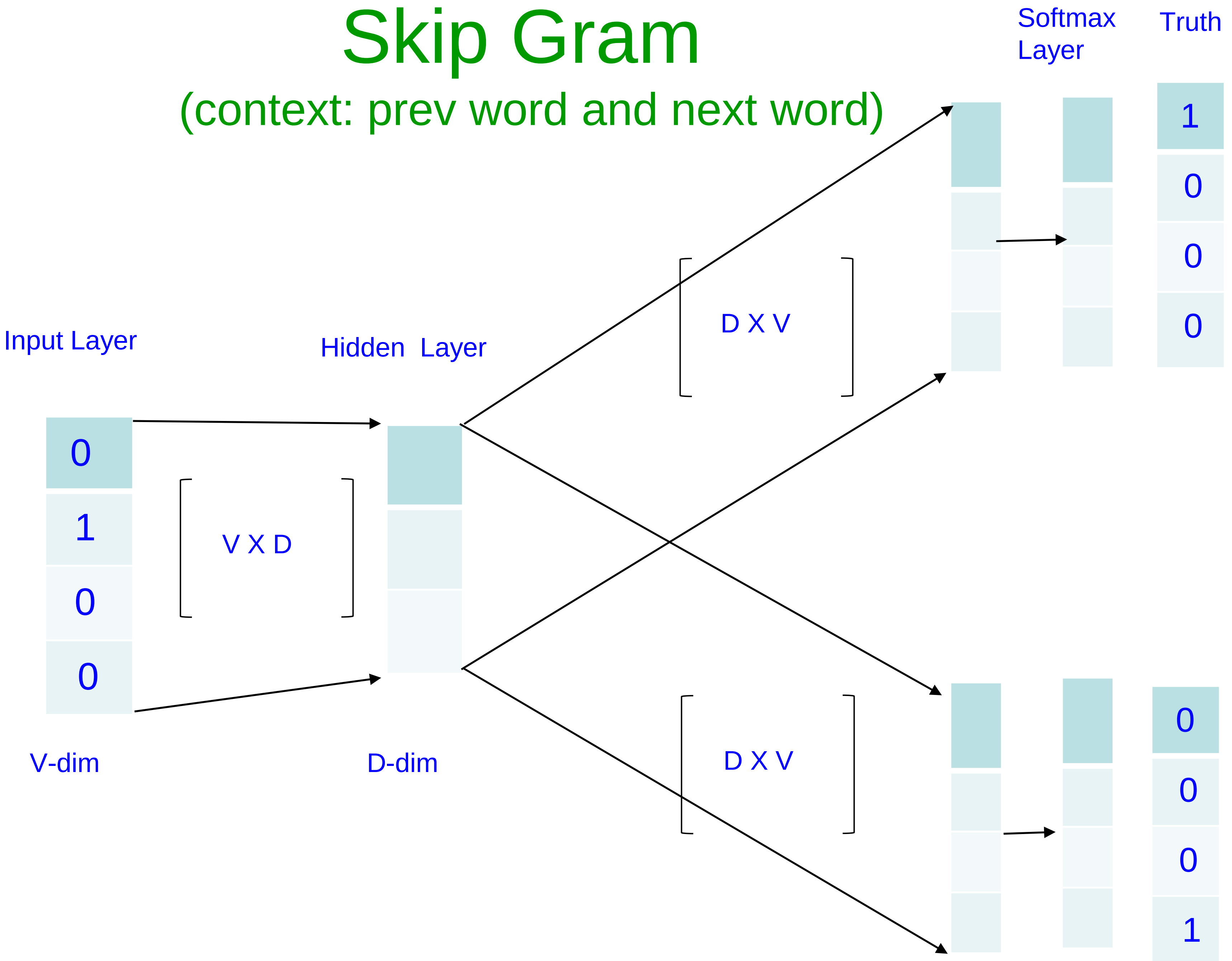
'king'-'man'+'woman'='queen'

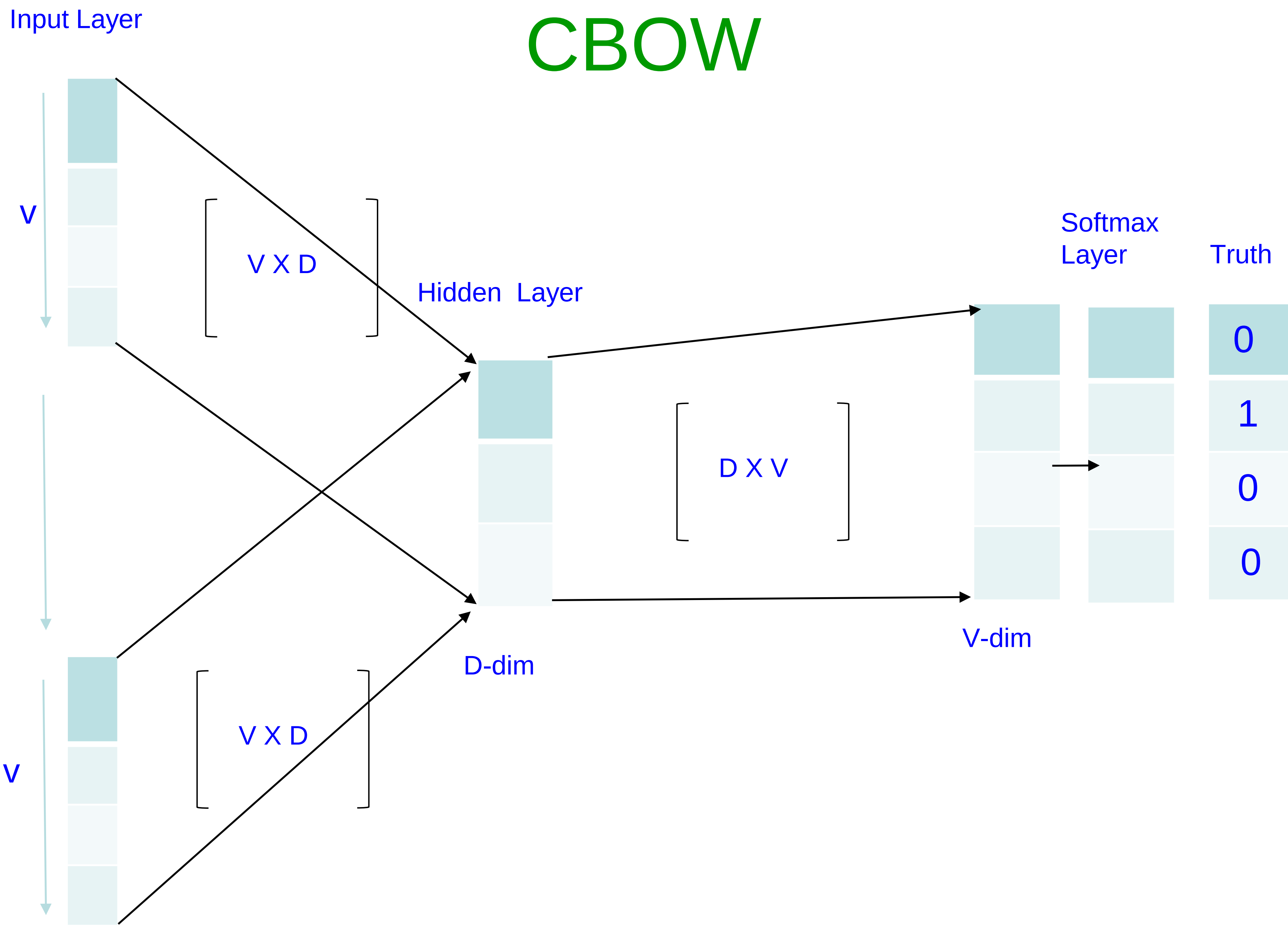


N-dimensional space

Skip Gram

(context: prev word and next word)





CBOW
Input Layer
V
V X D
Hidden Layer
D X V
Softmax
Layer
Truth
0
1
0
0
V-dim
D-dim
V X D
V

Symbolic approach to representing word meaning

Syntagmatic and Paradigmatic Relations

- Syntagmatic and paradigmatic relations
  - Lexico-semantic relations: synonymy, antonymy, hypernymy, mernymy, troponymy etc. **CAT is-a ANIMAL**
  - Coccurence: **CATS MEW**
- Resources to capture semantics:
  - Wordnet: primarily paradigmatic relations
  - ConceptNet: primarily Syntagmatic Relations

Fundamental Device- Lexical Matrix (with examples)

Word Meanings	**Word Forms**				
	F_1	F_2	F_3	…	F_n
M_1	(*depend*) $E_{1,1}$	(*bank*) $E_{1,2}$	(rely) $E_{1,3}$		
M_2		(*bank*) $E_{2,2}$		(*embankment*) $E_{2,\ldots}$	
M_3		(*bank*) $E_{3,2}$	$E_{3,3}$		
…				…	
M_m					$E_{m,n}$

Week5

Two main models for learning word vectors

- 1) global matrix factorization methods, such as latent semantic analysis (LSA) (Deerwester et al., 1990) and
- 2) local context window methods, such as the skip-gram model of Mikolov et al. (2013)
- Currently, both families suffer significant drawbacks.

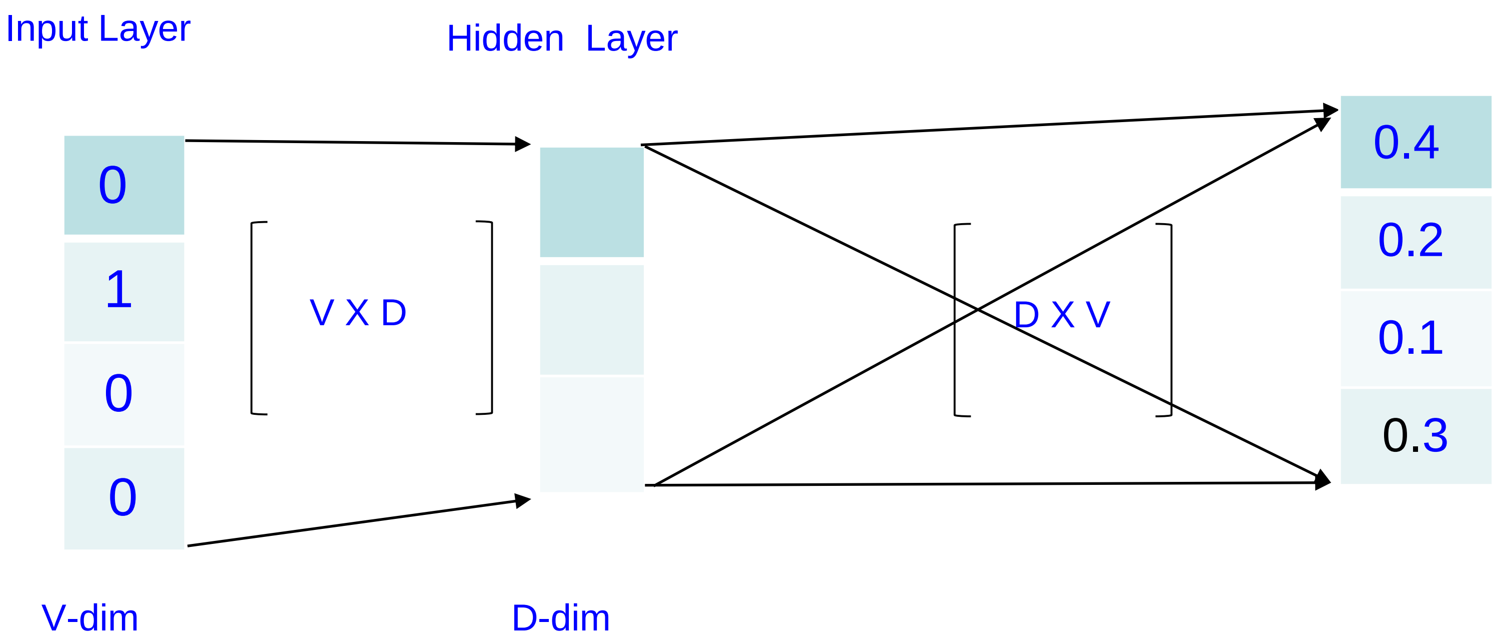
Matrix Factorization: drawback

- “most frequent words contribute a disproportionate amount to the similarity measure: the number of times two words co-occur with *the* or *and*, for example, will have a large effect on their similarity despite conveying relatively little about their semantic relatedness.”

Skip Gram & CBOW: drawback

- “shallow window-based methods suffer from the disadvantage that they do not operate directly on the co-occurrence statistics of the corpus. Instead,these models scan context windows across the entire corpus, which fails to take advantage of the vast amount of repetition in the data”

Can this architecture for Glove work?



Representation using syntagmatic relations: Co-occurrence Matrix

Corpora: I enjoy cricket. I like music. I like deep learning

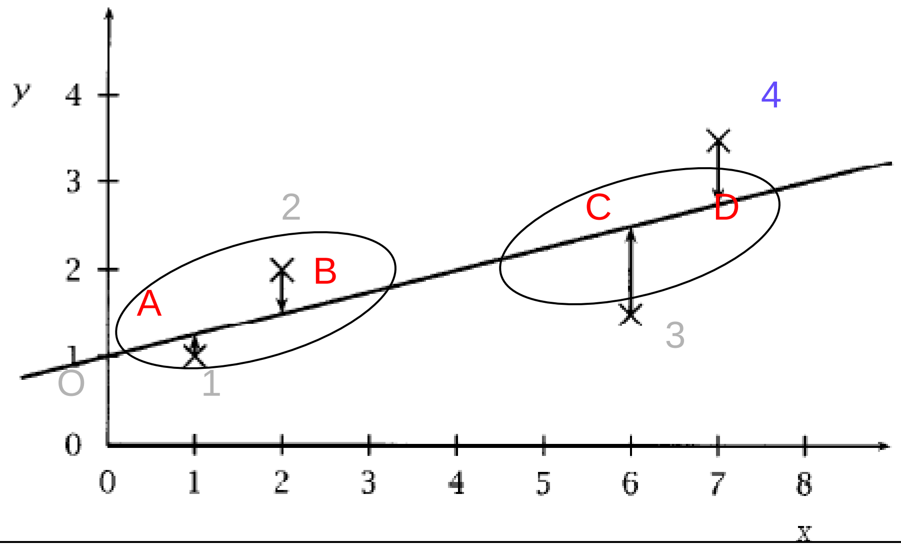
	I	**enjoy**	**cricket**	**like**	**music**	**deep**	**learning**
I	-	1	1	2	1	1	1
enjoy	1	-	1	0	0	0	0
cricket	1	1	-	0	0	0	0
like	2	0	0	-	1	1	1
music	1	0	0	1	-	0	0
deep	1	0	0	1	0	-	1
learning	1	0	0	1	0	1	-

Solution: uses co-occurences

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Dimensionality Reduction by PCA

Intuition for Dimensionality Reduction



- 1, 2, 3, 4: are the points
- A, B, C, D: are their projections on the fitted line by linear regression

- Suppose 1, 2 form a class and 3, 4 another class
- Of course, it is easy to set up a hyper plane that will separate 1 and 2 from 3 and 4
- That will be classification in **2 dimension**

- But suppose we form another attribute of these points, viz., distances of their
- projections On the line from “O”
- Then the points can be classified by a threshold on these distances
- This effectively is classification in the **reduced dimension (1 dimension)**

Principal Component Analysis

Example: *IRIS Data (only 3 values out of 150)*

ID	Petal Length (a_1)	Petal Width (a_2)	Sepal Length (a_3)	Sepal Width (a_4)	Classification
001	5.1	3.5	1.4	0.2	Iris-setosa
051	7.0	3.2	4.7	1.4	Iris-versicolor
101	6.3	3.3	6.0	2.5	Iris-virginica

Training and Testing Data

- Training: 80% of the data; 40 from each class: total 120
- Testing: Remaining 30
- Do we have to consider all the 4 attributes for classification?
- Less attributes is likely to increase the generalization performance (Occam Razor Hypothesis: *A simpler hypothesis generalizes better*)

The multivariate data: *n* instances, *p* attributes

$$
\begin{array}{cccccccc}
\mathbf{X_1} & \mathbf{X_2} & \mathbf{X_3} & \mathbf{X_4} & \mathbf{X_5} & \cdots & \mathbf{X_p} \\
x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & \cdots & x_{1p} \\
x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & \cdots & x_{2p} \\
x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & \cdots & x_{3p} \\
x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & \cdots & x_{4p} \\
 & & & \cdots & & & \\
 & & & \cdots & & & \\
x_{n1} & x_{n2} & x_{n3} & x_{n4} & x_{n5} & \cdots & x_{np}
\end{array}
$$

Week6

PCA: Example

49 birds: 21 survived in a storm and 28 died.

5 body characteristics given

X_1: body length; X_2: alar extent; X_3: beak and head length

X_4: humerus length; X_5: keel length

Could we have predicted the fate from the body characteristic

$$
\begin{array}{cc}
 & \begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \end{array} \\
R = & \begin{bmatrix} 1.000 & & & & \\ 0.735 & 1.000 & & & \\ 0.662 & 0.674 & 1.000 & & \\ 0.645 & 0.769 & 0.763 & 1.000 & \\ 0.605 & 0.529 & 0.526 & 0.607 & 1.000 \end{bmatrix} \begin{array}{c} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{array}
\end{array}
$$

Eigenvalues and Eigenvectors of *R*

Eigenvalues: 3.612, 0.532, 0.386, 0.302, 0.165

First Eigen-vector: V_1	**V_2**	**V_3**	**V_4**	**V_5**
0.452	**0.462**	**0.451**	**0.471**	**0.398**
-0.051	**0.300**	**0.325**	**0.185**	**-0.877**
0.691	**0.341**	**-0.455**	**-0.411**	**-0.179**
-0.420	**0.548**	**-0.606**	**0.388**	**0.069**
0.374	**-0.530**	**-0.343**	**0.652**	**-0.192**

Which principal components are important?

- Total variance in the data=
  $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5$
  = sum of diagonals of $R = 5$
- First eigenvalue= 3.616 ≈ 72% of total variance 5
- Second ≈ 10.6%, Third ≈ 7.7%, Fourth ≈ 6.0% and Fifth ≈ 3.3%
- ***First PC is the most important and sufficient for studying the classification***

Forming the PCs

- $Z_1 = 0.451X_1 + 0.462X_2 + 0.451X_3 + 0.471X_4 + 0.398X_5$
- $Z_2 = -0.051X_1 + 0.300X_2 + 0.325X_3 + 0.185X_4 - 0.877X_5$
- For all the 49 birds find the first two principal components
- This becomes the new data
- Classify using them

For the first bird

X_1=156, X_2=245, X_3=31.6, X_4=18.5, X_5=20.5

After standardizing

Y_1=(156-157.98)/3.65=-0.54,

Y_2=(245-241.33)/5.1=0.73,

Y_3=(31.6-31.5)/0.8=0.17,

Y_4=(18.5-18.46)/0.56=0.05,

Y_5=(20.5-20.8)/0.99=-0.33

PC_1 for the first bird=

Z_1= 0.45X(-0.54)+ 0.46X(0.725)+0.45X(0.17)+0.47X(0.05)+0.39X(-0.33)

=0.064

Similarly, Z_2= 0.602

Reduced Classification Data

- Instead of

X_1	X_2	X_3	X_4	X_5
		49 rows		

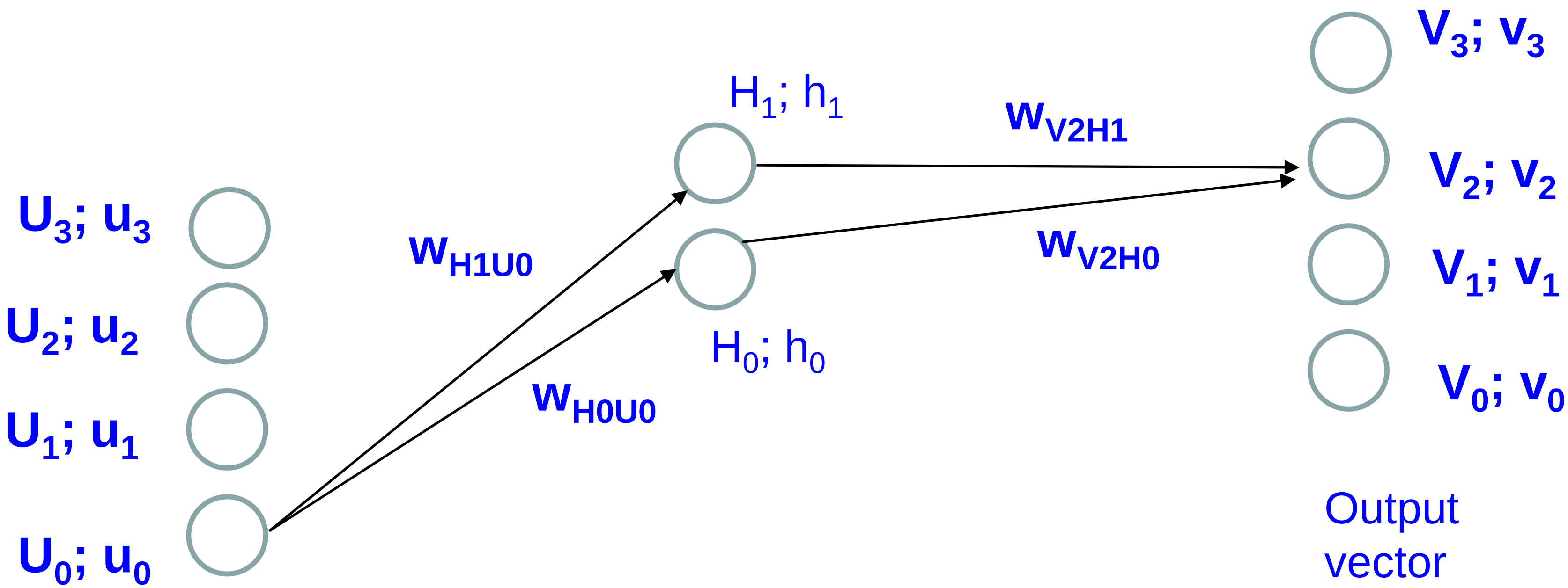
- Use

Z_1	Z_2
49	rows

Working out a simple case of word2vec

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Output vector V

Input vector U

Weights go from all neurons to all neurons in the next layer; shown For only one input and output

Computing Δw_{V2H0}

$$\Delta w_{V_2H_0} = -\eta \frac{\delta E}{\delta w_{V_2H_0}}$$

$$E = -net_{V_2} + \log(e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}})$$

$$= -W_{U_0}W_{V_2}^T + \log(e^{net_{V_0}} + e^{net_{V_1}} + e^{net_{V_2}} + e^{net_{V_3}})$$

$$W_{U_0}W_{V_2}^T = w_{V_2H_0}w_{H_0U_0} + w_{V_2H_1}w_{H_1U_0}$$

$$\frac{\delta E}{\delta w_{V_2H_0}} = -w_{H_0U_0} + \frac{e^{W_{V_2}.W_{U_0}}}{e^{W_{V_0}.W_{U_0}} + e^{W_{V_1}.W_{U_0}} + e^{W_{V_2}.W_{U_0}} + e^{W_{V_3}.W_{U_0}})}.w_{H_0U_0}$$

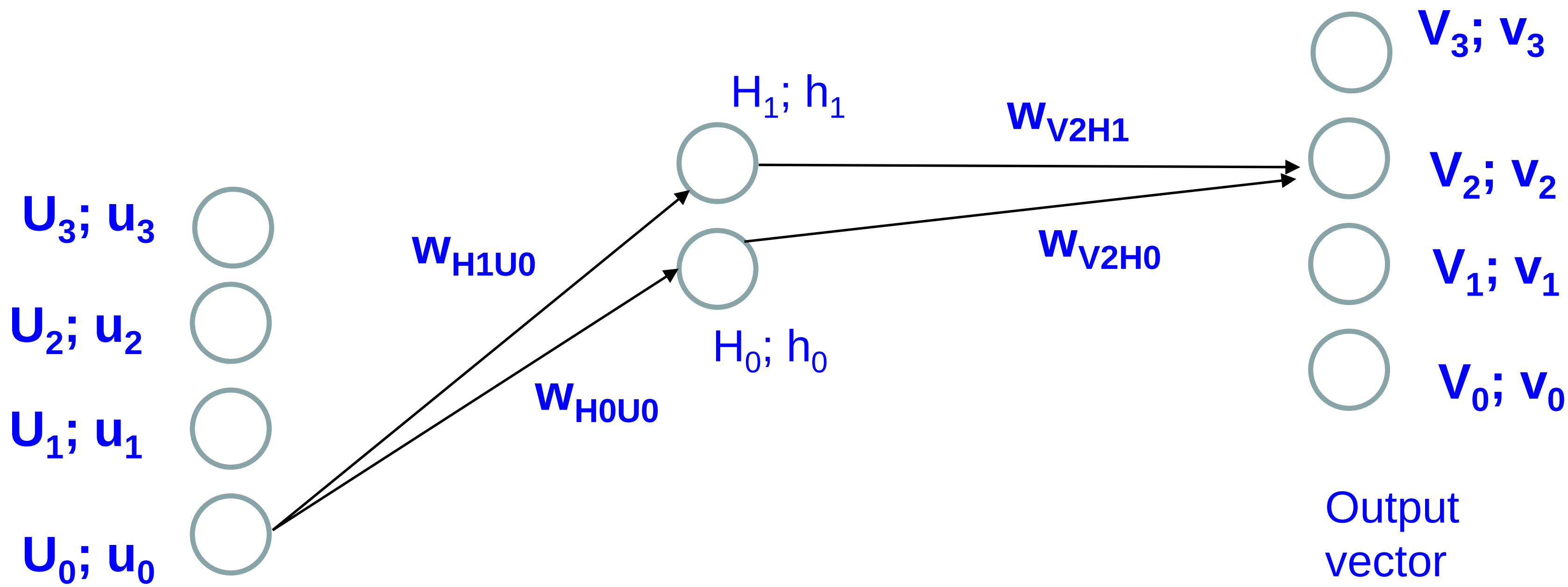
$$= -w_{H_0U_0} + v_2.w_{H_0U_0}$$

$$\Rightarrow \Delta w_{V_2H_0} = \eta(1 - v_2).w_{H_0U_0} = \eta(1 - v_2)o_{H_0}$$

o/p of hidden neuron H_0

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Output vector V

Input vector U

Weights go from all neurons to all neurons in the next layer; shown For only one input and output

Change in other weights to output layer, say, V_1, due to input U_0

$$\Delta w_{V_1H_0} = -\eta \frac{\delta E}{\delta w_{V_1H_0}}$$

$$E = -net_{V_2} + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$= -W_{U_0}W_{V_2}^T + \log(e^{net_{V0}} + e^{net_{V1}} + e^{net_{V2}} + e^{net_{V3}})$$

$$W_{U_0}W_{V_2}^T = w_{V_2H_0}w_{H_0U_0} + w_{V_2H_1}w_{H_1U_0}$$

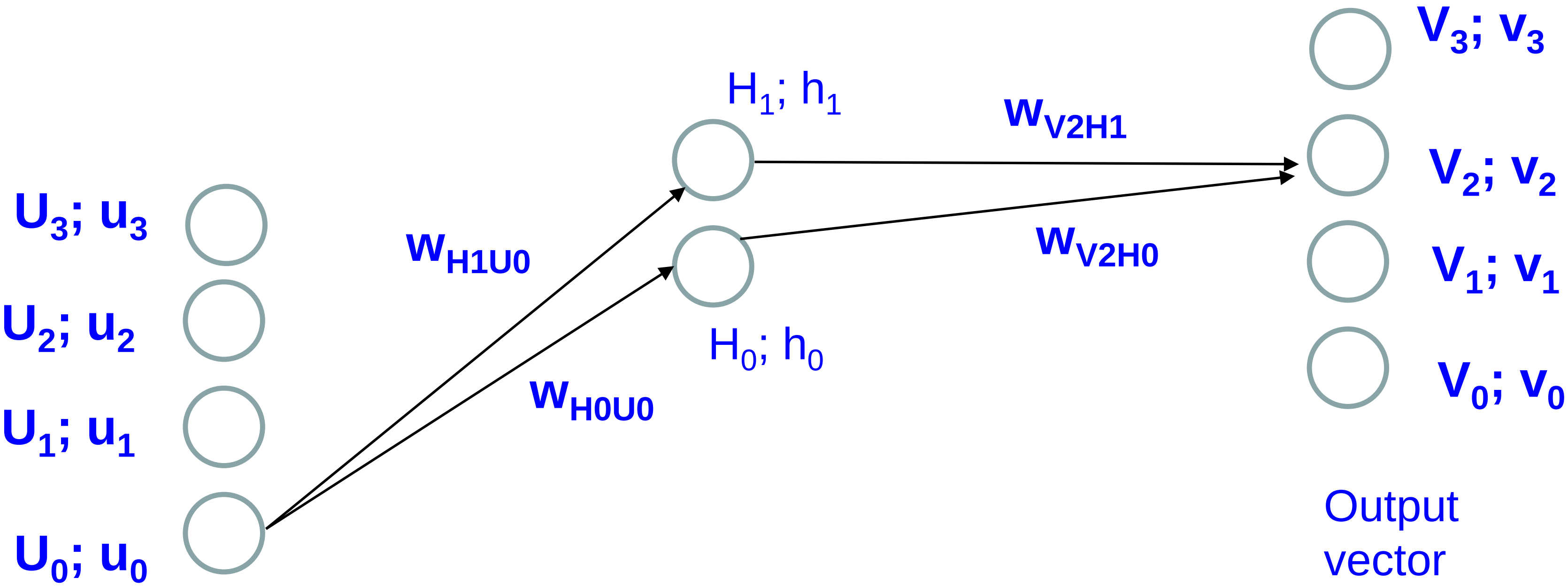
$$\frac{\delta E}{\delta w_{V_1H_0}} = -0 + \frac{e^{W_{V1}.W_{U0}}}{e^{W_{V0}.W_{U0}} + e^{W_{V1}.W_{U0}} + e^{W_{V2}.W_{U0}} + e^{W_{V3}.W_{U0}})}.w_{H_0U_0}$$

$$= v_1.w_{H_0U_0}$$

$$\Rightarrow \Delta w_{V_1H_0} = -\eta v_1 w_{H_0U_0} = -\eta v_1 o_{H_0}$$

Word2vec n/w

Capital letter for NAME of neuron; small letter for output from the same neuron



Output vector V

Input vector U

Weights go from all neurons to all neurons in the next layer; shown For only one input and output

Cntd: Weight change for input to hidden layer, say, w_{H0U0}

$$\frac{\delta E}{\delta w_{H_0U_0}}$$

$$= -w_{V_2H_0} + \frac{w_{V_0H_0}e^{W_{V_0}.W_{U_0}} + w_{V_1H_0}e^{W_{V_1}.W_{U_0}} + w_{V_2H_0}e^{W_{V_2}.W_{U_0}} + w_{V_3H_0}e^{W_{V_3}.W_{U_0}}}{e^{W_{V_0}.W_{U_0}} + e^{W_{V_1}.W_{U_0}} + e^{W_{V_2}.W_{U_0}} + e^{W_{V_3}.W_{U_0}}}$$

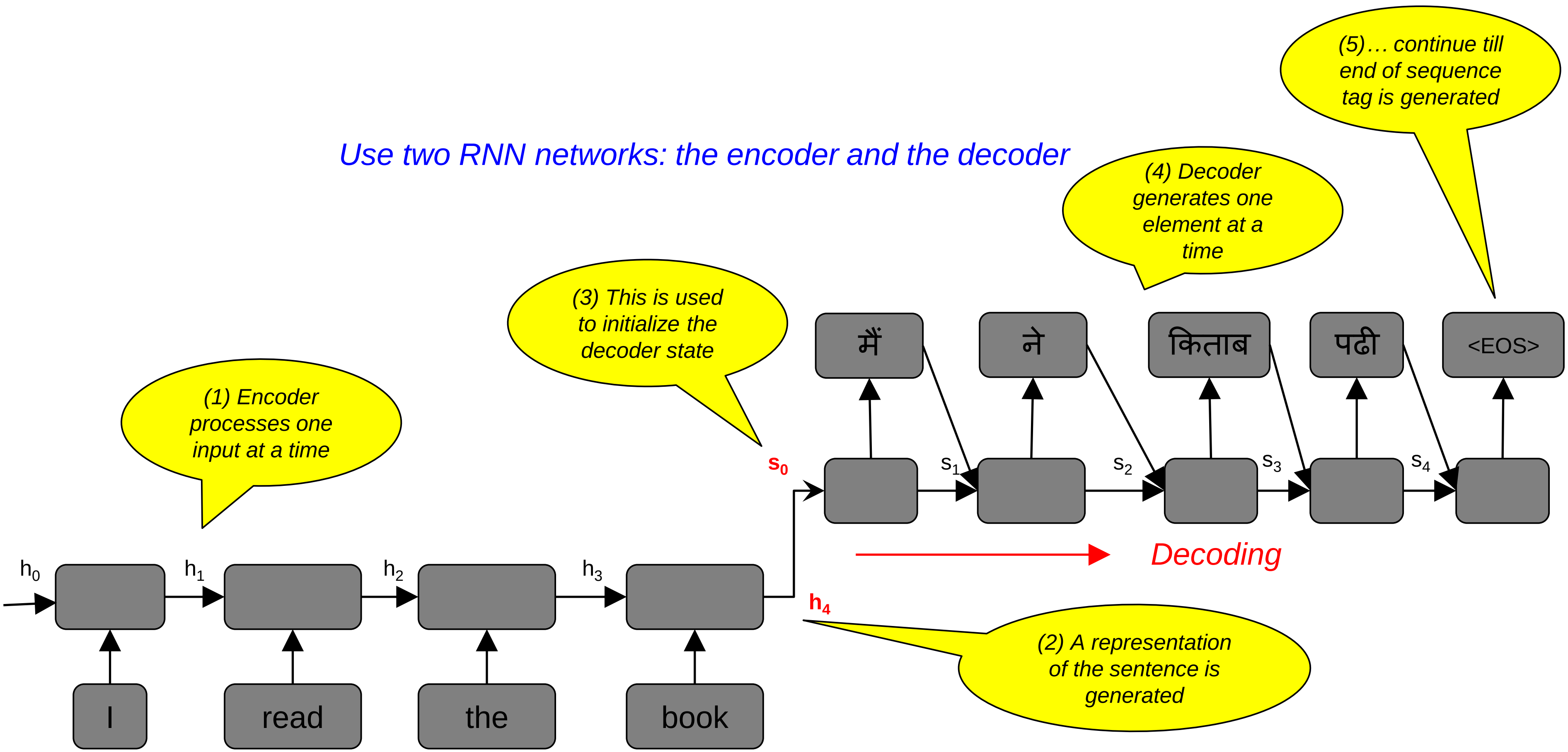
$$= -w_{V_2H_0} + w_{V_0H_0}v_0 + w_{V_1H_0}v_1 + w_{V_2H_0}v_2 + w_{V_3H_0}v_3$$

$$\Rightarrow \Delta w_{H_0U_0} = \eta[(1-v_2)w_{V_2H_0} - w_{V_0H_0}v_0 - w_{V_1H_0}v_1 - w_{V_3H_0}v_3]$$

Week7

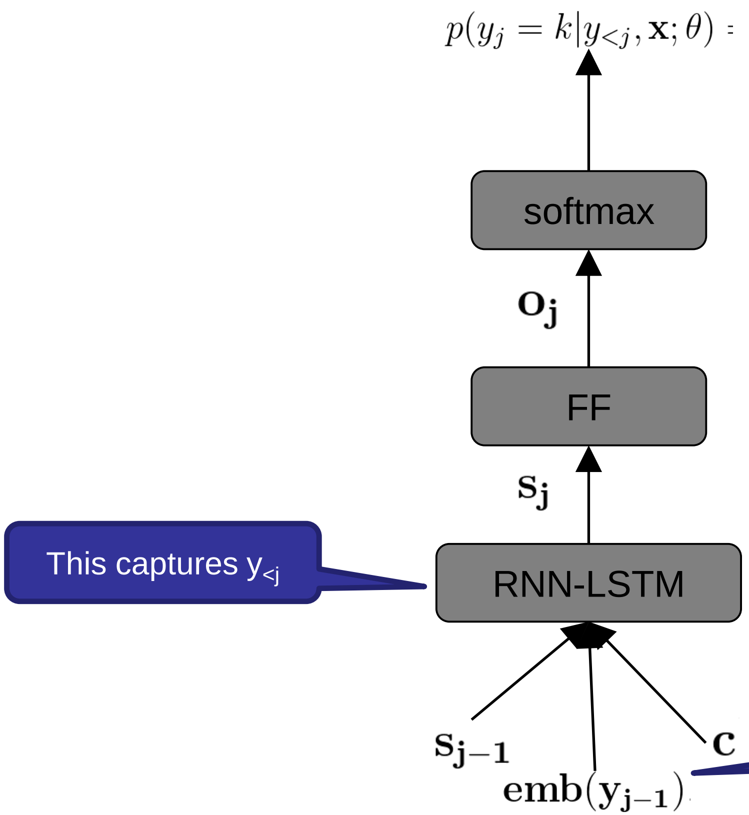
Encode - Decode Paradigm Explained

Use two RNN networks: the encoder and the decoder



https://developer.nvidia.com/blog/introduction-neural-machine-translation-gpus-part-2/
Sequence to Sequence Learning with Neural Networks Ilya Sutskever, Oriol Vinyals, Quoc V. Le. arxiv pre-print [link]

What is the decoder doing at each time-step?



$$softmax(o_{jk}) = \frac{\exp(o_{jk})}{\sum_{m=0}^{m=T} \exp(o_{jm})}$$

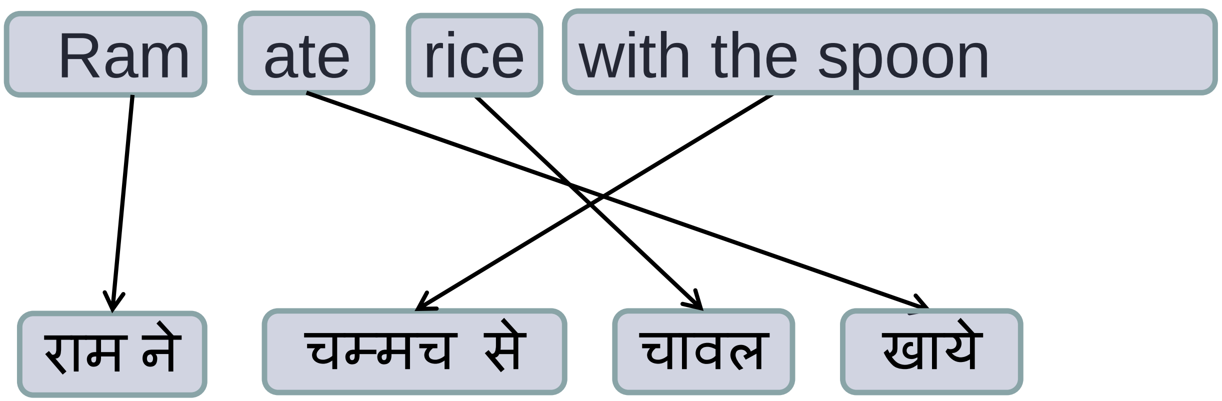
$$\mathbf{o_j} = FF(s_j)$$

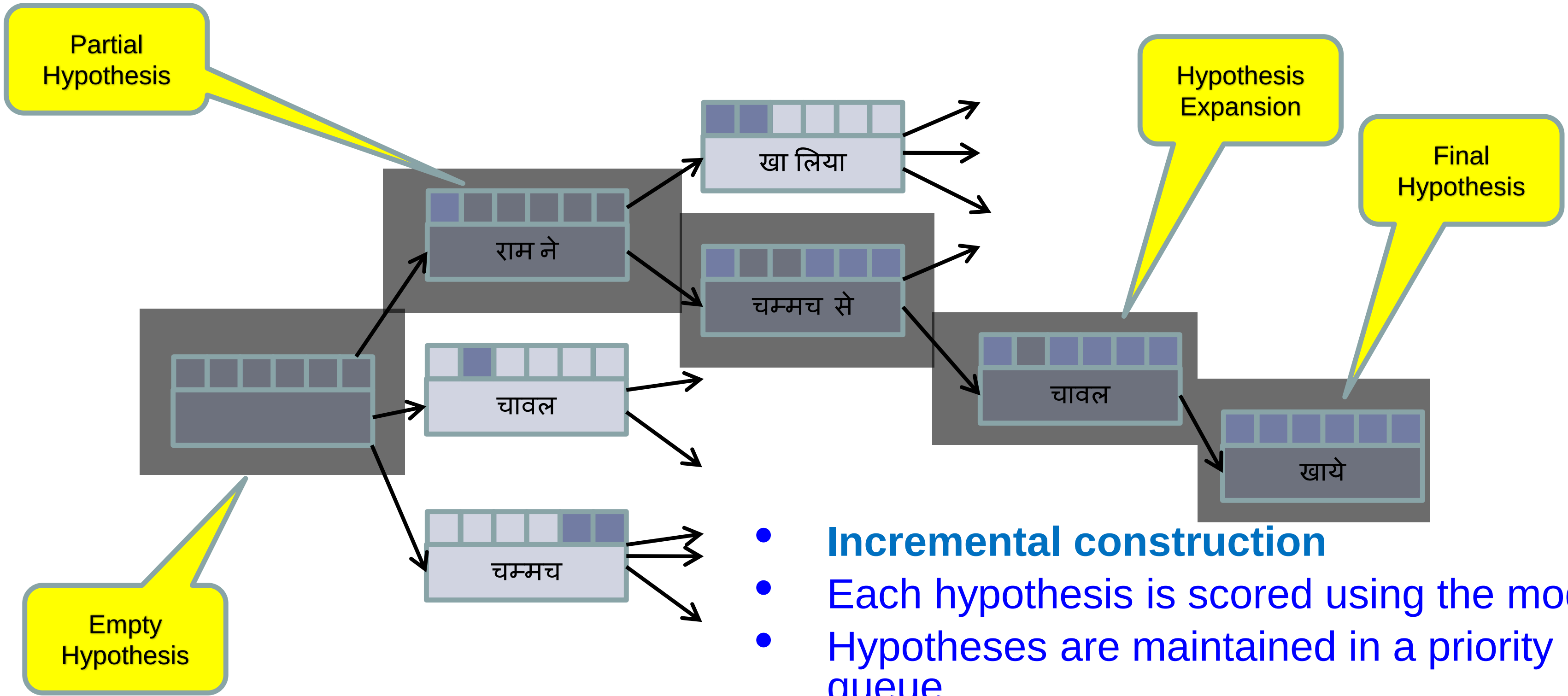
$$\mathbf{s_j} = g(\mathbf{s_{j-1}}, \mathbf{emb(y_{j-1})}, \mathbf{c})$$

This captures x, $c=h_4$

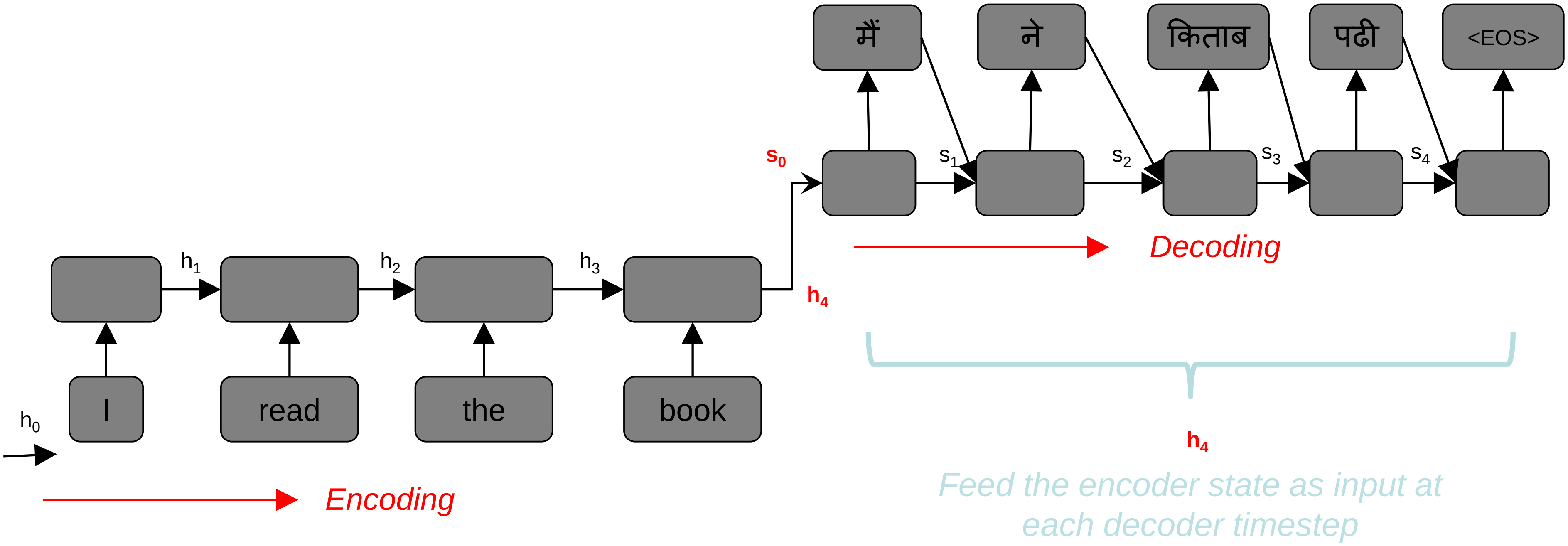
Decoding

Searching for the best translations in the space of all translations





- **Incremental construction**
- Each hypothesis is scored using the model
- Hypotheses are maintained in a priority queue



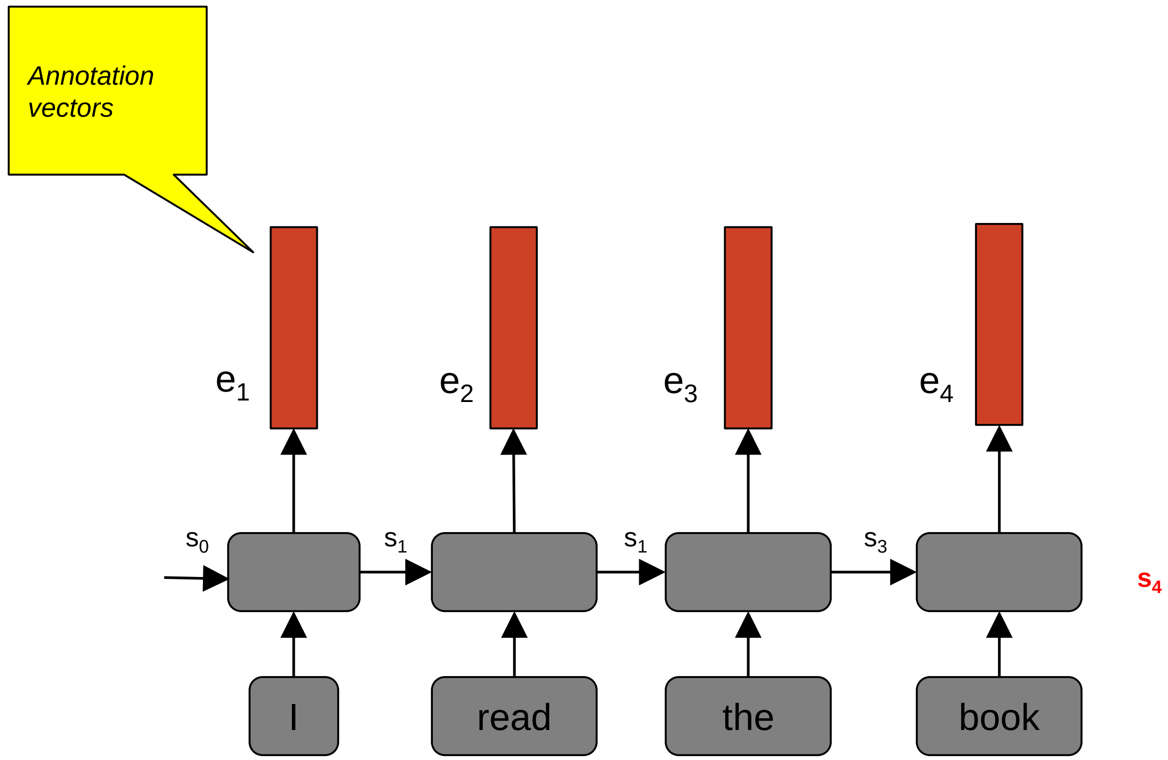
मैं
ने
किताब
पढी
<EOS>
s0
s1
s2
s3
s4
Decoding
h1
h2
h3
h4
I
read
the
book
h0
Encoding
h4
Feed the encoder state as input at
each decoder timestep

The entire source sentence is represented by a single vector

Problems

- Insufficient to represent to capture all the syntactic and semantic complexities
  - *Solution: Use a richer representation for the sentences*
- Long-term dependencies: Source sentence representation not useful after few decoder time steps
  - *Solution: Make source sentence information when making the next prediction*
  - *Even better, make **RELEVANT** source sentence information available*

Encode - Attend - Decode Paradigm



Represent the source sentence by the **set of output vectors** from the encoder

Each output vector at time *t* is a contextual representation of the input at time *t*

Let's call these encoder output vectors ***annotation vectors***

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *ICLR 2015.*

https://developer.nvidia.com/blog/introduction-neural-machine-translation-gpus-part-3/

CNN

Two motivation points

- 1. Reduced number of parameters

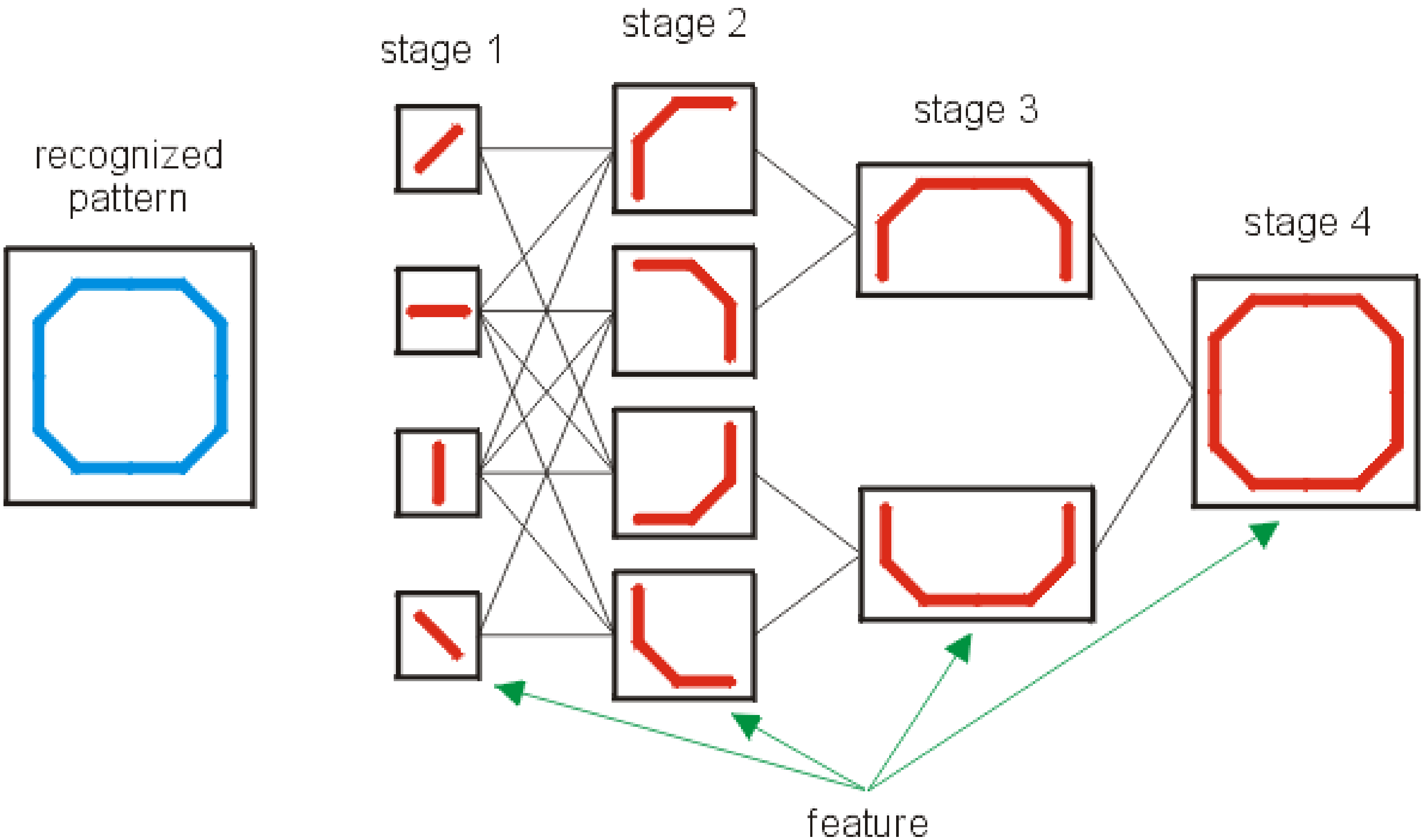
- 2. Stepwise extraction of features

- These two are applicable to any AI situation, and not only vision and image processing

CNN= feedforward like + recurrent like!

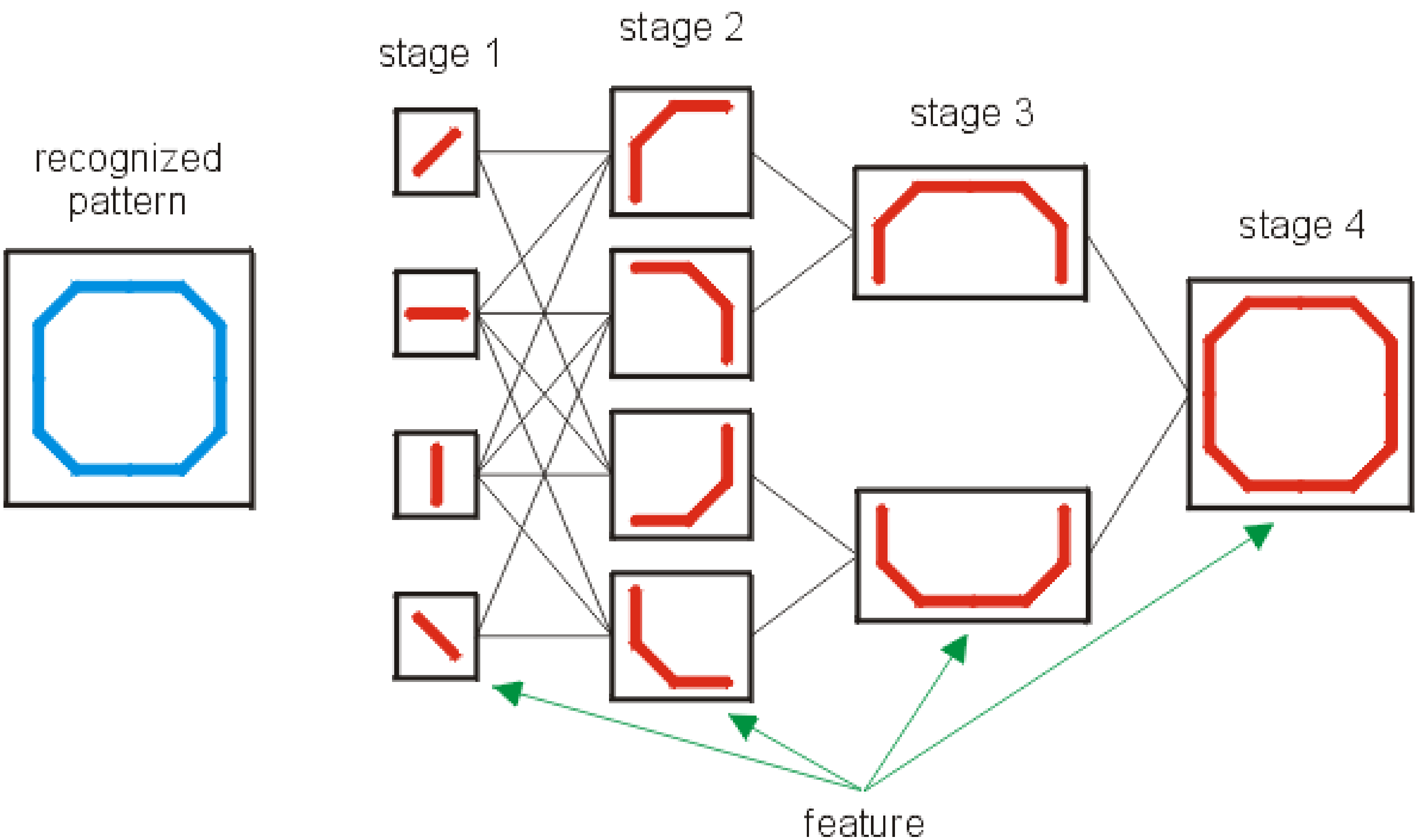
- Whatever we learnt so far in FF-BP is useful to understand CNN
- So also is the case with RNN (and LSTM)
- Input divided into regions and fed forward
- Window slides over the input: input changes, but ‘filter’ parameters remain same
- That is like RNN

Genesis: Neocognitron (Fukusima, 1980)

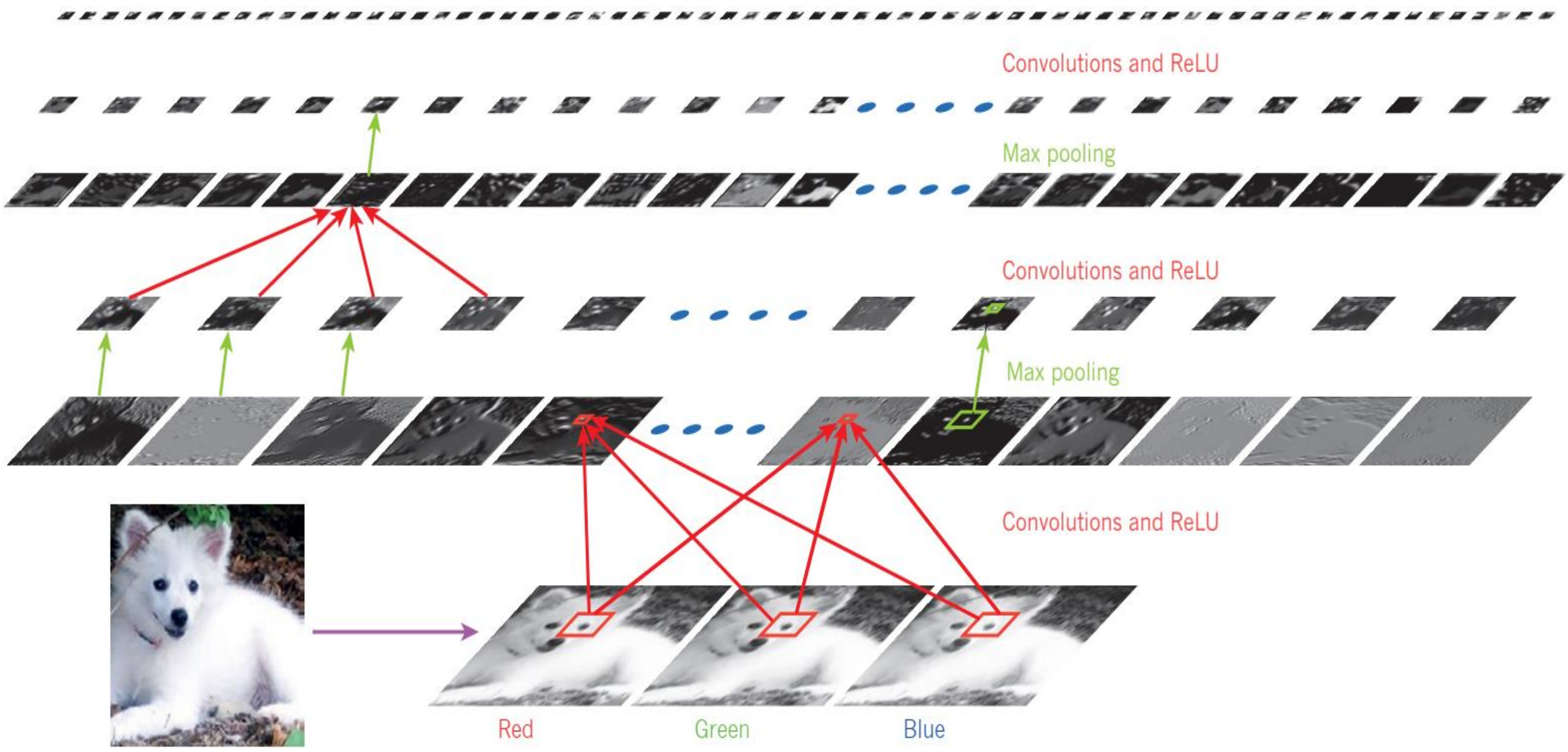


Week8

CNN Genesis: Neocognitron (Fukusima, 1980)



A typical ConvNet



Lecun, Bengio, Hinton, Nature, 2015

Why CNN became a rage: image

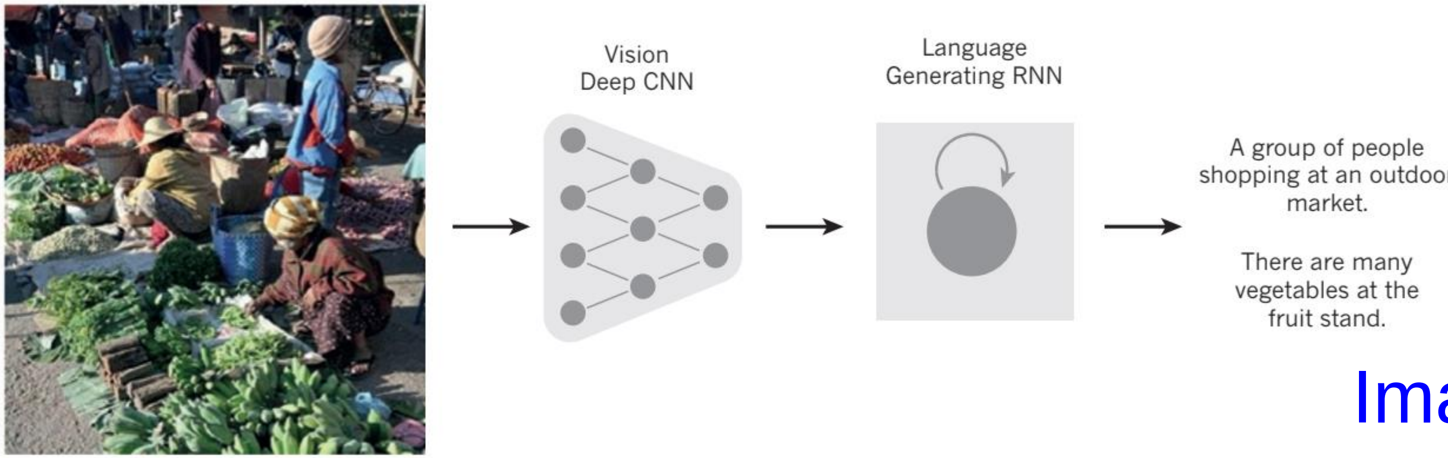


Image Captioning-1

A **stop** sign is on a road with a mountain in the background

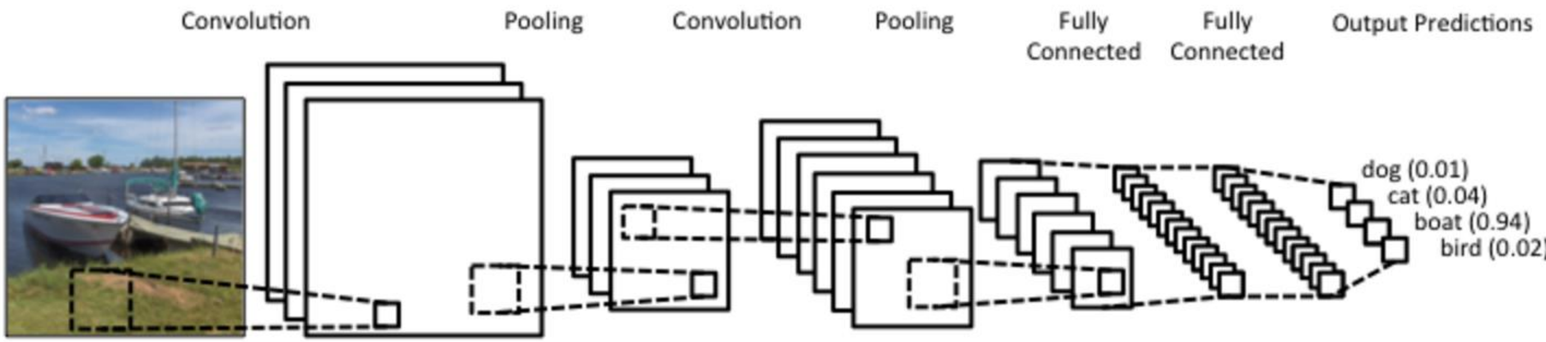
Image Captioning-2

Role of ImageNet

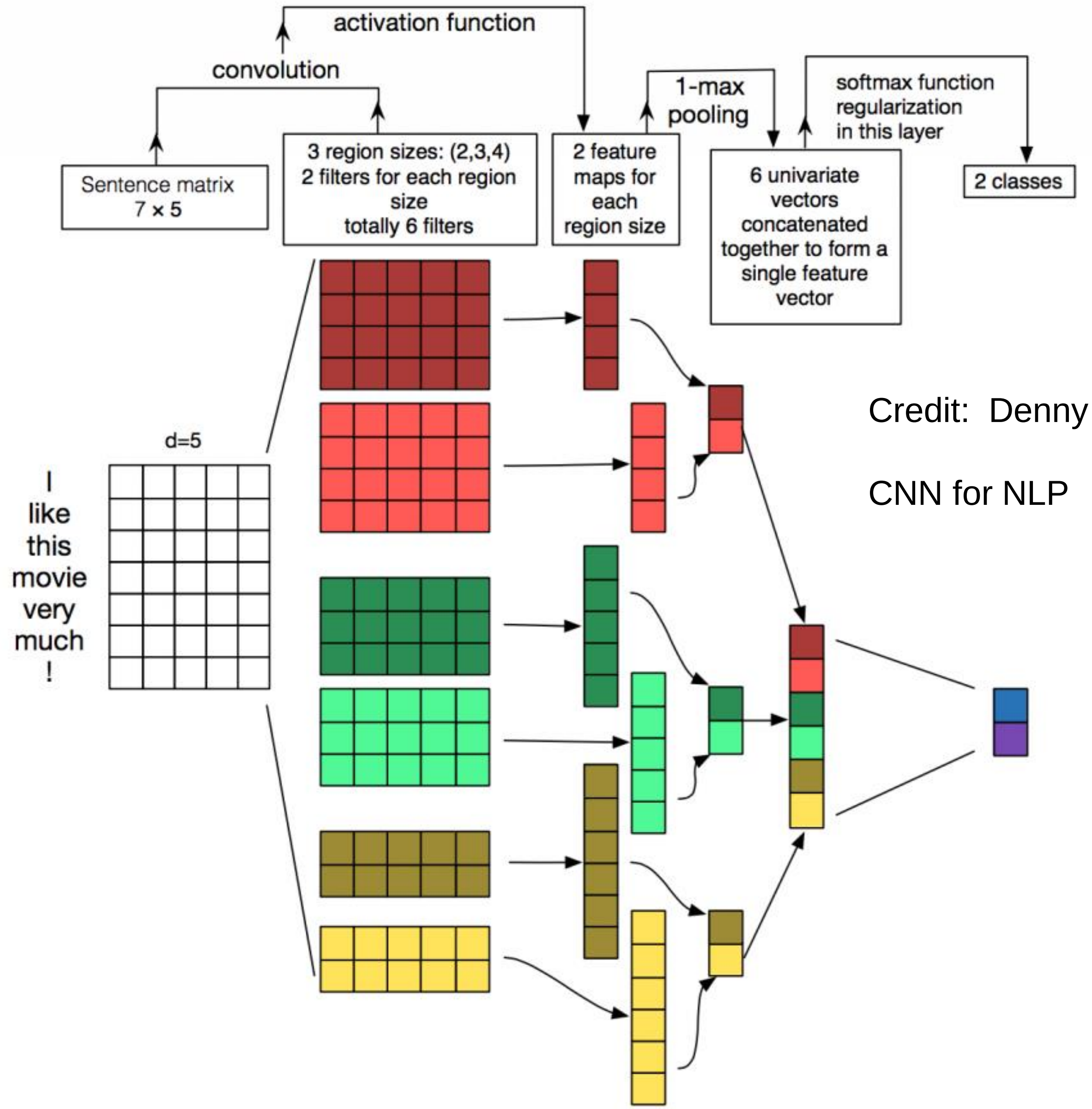
- Million images from the web
- 1,000 different classes
- Spectacular results!
- Almost halving the error rates of the best competing approaches1.

Learning in CNN

- **Automatically learns the values of its filters**
- For example, in Image Classification learn to
  - detect edges from raw pixels in the first layer,
  - then use the edges to detect simple shapes in the second layer,
  - and then use these shapes to deter higher-level features, such as facial shapes in higher layers.
  - The last layer is then a classifier that uses these high-level features.



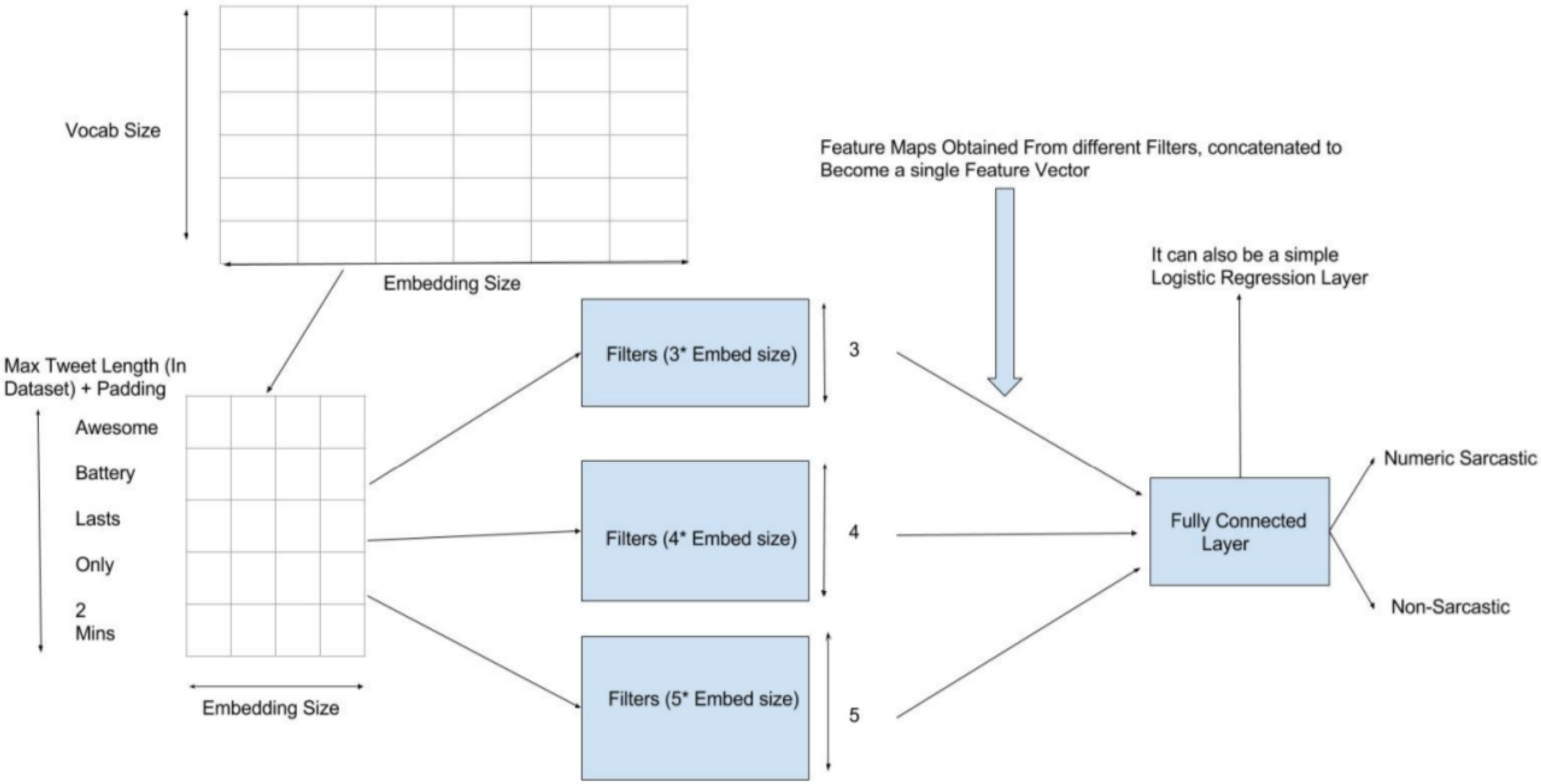
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/



Credit: Denny Britz

CNN for NLP

CNN-FF for Sarcasm

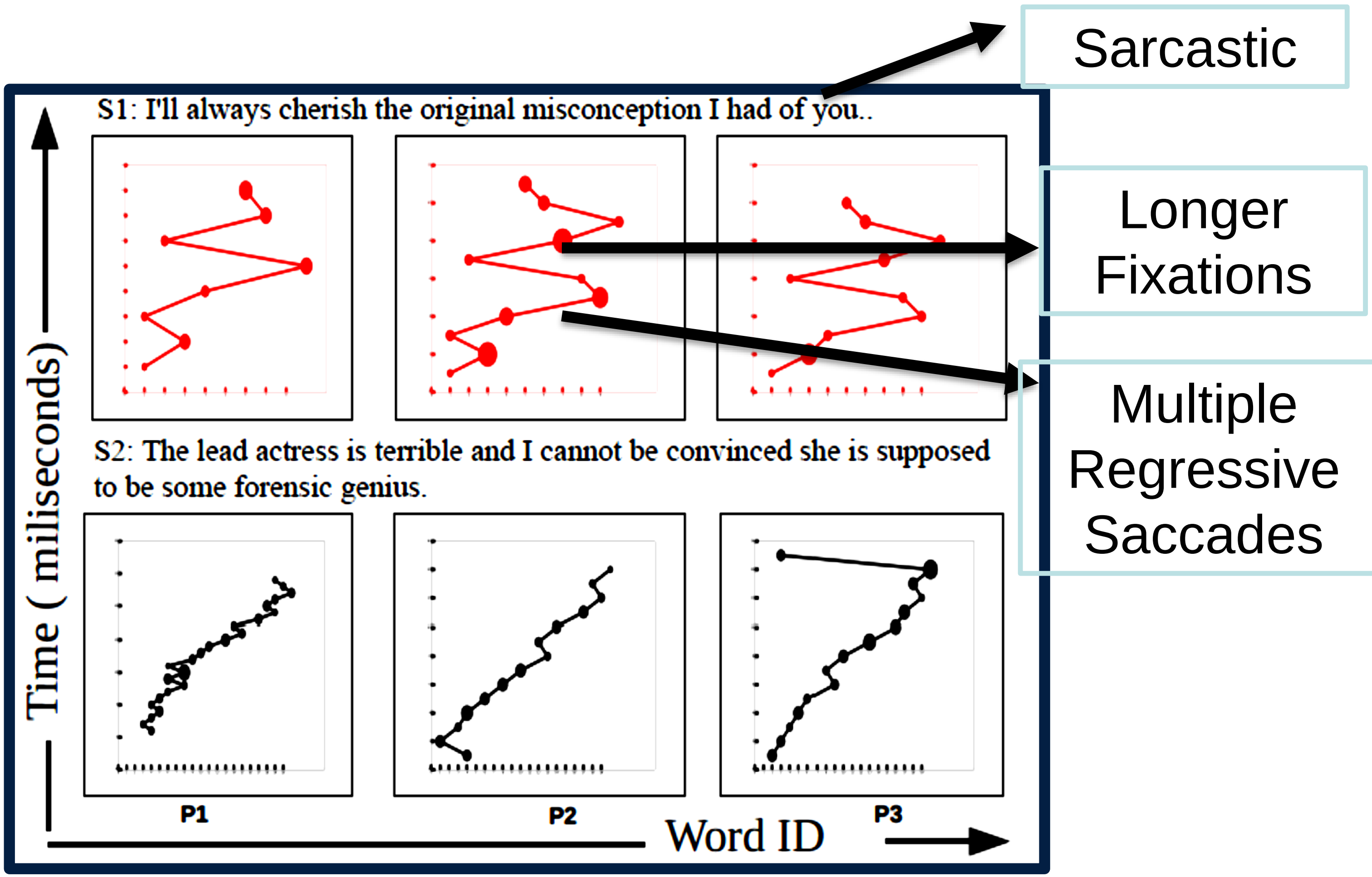


Comparison of results (1: sarcastic, 0: non-sarcastic)

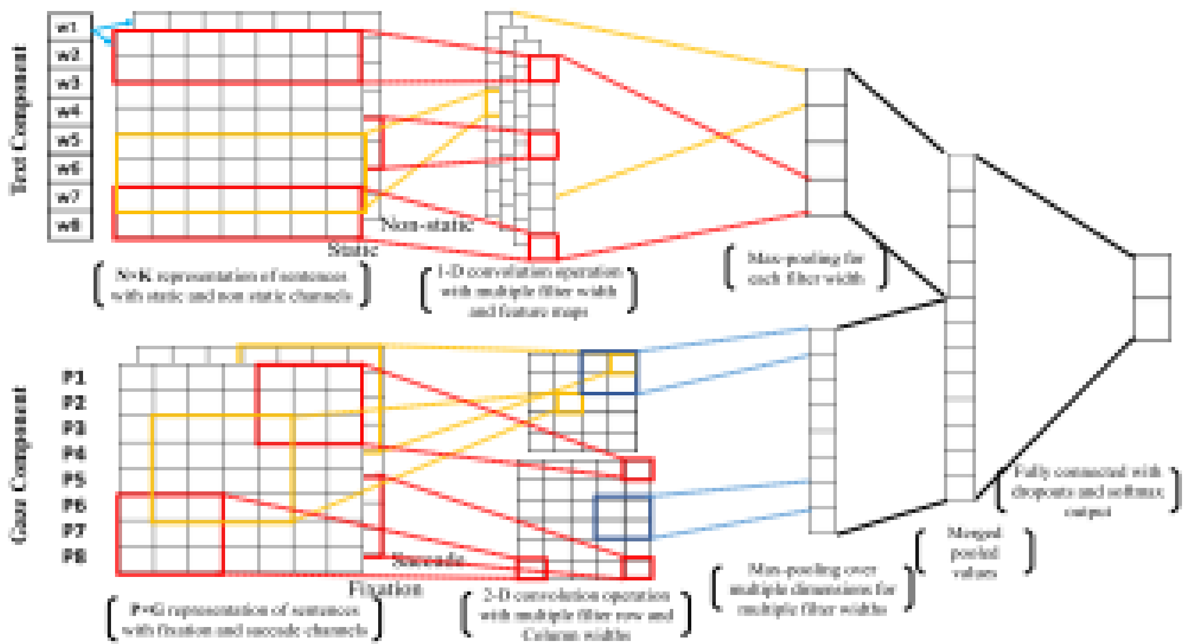
Approaches	Precision			Recall			F-score		
	P(1)	**P(0)**	**P(avg)**	**R(1)**	**R(0)**	**R(avg)**	**F(1)**	**F(0)**	**F(avg)**
Past Approaches									
Buschmeier et.al.	0.19	0.98	0.84	0.99	0.07	0.24	0.32	0.13	0.16
Liebrecht et.al.	0.19	1.00	0.85	1.00	0.07	0.24	0.32	0.13	0.17
Gonzalez et.al.	0.19	0.96	0.83	0.99	0.06	0.23	0.32	0.12	0.15
Joshi et.al.	0.20	1.00	0.86	1.00	0.13	0.29	0.33	0.23	**0.25**
Rule-Based Approaches									
Approach-1	0.53	0.87	0.81	0.39	0.92	0.83	0.45	0.90	**0.82**
Approach-2	0.44	0.85	0.78	0.28	0.92	0.81	0.34	0.89	0.79
Machine-Learning Based Approaches									
SVM	0.50	0.95	0.87	0.80	0.82	0.82	0.61	0.88	**0.83**
KNN	0.36	0.94	0.84	0.81	0.68	0.70	0.50	0.79	0.74
Random Forest	0.47	0.93	0.85	0.74	0.81	0.80	0.57	0.87	0.82
Deep-Learning Based Approaches									
CNN-FF	**0.88**	**0.94**	**0.93**	**0.71**	**0.98**	**0.93**	**0.79**	**0.96**	**0.93**
CNN-LSTM-FF	0.82	0.94	0.92	0.72	0.96	0.92	0.77	0.95	0.92
LSTM-FF	0.76	0.93	0.90	0.68	0.95	0.90	0.72	0.94	0.90

back

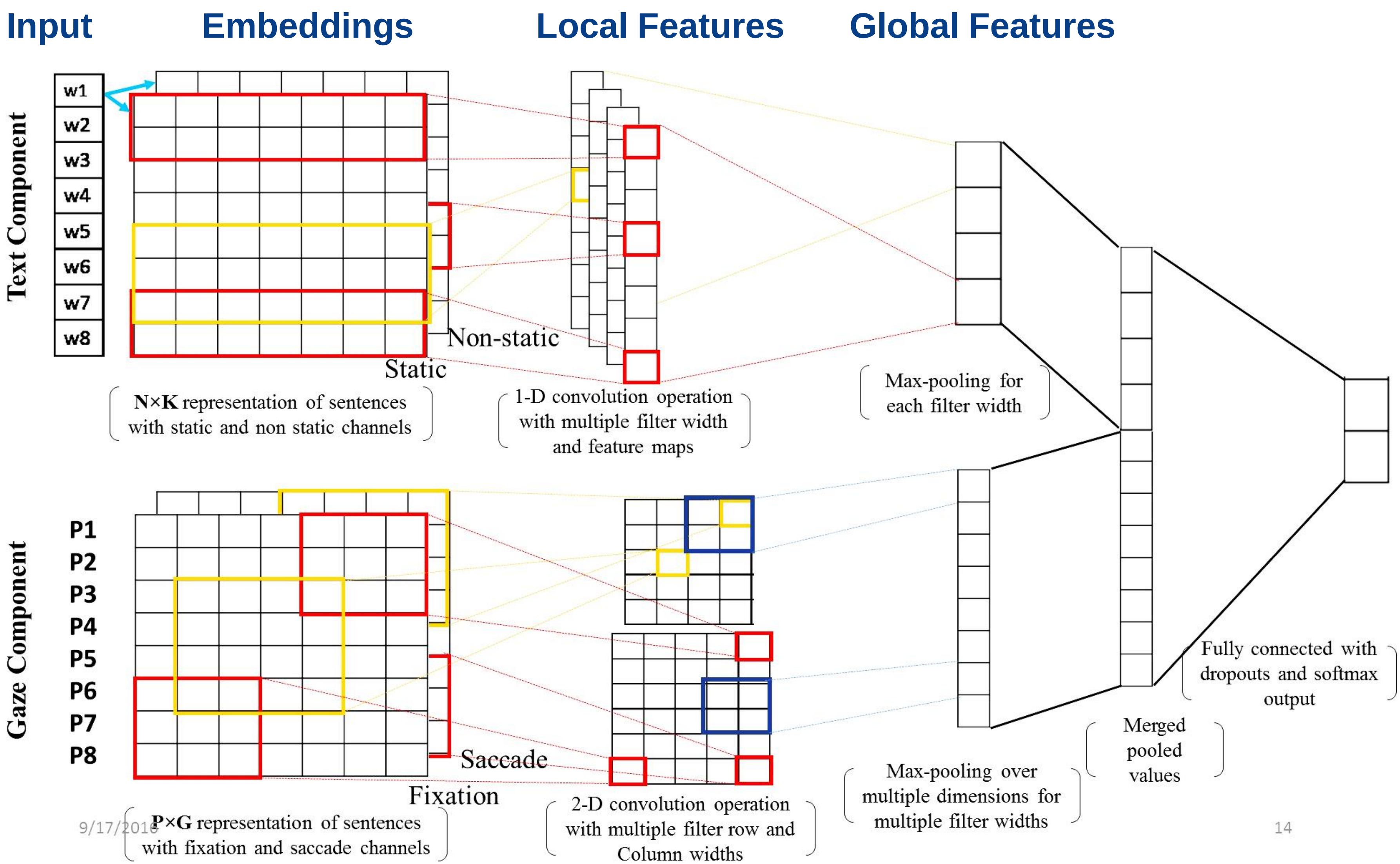
Sentiment Annotation and Eye Movement



Abhijit Mishra, Kuntal Dey and Pushpak Bhattacharyya, *Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification Using Convolutional Neural Network*, **ACL 2017**, Vancouver, Canada, July 30-August 4, 2017.



Neural Network Architecture



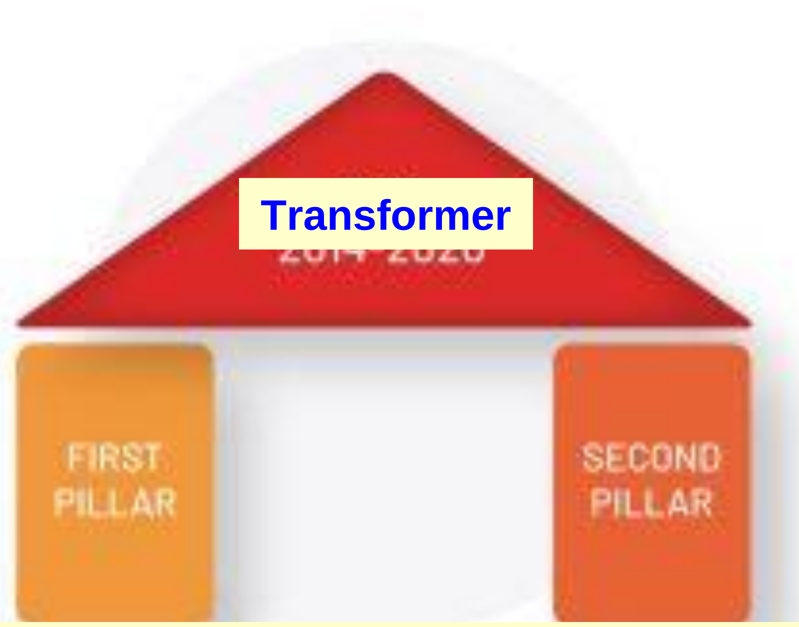
Results – Sarcasm Detection

	Configuration	P	R	F
Traditional systems based on textual features	Näive Bayes	69.1	60.1	60.5
	Multi-layered Perceptron	69.7	70.4	69.9
	SVM (Linear Kernel)	72.1	71.9	72
Systems by Riloff et al. (2013)	Text based (Ordered)	49	46	47
	Text + Gaze (Unordered)	46	41	42
System by Joshi et al. (2015)	Text based (best)	70.7	69.8	64.2
Systems by Mishra et al. (2016b)	Gaze based (Best)	73	73.8	73.1
	Text based (Best)	72.1	71.9	72
	Text + Gaze (Best)	76.5	75.3	75.7
CNN with only text input (Kim, 2014)	StaticText	67.17	66.38	66.77
	NonStaticText	84.19	**87.03**	85.59
	MultiChannelText	84.28	**87.03**	85.63
CNN with only gaze input	Fixation	74.39	69.62	71.93
	Saccade	68.58	68.23	68.40
	MultiChannelGaze	67.93	67.72	67.82
CNN with both text and gaze Input	StaticText + Fixation	72.38	71.93	72.15
	StaticText + Saccade	73.12	72.14	72.63
	StaticText + MultiChannelGaze	71.41	71.03	71.22
	NonStaticText + Fixation	**87.42**	85.2	86.30
	NonStaticText + Saccade	84.84	82.68	83.75
	NonStaticText + MultiChannelGaze	84.98	82.79	83.87
	MultiChannelText + Fixation	87.03	86.92	**86.97**
	MultiChannelText + Saccade	81.98	81.08	81.53
	MultiChannelText + MultiChannelGaze	83.11	81.69	82.39

Attention and Transformer

Arguably, the most important application-
MACHINE TRANSLATION

Two Pillars of Transformer



Attention + Positional Encoding

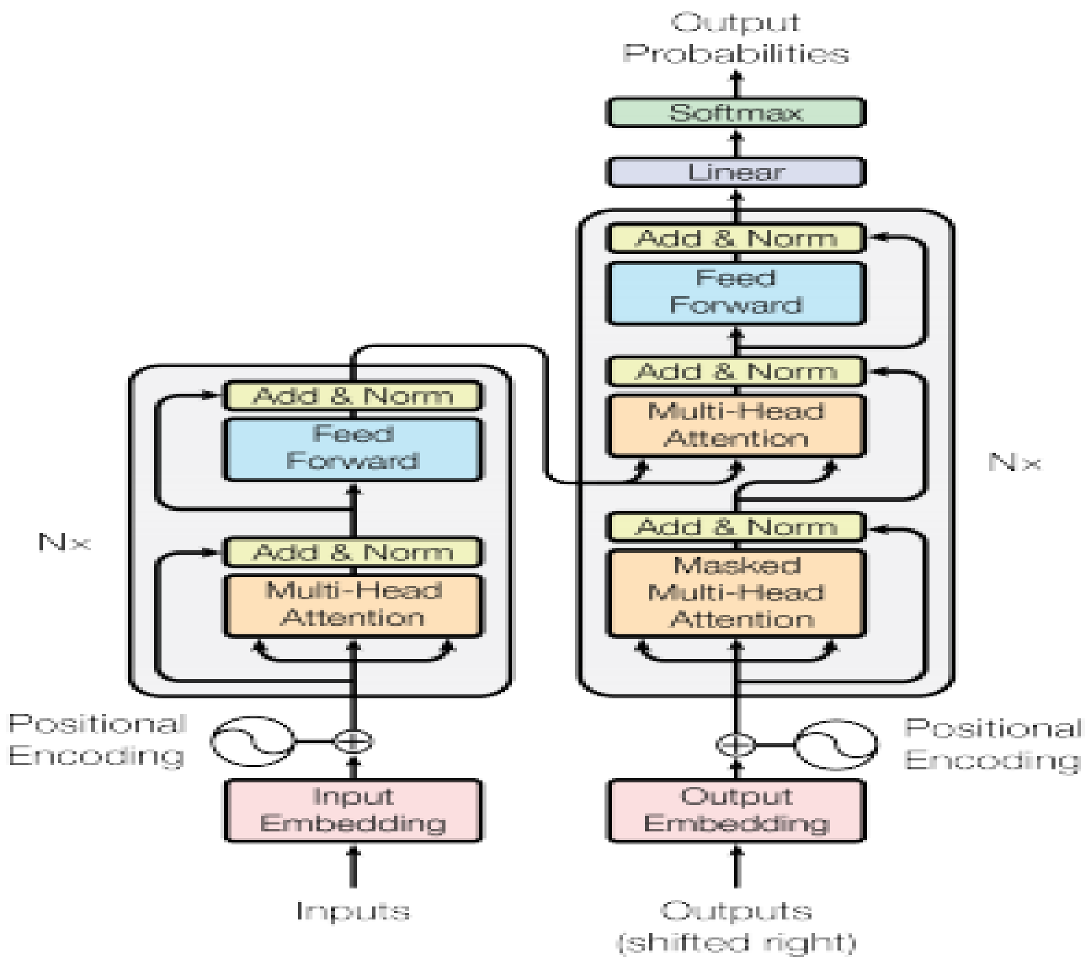
= Transformer

Week9

Prompting, Reasoning, Bias, SSMT, QE, APE, Fake-News & Half-Truth Detection, Query Intent Detection and Speech Emotion Recognition

Week10

A classic diagram and a classic paper

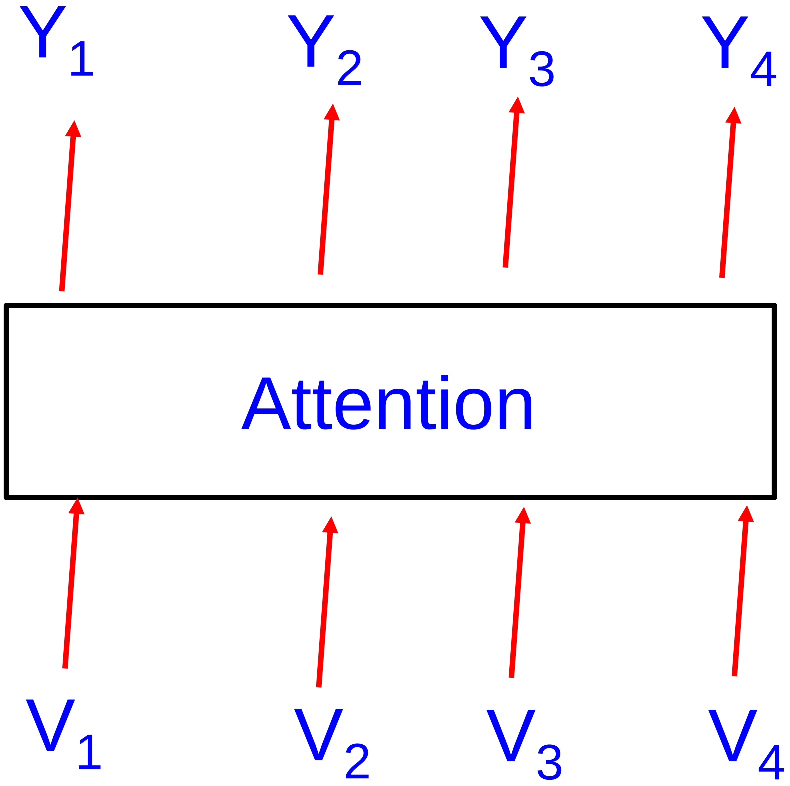


Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." NeurIPS (2017).

http://nlp.seas.harvard.edu/2018/04/03/attention.html
http://jalammar.github.io/illustrated-transformer/

Attention: Self, Multi-headed, Cross

Self Attention Block



Bank of the river

Word Embedding and Contextual Word Embedding

- Consider the phrase "*bank of the river*"
- Word embeddings of '*bank*', '*of*', '*the*', '*river*': V_1, V_2, V_3, V_4
- Now create a 'score' vector S_i for each word vector
- S_1: ($V_1.V_1$, $V_1.V_2$, $V_1.V_3$, $V_1.V_4$)
- Similarly, S_2, S_3, S_4

S-matrix

$$S = \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

S-scaled matrix

$$S - scaled = \frac{1}{\sqrt{d_k}} \times \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

W-matrix

$$W = \begin{bmatrix} w_{11}\ w_{12}\ w_{13}\ w_{14} \\ w_{21}\ w_{22}\ w_{23}\ w_{24} \\ w_{31}\ w_{32}\ w_{33}\ w_{34} \\ w_{41}\ w_{42}\ w_{43}\ w_{44} \end{bmatrix}$$

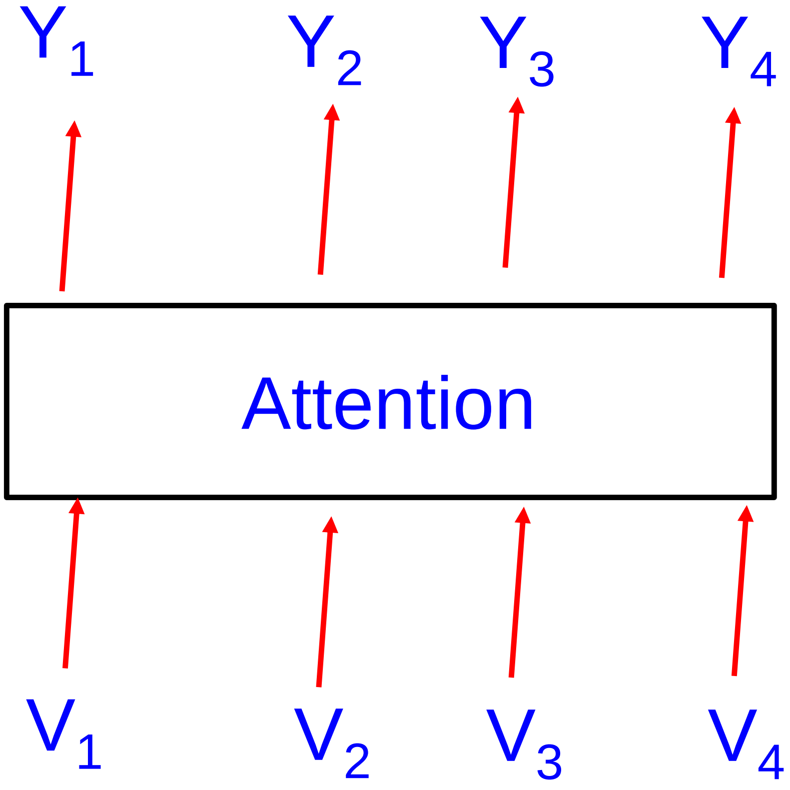
$$W_i - vector = soft\max\left(\frac{S_i - vector}{\sqrt{d_k}}\right)$$

Y-matrix

$$Y = \begin{bmatrix} y_{11}\, y_{12}\, y_{13}\, y_{14} \\ y_{21}\, y_{22}\, y_{23}\, y_{24} \\ y_{31}\, y_{32}\, y_{33}\, y_{34} \\ y_{41}\, y_{42}\, y_{43}\, y_{44} \end{bmatrix}$$

$$Y_i - vector = w_{11}.V_1 + w_{12}.V_2 + w_{13}.V_3 + w_{14}.V_4$$

Attention Block



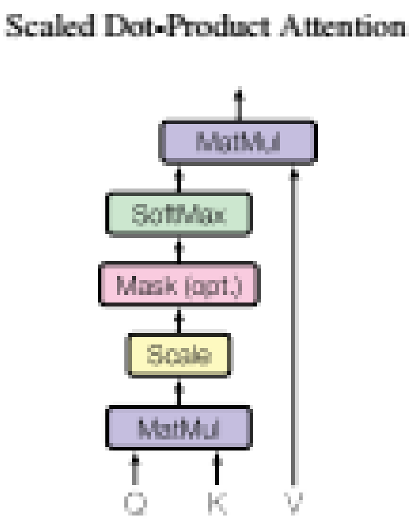
Bank of the river

Query, Key and Value

$$attention(Q,K,V) = soft\max\left(\frac{Q\,.K^{T}}{\sqrt{d_k}}\right).V$$

Query, Key and Value with LEANABLE Parameter (1/2)

$$attention(Q,K,V) = soft\max\left(\frac{W^Q Q \; .W^K K^T}{\sqrt{d_k}}\right).W^V V$$



W^Q, W^K and W^V can be the weights of 3 linear layers of neurons which can be learnt by gradient descent

Query, Key and Value with LEANABLE Parameter (2/2)

$$(W_Q Q^T).(W_K K) = (W_V V)$$

W_Q, W_K and W_V can be the weights of 3 linear layers of neurons which can be learnt by gradient descent

Week11

Attempts at Automation

- InstructGPT:
  - ***Command/Request/Order → Response***
- ChatGPT:
  - Carry out a ***conversation***
  - Respect context (state), personalization, quality and quantity and respond
    - Input: *I have been promoted*
    - Appropriate response: *I am delighted/congratulations/great* ..
    - Inappropriate: *why did they promote you?*

Gricean Maxims: Cooperative Principle in Converstaion (Wikipedia)

- **Quantity, Quality, Relation, and Manner**
- Paul Grice, philosopher of language
- "*Make your contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged*".
- Captures the LINK between utterances

Maxim of Quantity (length and depth)

- *Be **informative,*** and submaxims are:
  - Make your contribution as informative as is required (for the current purposes of the exchange).
  - Do not make your contribution more informative than is required.
- Grice’s analogy: "If you are assisting me to mend a car, I expect your contribution to be neither more nor less than is required. If, for example, at a particular stage I need four screws, I expect you to hand me four, rather than two or six."

Maxim of Quality (truth)

- Be ***Truthful***
- Submaxims:
  - Do not say what you believe is false.
  - Do not say that for which you lack adequate evidence
- Grice’s analogy: "I expect your contributions to be genuine and not spurious. If I need sugar as an ingredient in the cake you are assisting me to make, I do not expect you to hand me salt; if I need a spoon, I do not expect a trick spoon made of rubber."

Maxim of Relation (relevance)

- Information is ***relevant*** to the current exchange; therefore omitting any irrelevant information
- Grice’s analogy for this maxim: "I expect a partner’s contribution to be appropriate to the immediate needs at each stage of the transaction. If I am mixing ingredients for a cake, I do not expect to be handed a good book, or even an oven cloth (though this might be an appropriate contribution at a later stage)."

Maxim of Manner (clarity)

- Be ***perspicuous***
- Submaxims:
  - Avoid obscurity of expression — i.e., avoid language that is difficult to understand.
  - Avoid ambiguity — i.e., avoid language that can be interpreted in multiple ways.
  - Be brief — i.e., avoid unnecessary prolixity.
  - Be orderly — i.e., provide information in an order that makes sense, and makes it easy for the recipient to process it.

Week12

AI chatbots compared: Bard vs. Bing vs. ChatGPT

https://www.theverge.com/2023/3/24/23653377/ai-chatbots-comparison-bard-bing-chatgpt-gpt-4

Comparison: Chatbots

Google’s Bard (https://bard.google.com/),

Microsoft’s Bing (https://www.theverge.com/2023/3/24/23653377/ai-chatbots-comparison-bard-bing-chatgpt-gpt-4),

OpenAI’s ChatGPT (https://chat.openai.com/chat#)

3 stages of LLM based CAI

- Generative Pretraining (GP)
- Supervised Fine Tuning (SFT)
- Reinforcement Learning from Human Feedback (RLHF)

Enter Pragmatics

Modeling

***P(e)*: "language" model**

$$e^{*} = \arg\max_e P(e \mid f)$$

$$= \arg\max_e [P(e)P(f \mid e)]$$

- Dialogue Act Classification (DAC): *f* → *Dialogue Sequence, e* → *Dialogue turn labels*
- Dialogue Intent: *f* → *dialogue sequence, e* → *dialogue turns with Intent like 'question', 'elaboration', 'affirmation', 'command/request' etc.*

Elements of Pragmatics (1/2)

- Deixis (literally, ‘pointing with words’: temporal- *now, then;* spatial- *here, there*; personal- *I*, *you, he, they*; definite-indefinite- *this, that, those)*

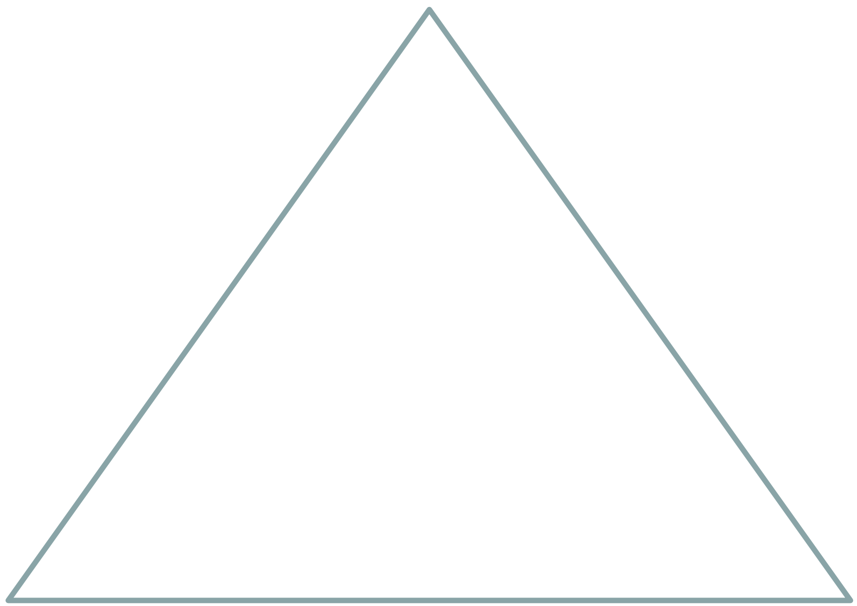
- Presupposition: (*untie the shoe* → presupposes *the shoe was tied before*)

Elements of Pragmatics (2/2)

- Speech Acts: (*I pronounce you man and wife)*- **locutionary, illocutionary, and perlocutionary**
- Implicatures: (*A: shall we go for a walk? B: It is raining outside*)
- Politeness: (*close the door → please close the door → can you close the door → would you mind closing the door)*
- Information Structure: ordering of information (??*The table is under the flower pot*- odd; smaller object first mention) credit: Handke

The Trinity of Pragmatics

Linguistic Expression



Speaker

Hearer

Diexis

Credit:
https://doi.org/10.1093/acrefore/9780199384655.013.213

Speech Act

Kinds of Speech Act

- Locutionary
- Illocutionary
- Perlocutionary
- Performative Speech acts

Implicatures

Computational Perspective: Conversational AI

Dialogue Based Computation

Zihao He, Leili Tavabi, Kristina Lerman, and Mohammad Soleymani. 2021. Speaker Turn Modeling for Dialogue Act Classification. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2150–2157, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tulika Saha, Aditya Patra, Sriparna Saha and Pushpak Bhattacharyya, *Towards Emotion-aided Multi-modal Dialogue Act Classification*, Association of Computational Linguistics Conference (**ACL 2020**), Seattle USA, 5-10 July, 2020.

Week13

Summarization

SUMMARIZATION

- Task of automatically creating a compressed version of the text document (set of tweets, web-page, single/multi-document) that should be relevant, non-redundant and representative of the main idea of the text.
- A text that is produced from one or more texts that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that.
- Metric:

$$\textit{Compression Ratio} = \#word_{summary} / \#word_{document}$$

NLP Layer

Pragmatics

Semantics

Syntax

Lexical Level

Summarization Categorization

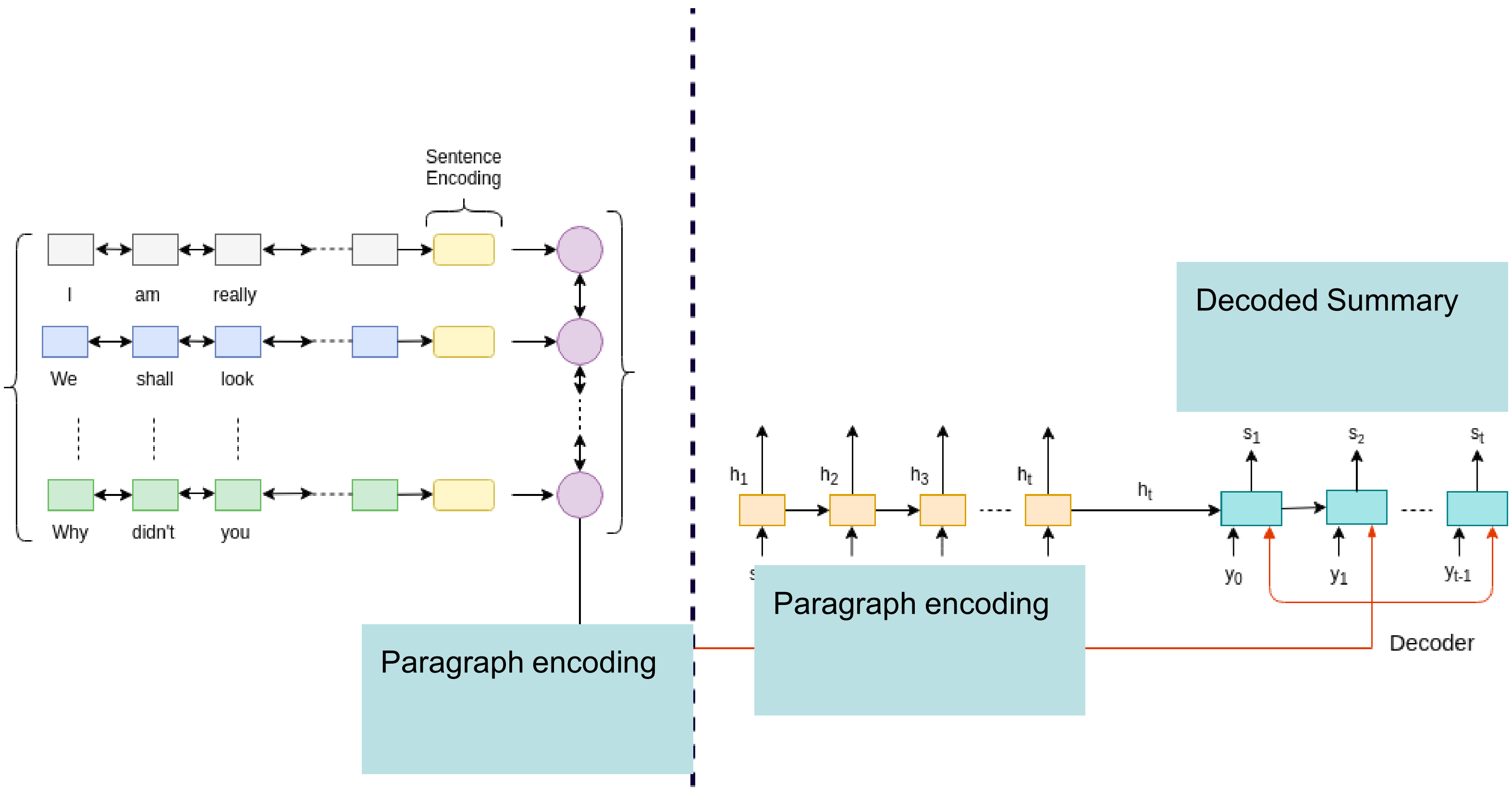
- Broad Categorization
  - Extractive: sentences from the input text form part of summary
  - Abstractive: Essence+Natural Language Generation
- Other categorizations:
  - # Document: Single and Multi document
  - Purpose: Generic and Query focused
  - Miscellaneous: Personalized, Sentiment-based, Update, E-mail-based, web-based

Handling Morphology in Abstractive Summarization

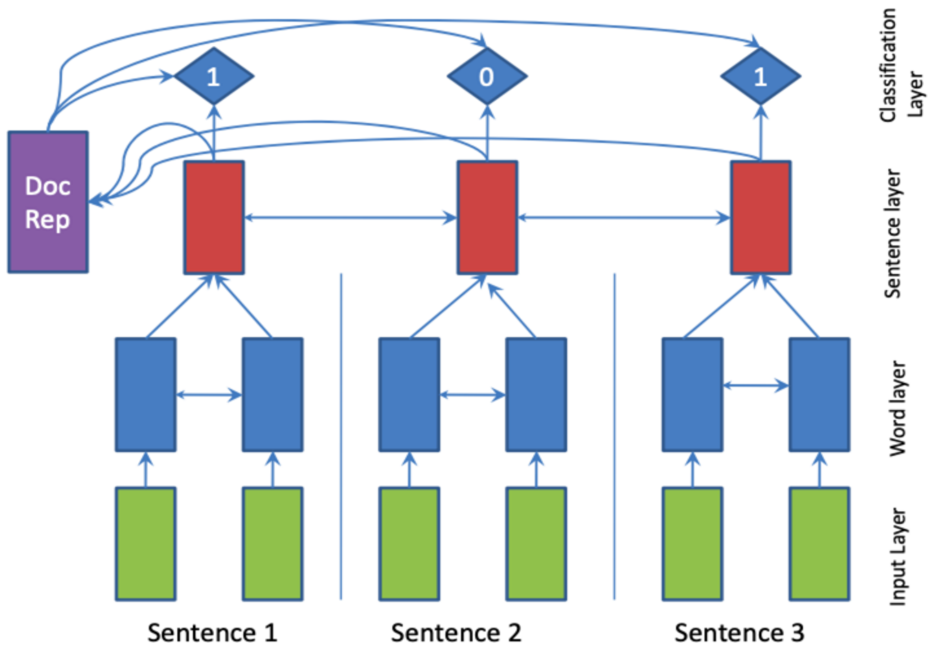
- Fasttext tried solving the morphology generation problem by BPE (byte pair encoding)
- Given “going”, divide the string into “go” and “ing”
- Use these parts to generate say “walking”
- Each subword will have its own probability
- If not subwording, then no way other than showing all forms of the root word: *go, went, going, gone*
- Languages differ in morphological complexity
- French more complex than English

Computation of Summaries

Hierarchical Encoder-Decoder



SummaRuNNer



[1]

Figure 1: SummaRuNNer: A two-layer RNN based sequence classifier

Summarization with Pointer-Generator Network

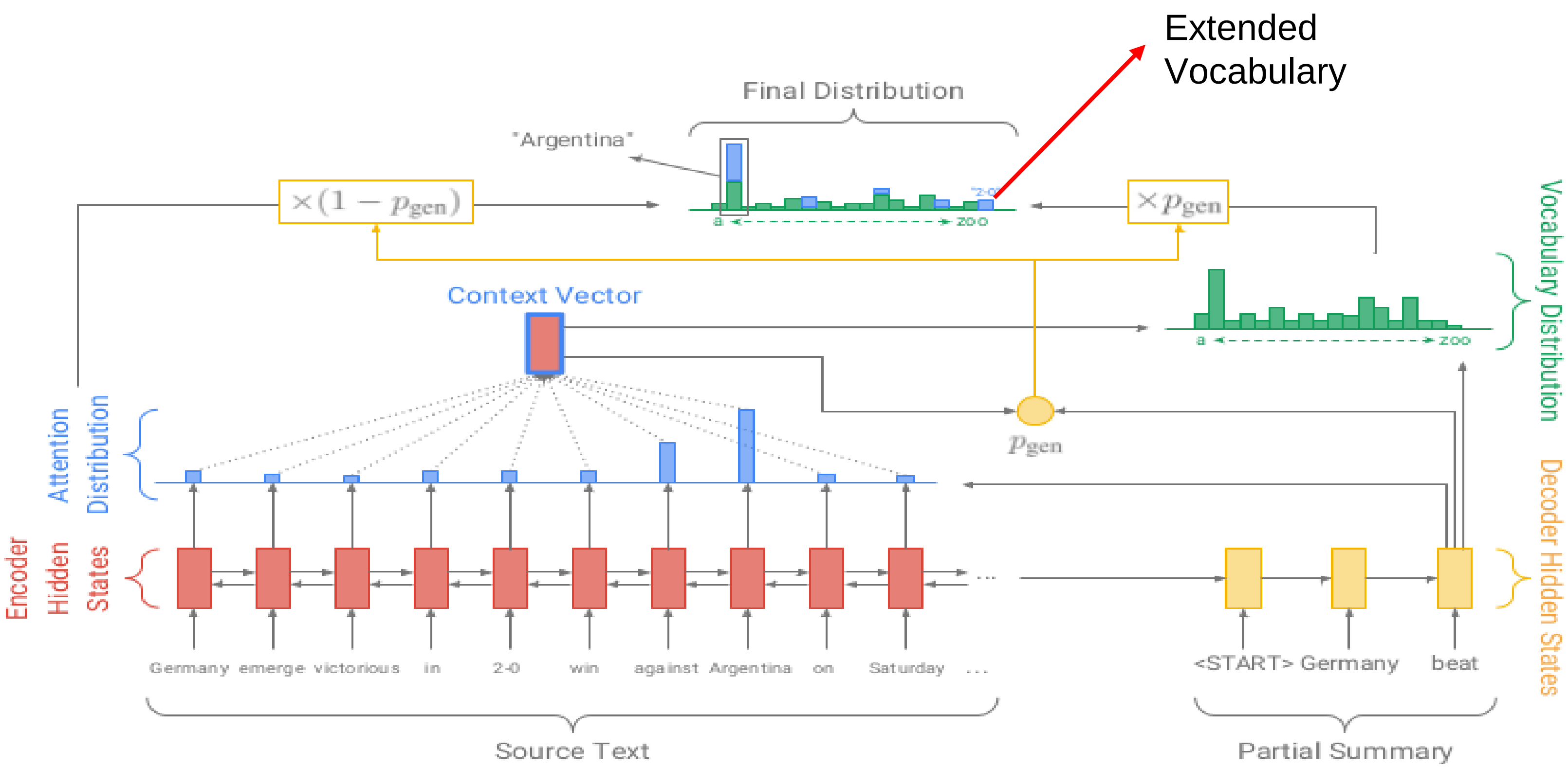


Figure: Pointer-generator Model [2]

BART

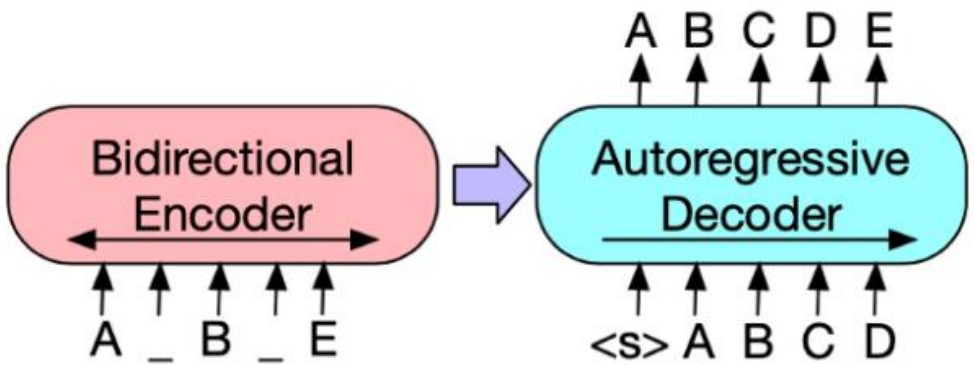


Fig 1: BART architecture.

- BERT (12 layers) + GPT (12 layers)
- Pre-trained on 160GB of news, books and web text
- Fine-tuned on CNN/DM dataset

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).

Now GPT…

1. Generative Pre-training
2. Supervised Fine Tuning
3. Reinforcement Learning with Human Feedback (RLHF)

Opinion/Review Summaries

Properties of Opinion Summaries

- **Monotonicity**: As more sentences are added to opinion summary, subjectivity increases along with information content

- **Diminishing Return:** If multiple sentences of varying intensity are added to opinion summary, the effect of lower intensity diminishes in presence of higher intensity bearing polar sentences

Examples from cricket: diminishing return

A: *Rahul Dravid is a great batsman*

B: *Rahul Dravid is a very consistent player*

$A \cup B$:

Rahul Dravid is a great batsman. He is a very consistent player

Compare *B* and $A \cup B$; "effect" of *B* diminished in presence of *A*

When asked to summarize $A \cup B$ *in one sentence, B is likely to be dropped*

Example from cricket: coverage

A: *"Sachin is a great batsman"*
B: "His backfoot batting is unmatchable"
C: "He also bowls decent spin"

- If the budget allows only two subjective sentences, then picking up *A* and *B* have captured only batting
- Picking up *C* with the one of *A* and *B* would have covered both aspects (i.e. batting and bowling)
- Sentences are not overlapping in aspects, hence no diminishing return
- Higher intensity dominates

Submodular Function (1/2)

Finite set V

Set Function $F: 2^V \rightarrow R$, $F(\varphi) = 0$

Definition: $F : 2^V \rightarrow R$ is submodular iff

$$\forall A, B \subset V, F(A) + F(B) \geq F(A \cap B) + F(A \cup B)$$

Submodular Function (2/2)

Equivalent definition:

$\forall k \in V,\ \forall A \subset V,$

$F(A \cup \{k\}) - F(A)$ is non-increasing

(diminishing return)

$$\Leftrightarrow$$

$\forall A \subset B,\ \forall k \notin A,$

$$F(A \cup \{k\}) - F(A) \geq F(B \cup \{k\}) - F(B)$$

Example of Submodular Functions: *Cut Functions, Set Cover*

Extractive Summarization and Submodularity

- Find a set $S \subseteq V$
- S is set of sentences in summary, V is set of sentences in Document
- which maximizes a submodular function $f(S)$ subject to budget constraints.

Monotone Submodular Objective

$$F(S) = L(S) + \lambda R(S)$$

$F(S)$ -> Total Utility of summary

$L(S)$ -> Relevance

$R(S)$ -> Diversity

Pointer Generator Network

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks, ACL.

Abstract (1/2)

- Proposes a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways.
- First: uses a hybrid pointer-generator network that can copy words from the source text via *pointing*, which aids accurate reproduction of information, while retaining theability to produce novel words through the *generator*

Abstract (2/2)

- Second: uses *coverage* to keep track of what has been summarized, which discourages repetition
- Applies the model to the *CNN/Daily Mail* summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points

Basic seq2seq n/w

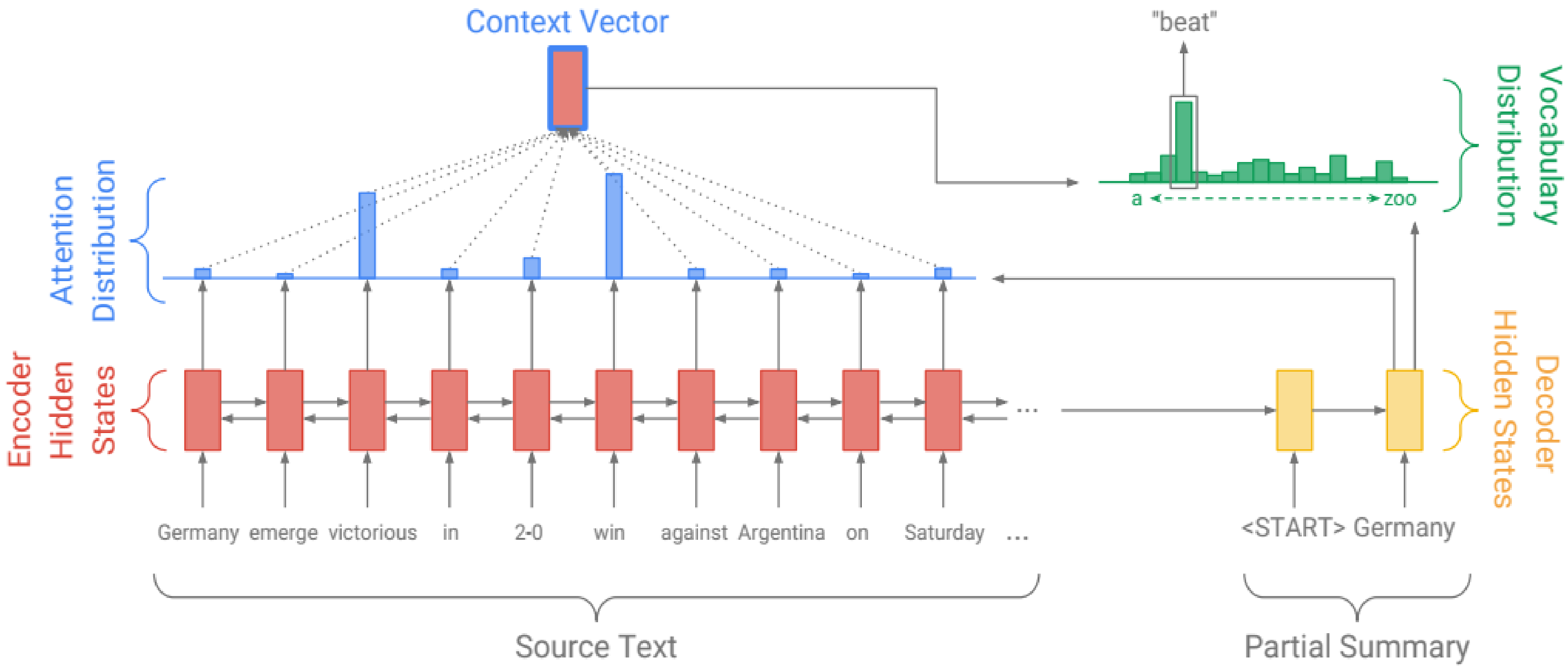


Figure 2: Baseline sequence-to-sequence model with attention. The model may attend to relevant words in the source text to generate novel words, e.g., to produce the novel word *beat* in the abstractive summary *Germany* ***beat*** *Argentina 2-0* the model may attend to the words *victorious* and *win* in the source text.

Pointer Generator N/W: copy word *vs.* new word

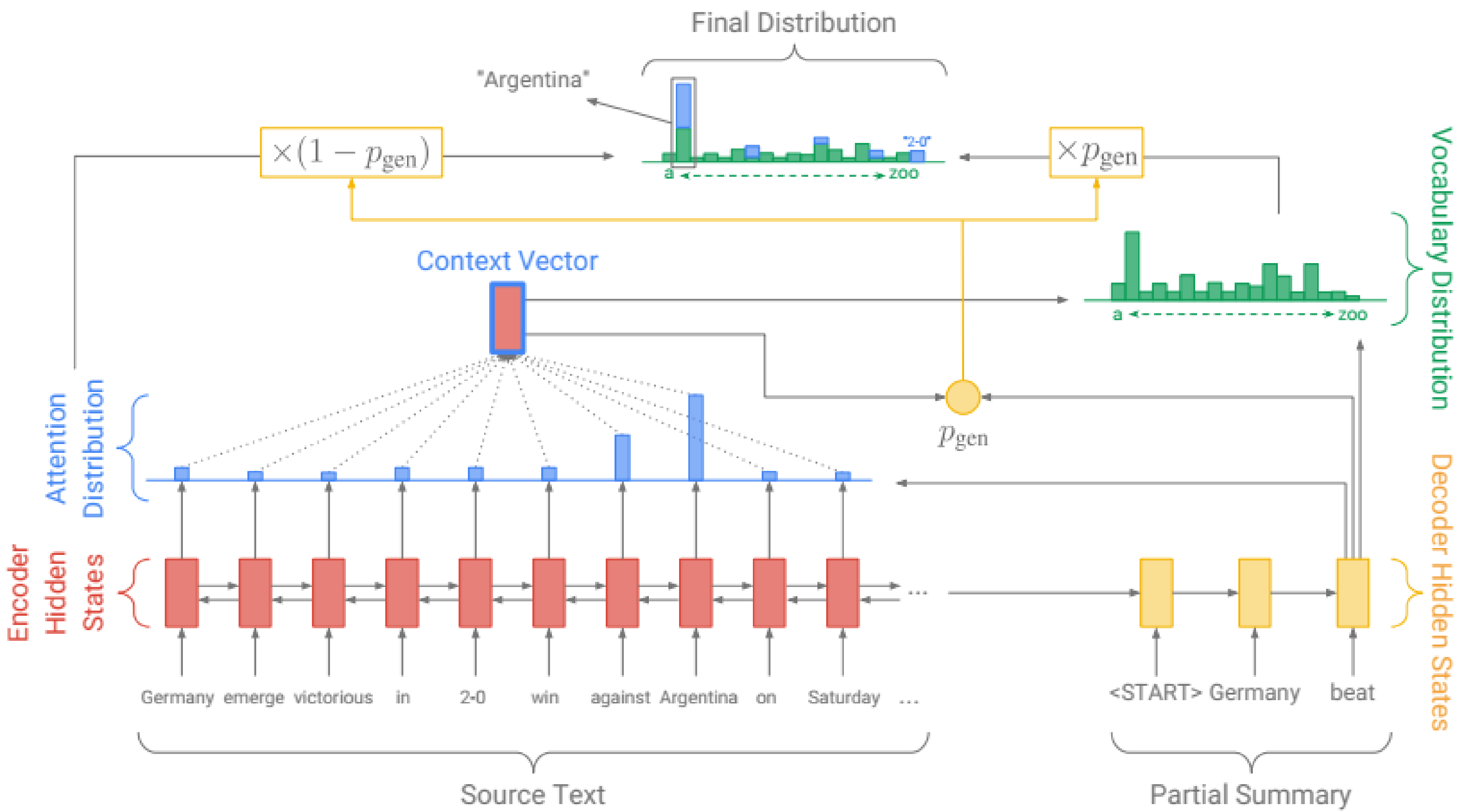


Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0, 1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article

Modeling: input processing

- Tokens w_i fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of *encoder hidden states* h_i

- At each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word

Modeling: encoder hidden states

- While training, this is the previous word of the reference summary;
- at test time it is the previous word emitted by the decoder), and has *decoder state* s_t.

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \qquad (1)$$

$$a^t = \text{softmax}(e^t) \qquad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters.

Modeling: encoder hidden states

- Attention is a probability distribution over the source words, that tells the decoder where to look to produce the next word.
- Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the *context vector* h_t^*

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

Modeling: vocab distribution

- The context vector is a fixed size representation of what has been read from the source for this step
- It is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \tag{4}$$

where V , V_0, b and b_0 are learnable parameters.

Modeling: probability distribution over vocab

- *Pvocab* is a probability distribution over all words in the vocabulary, and provides the final distribution from which to predict words *w*.

$$P(w) = P_{\text{vocab}}(w) \tag{5}$$

Modeling: Loss

- During training, the loss for timestep *t* is the negative log likelihood of the target word $w_t{}^*$ for that time step
- Overall loss for the whole sequence is:

$$\text{loss}_t = -\log P(w_t^*) \qquad (6)$$

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T} \text{loss}_t \qquad (7)$$

Modeling: Pointer Generator

- Allows both copying words via pointing, and generating words from a fixed vocabulary.
- *Generation probability* $p_{gen} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \quad (8)$$

Pointer Generator N/W: copy word *vs.* new word

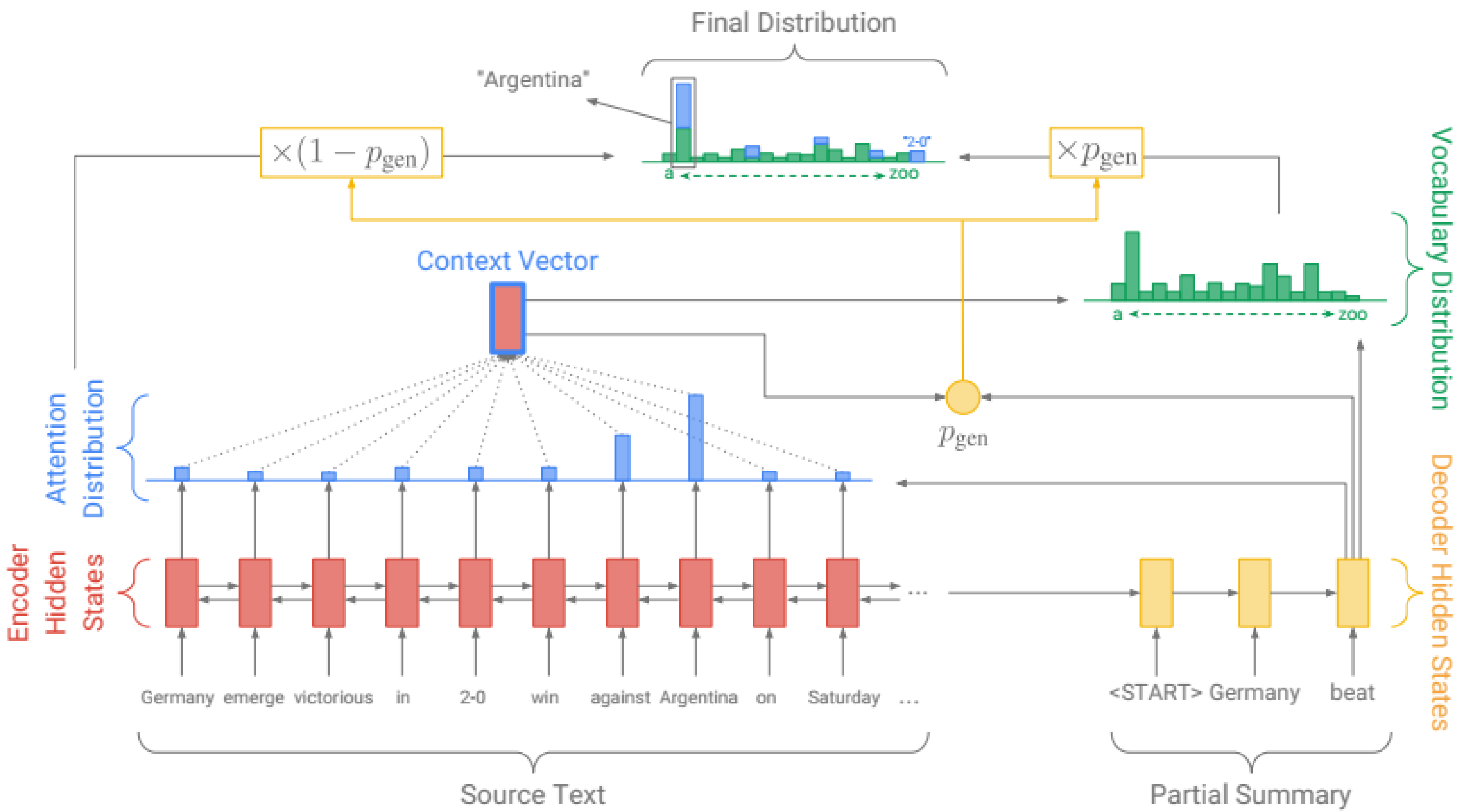


Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0, 1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article

Modeling: Generator Probability

- Allows both copying words via pointing, and generating words from a fixed vocabulary.
- *Generation probability* $p_{gen} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \qquad (8)$$

where vectors w_h^*, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function.

Modeling: to point or to generate

- p_{gen} is used as a soft switch to choose between *generating* a word from the vocabulary by sampling from P_{vocab}, or *copying* a word from the input sequence by sampling from the attention distribution a_t

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t \quad (9)$$

- if *w* is an out-of-vocabulary (OOV) word, then *P*vocab(*w*) is zero; similarly if *w* does not appear in the source document, then $\sum_{i:w_i=w} a_t^i$ is zero.

Result: superiority of pointer-generator

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/syn/para
abstractive model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extractive model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

Table 1: ROUGE F_1 and METEOR scores on the test set. Models and baselines in the top half are abstractive, while those in the bottom half are extractive. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script. The METEOR improvement from the 50k baseline to the pointer-generator model, and from the pointer-generator to the pointer-generator+coverage model, were both found to be statistically significant using an approximate randomization test with $p < 0.01$.

Giving importance to Recall: Ref n-grams: ROUGE

ROUGE

- **R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
- ROUGE is a package of metrics: ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S

ROUGE-N

$$\text{ROUGE-N} = \frac{\sum_{S\in\{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\boxed{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}}$$

$$p_n = \frac{\sum_{C\in\{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\boxed{\sum_{C'\in\{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}}$$

ROUGE-N incorporates Recall

Will BLEU be able to understand quality of long sentences?

Reference translation:
क्या ब्लू लंबे वाक्य की गुणवत्ता को समझ पाएगा?
Kya bloo lambe waakya ki guNvatta ko samajh paaega?

Candidate translation:
लंबे वाक्य
Lambe vaakya

ROUGE-N: 1 / 8
Modified n-gram Precision: 1

Other ROUGEs

- ROUGE-L
  - Considers longest common subsequence
- ROUGE-W
  - Weighted ROUGE-L: All common subsequences are considered with weight based on length
- ROUGE-S
  - Precision/Recall by matching skip bigrams

ROUGE v/s BLEU

	ROUGE	BLEU
Handling incorrect words	Skip bigrams, ROUGE-N	N-gram mismatch
Handling incorrect word order	Longest common sub-sequence	N-gram mismatch
Handling recall	ROUGE-N incorporates missing words	Precision cannot detect ‘missing’ words. Hence, brevity penalty!

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n \right)$$

Thank you

http://www.cse.iitb.ac.in/~pb

http://www.cfilt.iitb.ac.in