SEQUENCE MODELS

BY:SATISH DESHBHRATAR

SEQUENCE DATA

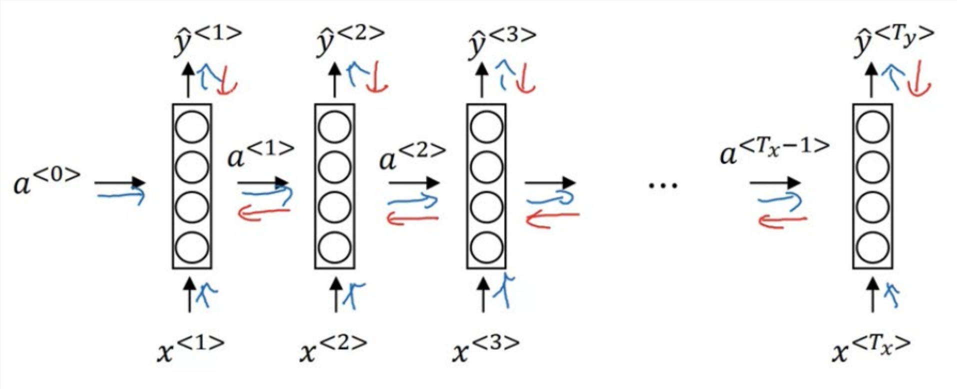
- **Examples**: Speech recognition, music generation, sentiment classification, DNA sequence analysis, machine translation, video activity recognition, name entity recognition
- **Representation**:

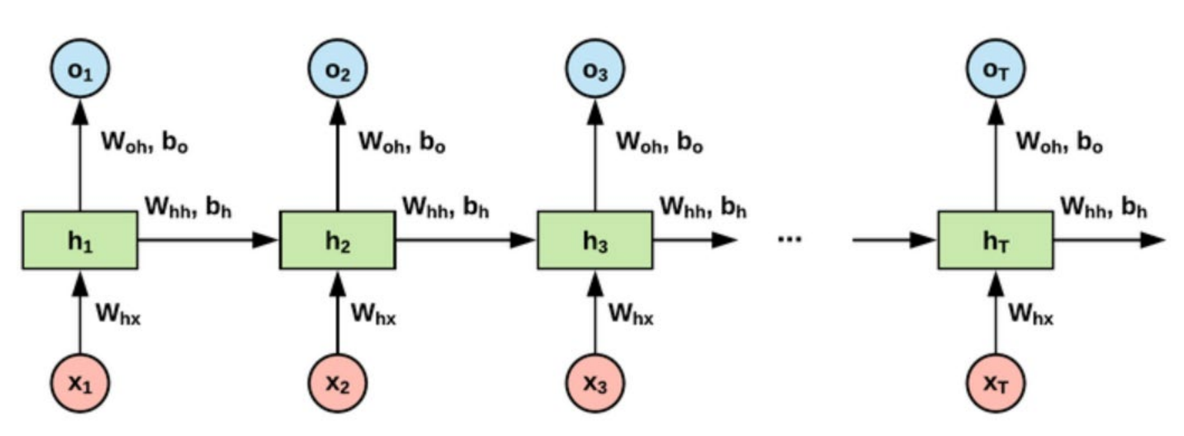
$$X^{(i)<t>}: t^{th} word\ in\ i^{th} training\ example$$
$$Y^{(i)<t>}: t^{th} label\ for\ i^{th} training\ example$$
$$T_x^{(i)}: length\ of\ i^{th} training\ example$$
$$T_y^{(i)}: length\ of\ label\ for\ i^{th} training\ example$$

$$\boldsymbol{Vocabulary} = \begin{bmatrix} a \\ aaron \\ \vdots \\ harry \\ \vdots \\ zulu \end{bmatrix} \in (10000,1); \boldsymbol{One - hot\ encode}:\ x^{<t>} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \in (10000,1)$$

- **Problems with a standard network**:
  - Inputs, outputs can be ≠ lengths in ≠ examples
  - Doesn't share features learned across ≠ positions of text

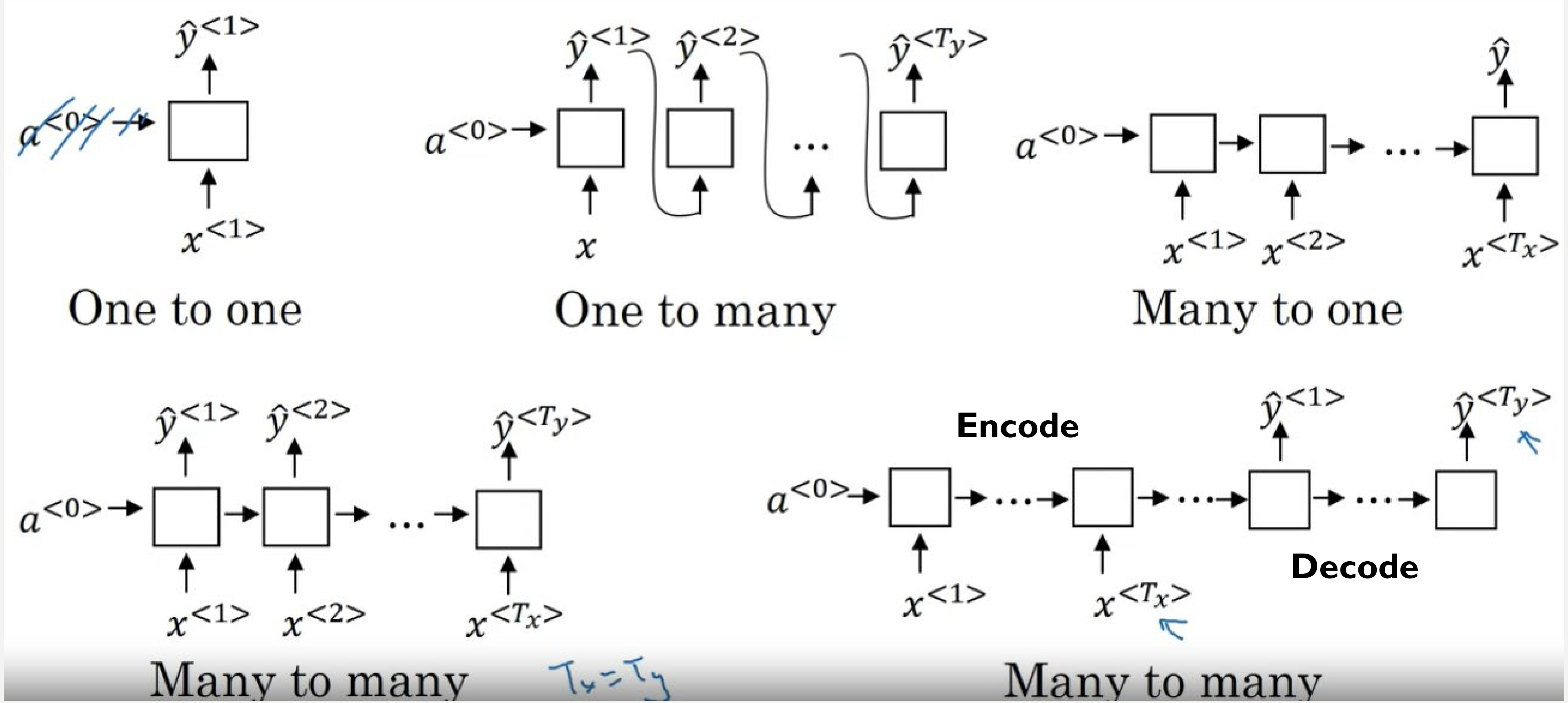
RECURRENT NEURAL NETWORKS





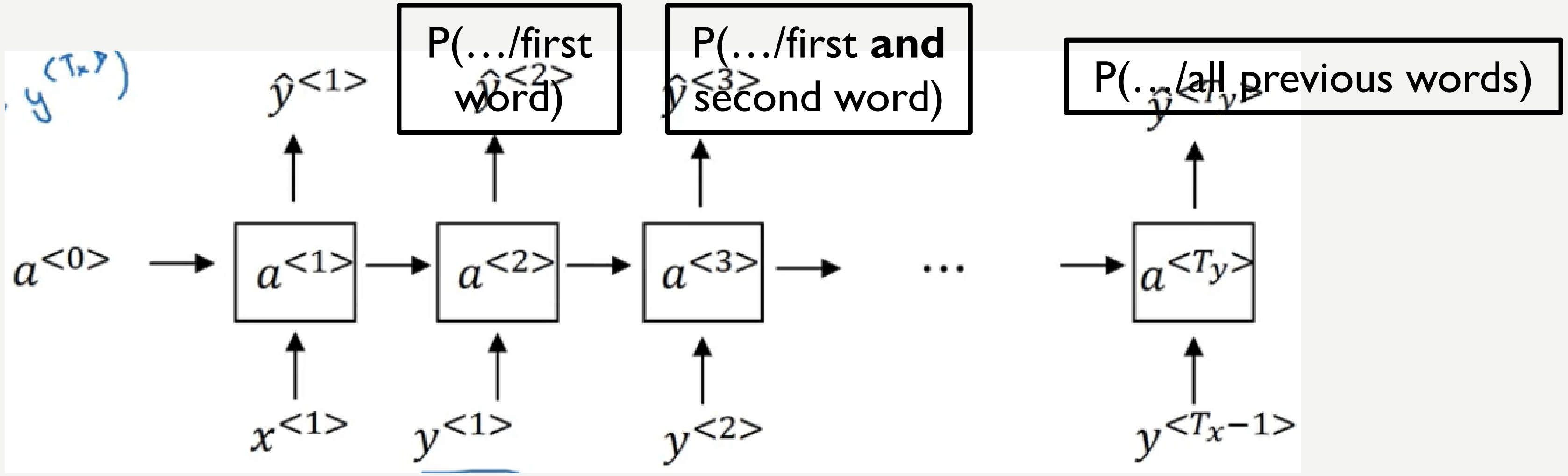
- **<u>Forward propagation</u>**:
  - $a^{<t>} = tanh\left(\boldsymbol{W_{aa}a}^{<t-1>} + \boldsymbol{W_{ax}x}^{<t>} + b_a\right) = \tanh\left(\boldsymbol{W_a}\left[\boldsymbol{a}^{<t-1>}, \boldsymbol{x}^{<t>}\right] + b_a\right)$
    - $W_a = [W_{aa} \mid W_{ax}] \in (\#examples, \#examples + \#vocabulary)$
    - $[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \in (\#examples + \#vocabulary, 1)$
    - $a^{<0>} = 0$
  - $\hat{y}^{<t>} = sigmoid\left(W_{ya}a^{<t>} + \mathrm{b_y}\right) = sigmoid\left(W_y a^{<t>} + \mathrm{b_y}\right)$
- $L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} log\hat{y}^{<t>} - (1 - y^{<t>}) \log(1 - \hat{y}^{<t>})$
- $\boldsymbol{Backpropagation\ through\ time}$: $L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$

RNN TYPES


$\hat{y}^{<1>}$
$a^{<0>}$
$x^{<1>}$
One to one
$\hat{y}^{<1>}$ $\hat{y}^{<2>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
x
One to many
$\hat{y}$
$a^{<0>}$
$x^{<1>}$ $x^{<2>}$ $x^{<T_x>}$
Many to one
$\hat{y}^{<1>}$ $\hat{y}^{<2>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
$x^{<1>}$ $x^{<2>}$ $x^{<T_x>}$
Many to many
$T_x = T_y$
Encode
$\hat{y}^{<1>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
$x^{<1>}$ $x^{<T_x>}$
Decode
Many to many

LANGUAGE MODELING

- **<u>Training set</u>**: Large corpus (data set) of English text
  - <EOS> = Word for End Of Sentence
  - <UNK> = Word for words that are not in the vocabulary



- $L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} log\hat{y}_i^{<t>}$
- $P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>})P(y^{<2>}/y^{<1>})P(y^{<3>}/y^{<1>}, y^{<2>})$
- **<u>Sampling a sequence from a trained RNN</u>**: Start with a random word, and generate next words based on probabilities
- **<u>PS</u>**: There is also **character-level language model**, that has a vocabulary of characters individually
- **<u>PS</u>**: Problem of **vanishing gradients / Exploding gradients:** No possibility to remember past entries

GRU & LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[\boxed{\Gamma_r} * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\boxed{\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Adding *orange* → <u>Full GRU</u>
c: Memory cell
Γ_u: Update gate
Γ_o: Output gate
Γ_f: Forget gate

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

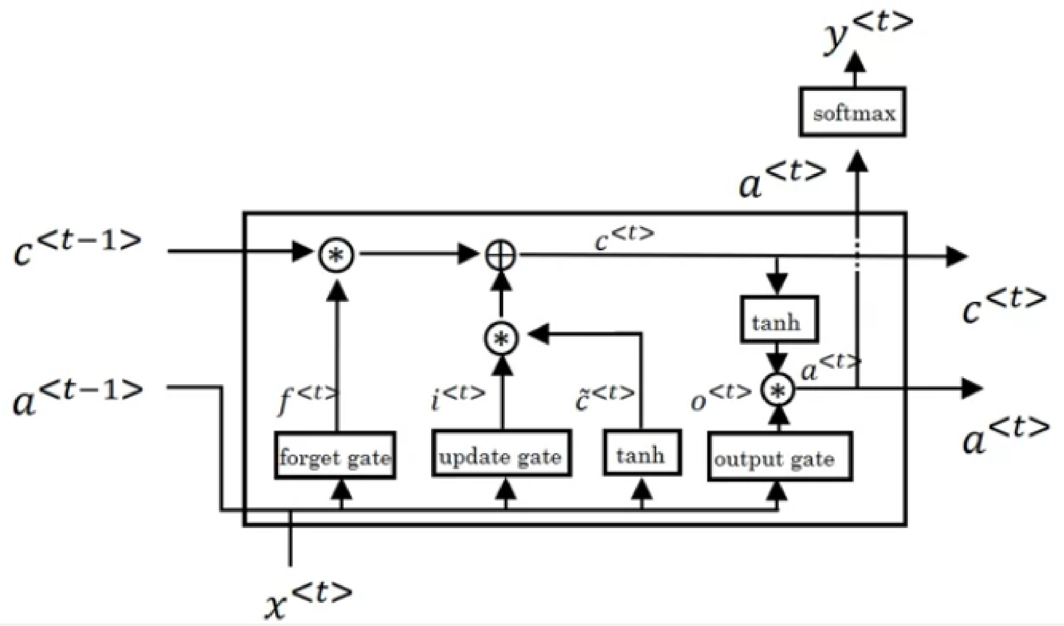
$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

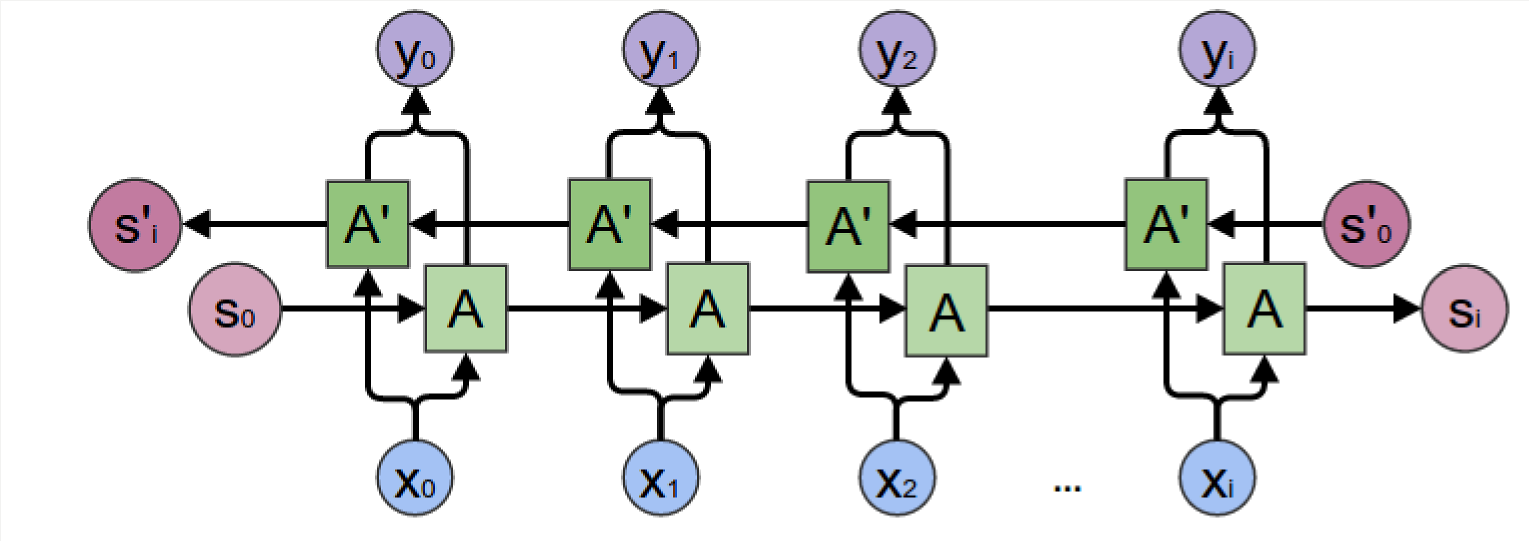
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



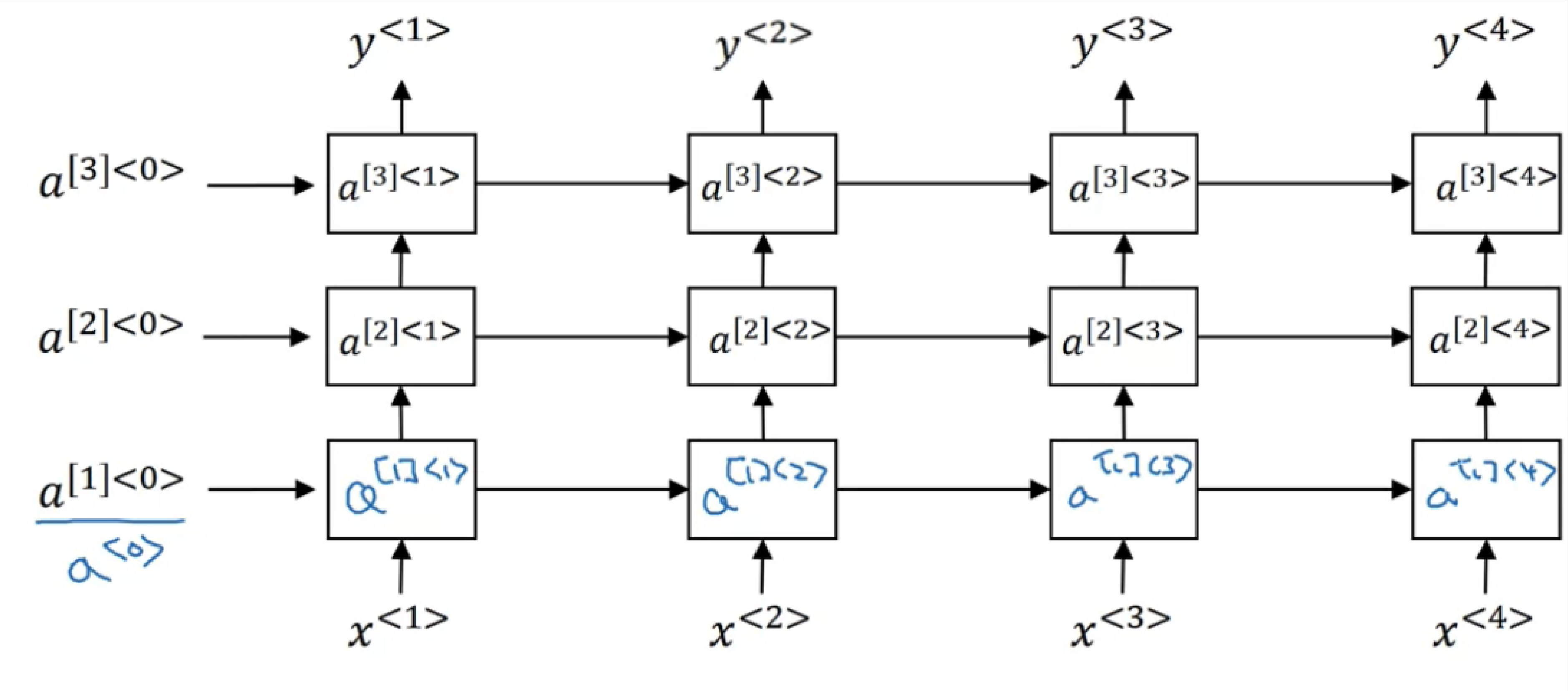
BIDIRECTIONAL RNN

- **<u>Goal:</u>** Learn from the future and the past → Acyclic graph
- $\hat{y}^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>} + b_y]$



DEEP RNN

- $a^{[l]<t>} = g(W_a^{[l]}\left[a^{[l]<t-1>}, a^{[l-1]<t>} + b_a^{[l]}\right]$



WORD EMBEDDING

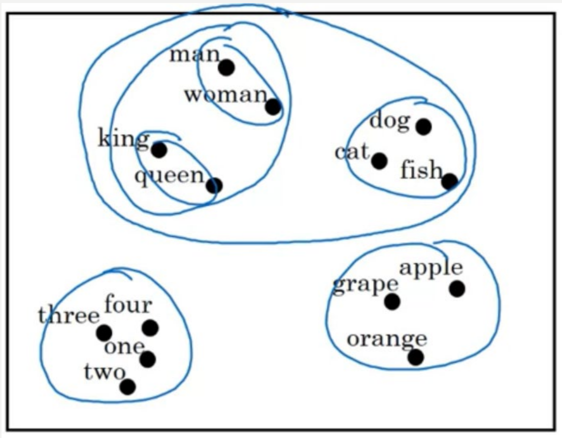
- The problem of word representation (one hot vectors) is that words are not linked to eachother

e_{word}

Features/Word	King	Queen	Apple	Orange
Gender	-1	1	0	0
Food	0	0	1	1
…	…	…	…	…

→Learning « orange **juice** » will lead to learning « apple **juice** » because orange and apple are linked

- **t-SNE algorithm**: #Features dimension → 2D:Visualize word embedding



- **Transfer learning**:
  1. Learn word embeddings from large text corpus (1-100B words)

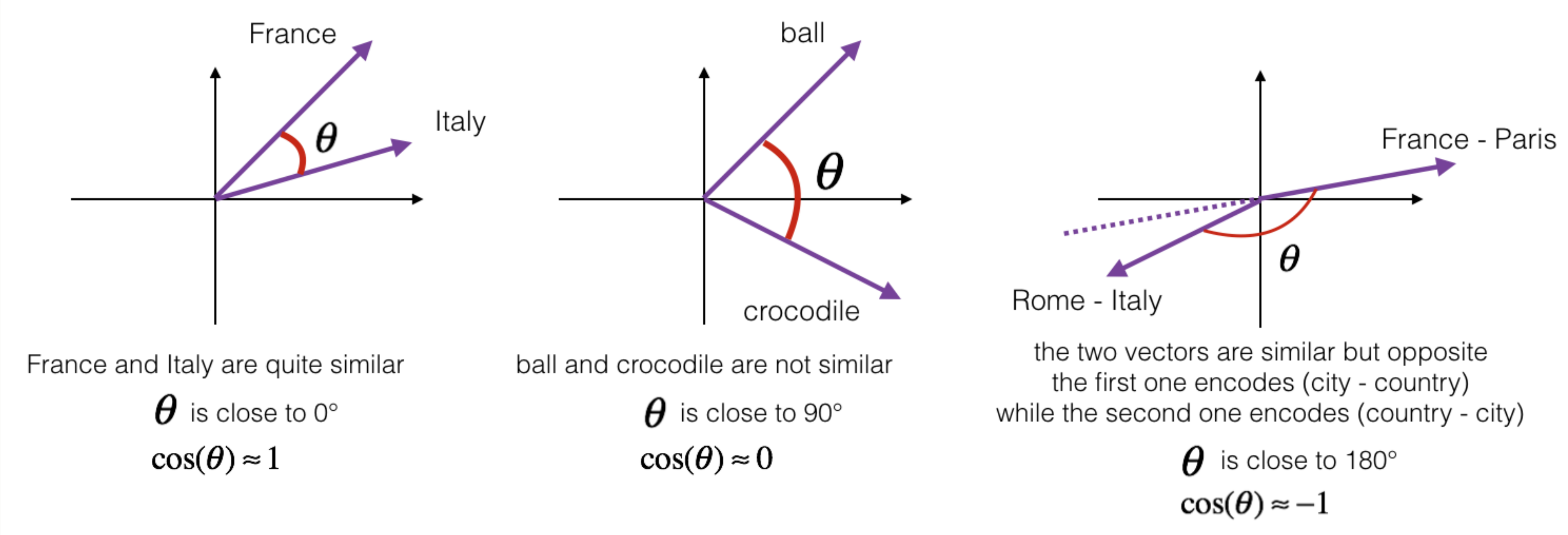
     (Or download pre-trained embedding online)
  2. Transfer embedding to new task with smaller training set (100k words)
  3. Optional: Continue to finetune the word embeddings with new data
- **PS:** Word embedding in RNN = Face encoding in CNN

SIMILARITY

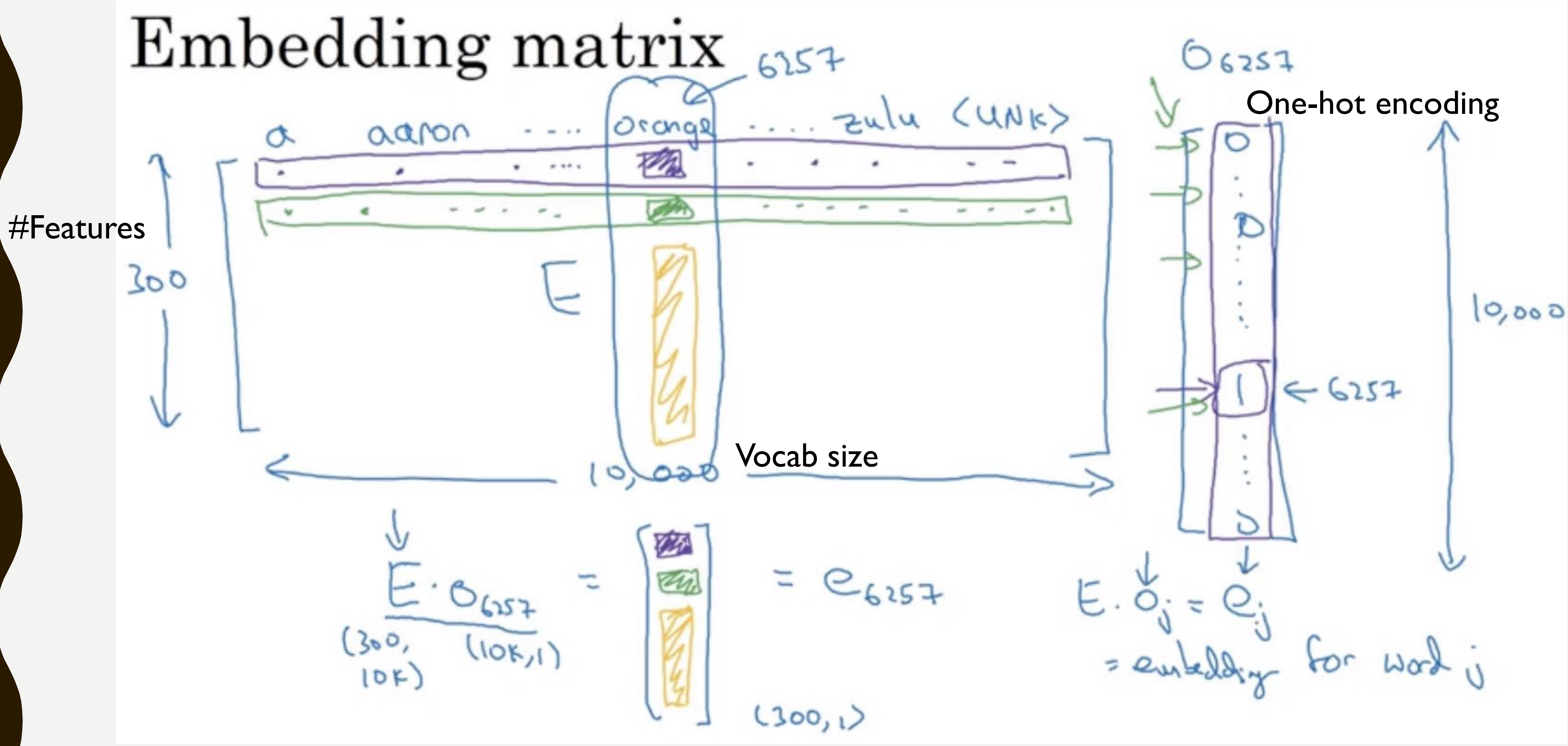
– Man to woman as king to ____?

– $e_{man} - e_{woman} = e_{king} - e_{?} \Rightarrow \textbf{arg max similarity}(\boldsymbol{e_{?}}, \boldsymbol{e_{king}} - \boldsymbol{e_{man}} + \boldsymbol{e_{woman}})$

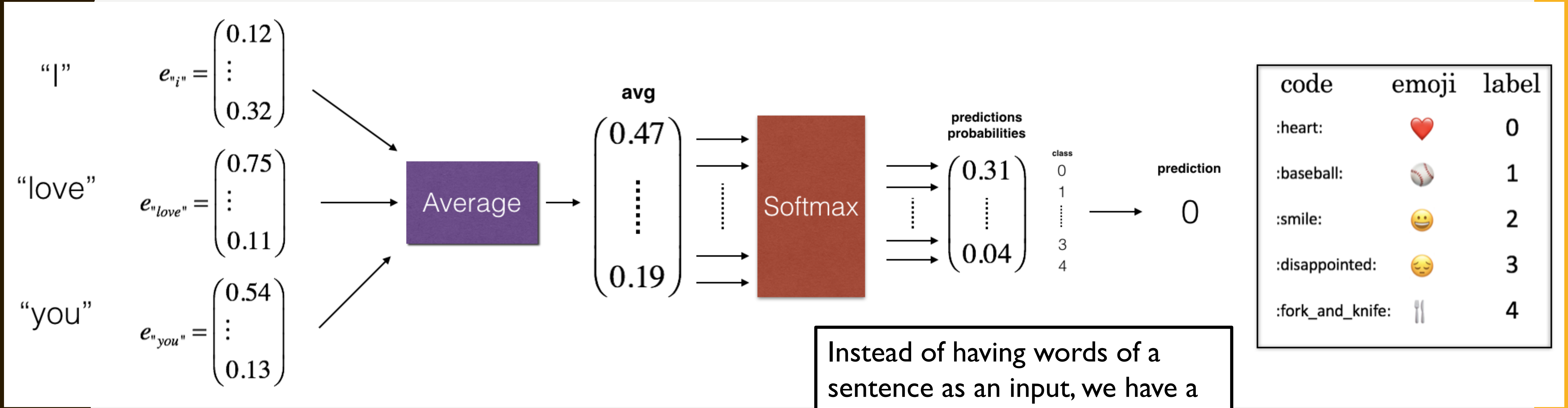
– Exp: $\boldsymbol{Cosine\ similarity}(\boldsymbol{u}, \boldsymbol{v}) = \frac{u^T v}{||u||_2 ||v||_2}$



EMBEDDING MATRIX



NATURAL LANGUAGE MODEL (1)



Instead of having words of a sentence as an input, we have a list of possible target words

- **Context/trager pairs**: Conext can be:
  - Last 4 words
  - 4 words on left & right
  - Last I word
  - Nearby I word
- PS: **Natual language model** works like **sentiment classification model**
- PS2: We can use **RNN many-to-one** as seniment classification model

NATURAL LANGUAGE MODEL (2)

- Softmax: $p(t/c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10000} e^{\theta_j^T e_c}}$; θ_t=$parameter\ associated\ with\ target\ t$
  e_c=$Embedding\ of\ context\ c$

➔ **Problem**: Computationally expensive → Hierarchical softmax (Not bad as a solution)

- **Solution**: Defining a new learning problem
  - Input = Couple (Context, Word), context being **static**
  - Output = $1\ if\ \boldsymbol{word}\ is\ \boldsymbol{target}$
    $0\ if\ not$
  - Training set size: 5-20 words for small data set and 2-5 words for large data set

  → $P(y = 1/c, t) = \sigma'(\theta_t^T e_c) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=1}^{10000} f(w_j)^{\frac{3}{4}}}$: 10000 binary classification problem

  - **PS**: ¾ to avoid negative examples (the, of, and…) and to consider positive rare examples (orange, apple…)

- **GloVe (Global Vectors for word representation)**:
  - $X_{ij} = times\ i\ appears\ in\ context\ of\ j\ (i = conext, j = target)$
  - Minimize $\sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i + b_j' - log X_{ij})^2$; $f(0) = 0$

BIAS IN WORD EMBEDDING

- PS: Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model

1. Identify bias direction

   $e_{he} - e_{she}$

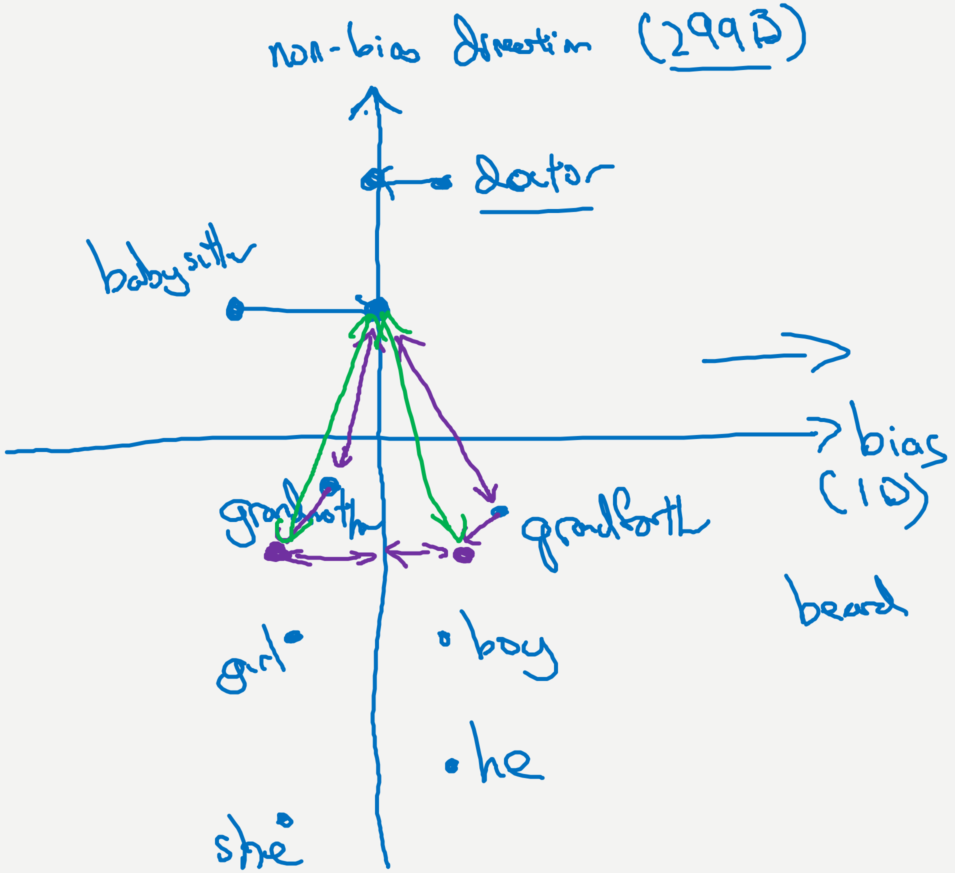
   $e_{male} - e_{female}$

   → Average

2. Neutralize: For every word that is not definitional, project to get rid of bias

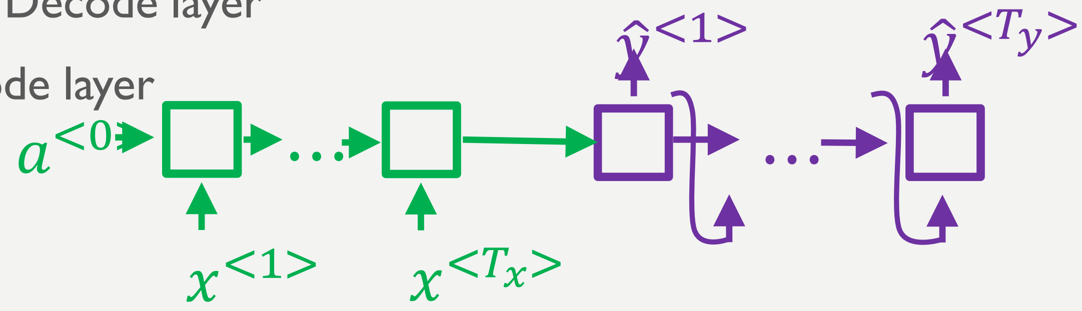
3. Equalize pairs

   $grandmother\ girl - grandfather\ boy$



SEQUENCE TO SEQUENCE MODEL

- **Text translation**: Encode layer → Final feature vector → Decode layer
- **Image captioning**: CNN → Final feature vector → Decode layer
- **PS**: Machine translation = Conditional language model:



➔ $\arg\max P\left(y^{<1>}, \dots, y^{<T_y>} \middle| x^{<1>}, \dots, x^{<T_x>}\right)$

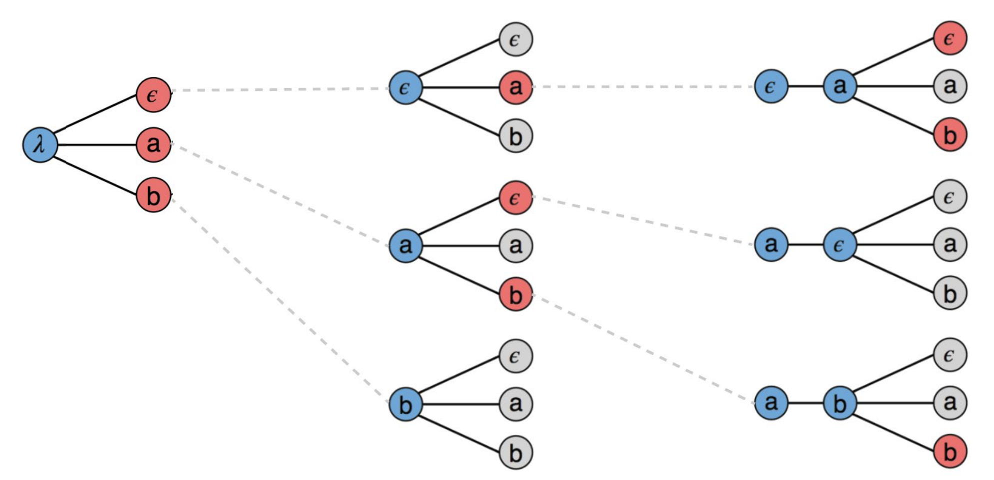
- **Greedy search**: $P\left(y^{<1>}, \dots, y^{<T_y>} \middle| x\right)$:

  Predict each word on itself, and go on…

- **Beam search**: Choose the best 3 possible words

  (maximum probabilities), and go on…

  - Beam width = Maximum number of words to select
    - Large B: Better result / Slower | Small B:Worse result / Fastera
  - Error analysis:
    - If $P(y_{human}|x) > P\left(y_{algorithm}\middle|x\right)$ → Beam search at fault
    - If $P(y_{human}|x) \leq P\left(y_{algorithm}\middle|x\right)$ → RNN at fault



$$\arg\max_{y} \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>}) \Leftrightarrow \arg\max_{y} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$$

BLEU SCORE

- BLEU = Bilingual Evaluation Understanding
  - p1 = Performance by choosing a unigram
  - p2 = Performance by choosing n-unigram

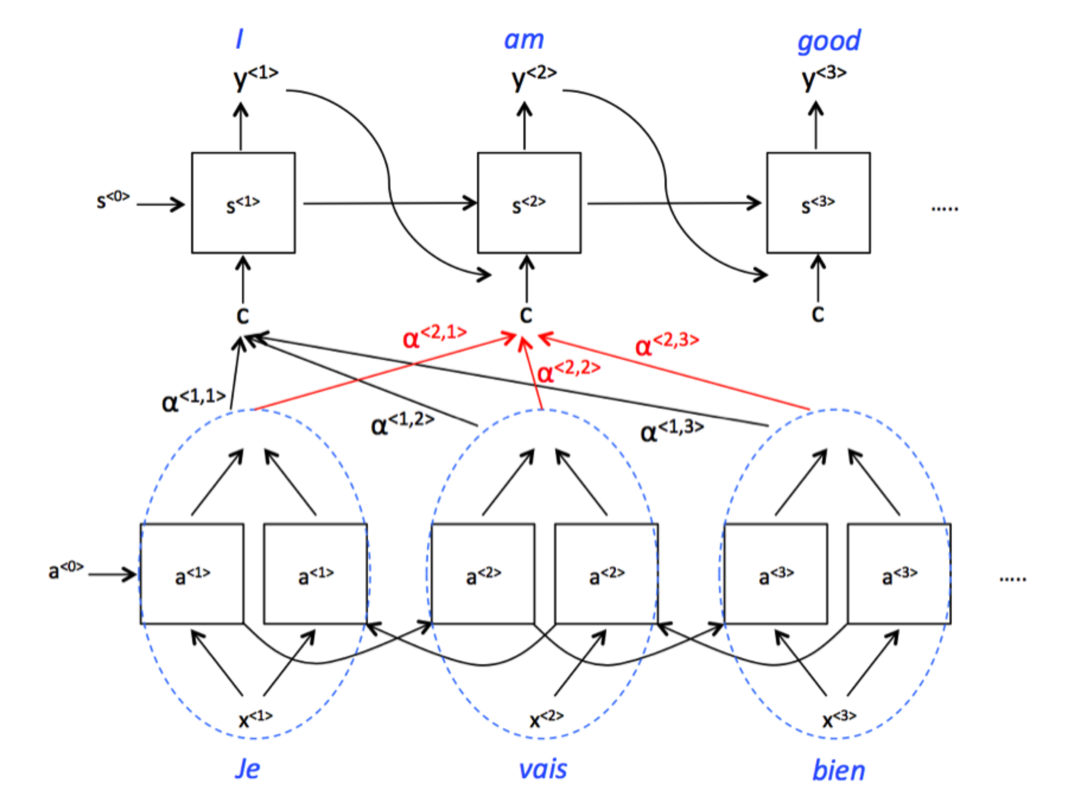
$$p_1 = \frac{\sum_{unigram \in \hat{y}} count_{clip}\,(unigram)}{\sum_{unigram \in \hat{y}} count\,(unigram)} \qquad p_n = \frac{\sum_{ngram \in \hat{y}} count_{clip}\,(ngram)}{\sum_{ngram \in \hat{y}} count\,(ngram)}$$

- Combine BLEU score: $BP.\, e^{\frac{1}{m}\sum_{n=1}^{m} p_n}$
  - With:

$$BP = \begin{cases} 1 & \text{if MT\_output\_length > reference\_output\_length} \\ \exp(1 - \text{MT\_output\_length/reference\_output\_length}) & \text{otherwise} \end{cases}$$

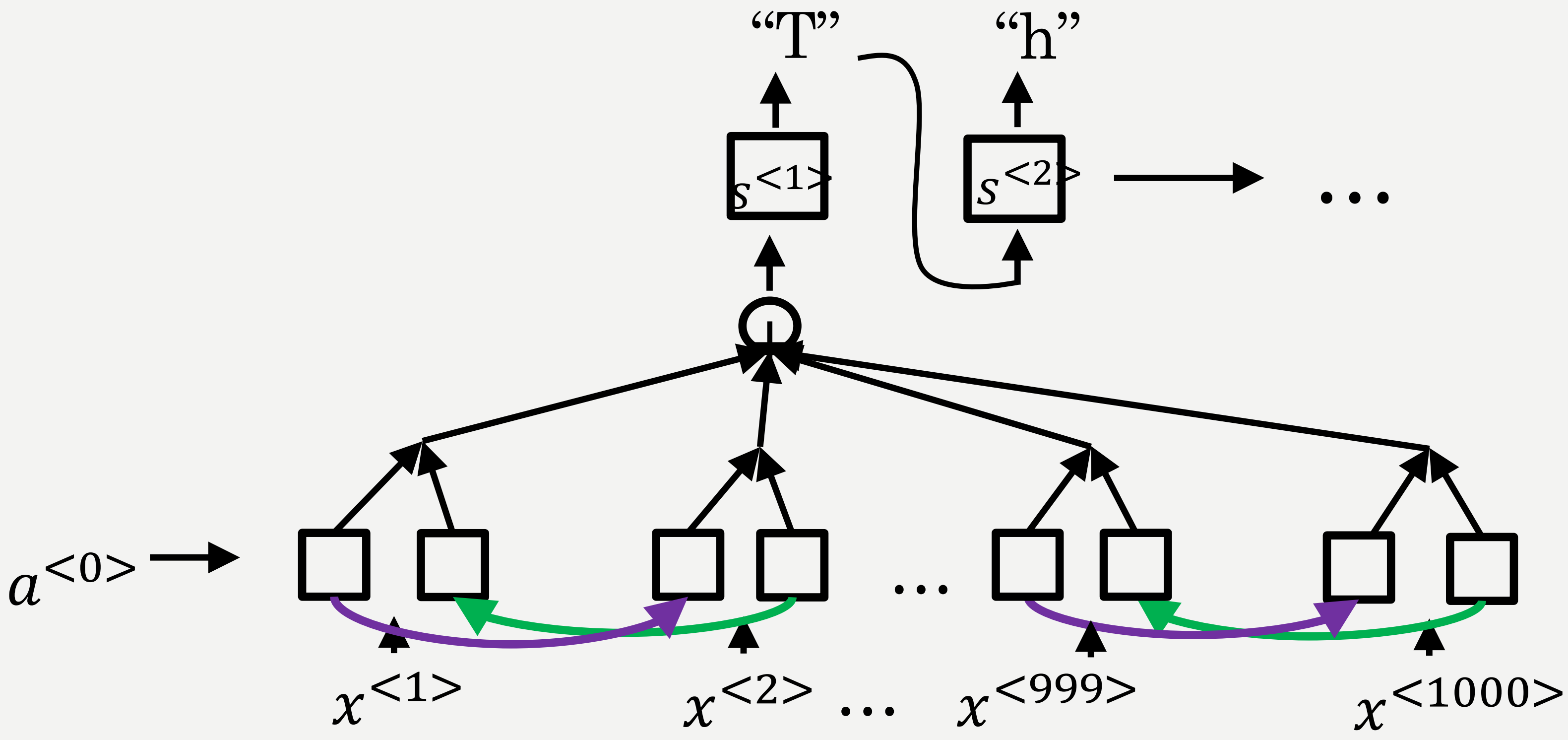
ATTENTION MODEL

- <u>**Goal**</u>: Resolve machine translation problem if a sentence is too long



- $\alpha^{<t,t'>} = \frac{e^{e^{<t,t'>}}}{\sum_{t'=1}^{T_x} e^{e^{<t,t'>}}}$ *Amount of "attention" $y^{<t>}$ should pay to $a^{<t'>}$*
- $c^{<t>} = \sum_{t'} \alpha^{<t,t'>} a^{<t'>}$; $a^{<t'>} = \left(\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>}\right)$
- $\sum_{t'} \alpha^{<t,t'>} = 1$

ATTENTION MODEL – SPEECH RECOGNITION



- CTC = Connectionist Temporal Classification
  - Basic rule: Collapse characters not seperated by « blank »
- PS: Trigger word detection algorithm → Put 1's seconds just after the trigger word is said