Domain Oriented Conversational Agent

Internal Guide – Dr. Gorthi R. K. S. S. Manyam
External Guide – Pavan Kumar Reddy Sannadi
Submitted by – Vamshi Kumar Kurva
SC15M045

Introduction

- Artificial conversational entity.
- Responds to domain-oriented user questions.
- Assists the user just like humans.
- A kind of customer care service.
- Technically, the task is to predict the next sentence given the context (previous sentences in the conversation)

Challenges in building such an agent:

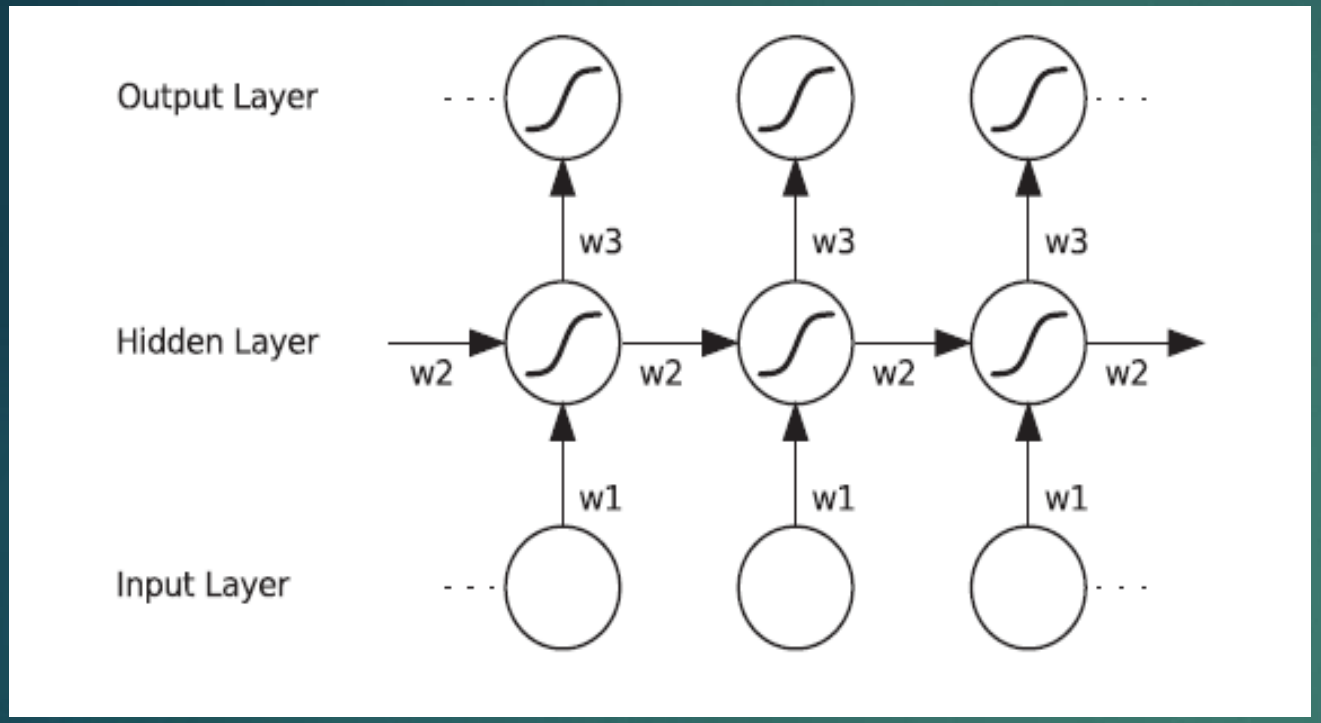
- Conversations have time-dependencies.
- Agent should be able to understand and reply even if the question asked is a bit different from the trained conversations
- Agent should be able to capture the long-term temporal dependencies.
- Agent has a limited vocabulary.

Literature survey

- NNs can learn to generalize given the huge dataset.
- DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality
- It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori.
- Question-answering, machine translation are sequential problems.
- Question-answering can also be seen as mapping a sequence of words representing the question to a sequence of words representing the answer

RNN

- RNN, in principle can map entire history of inputs to each output.
- RNN with a sufficient number of hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy
- RNNs allow a ‘memory’ of previous inputs to persist in the network’s internal state, thereby influence the network output.
- RNN can be thought of as a FFNN unfolded in time space
- Forward pass is same as that of a FFNN, except for the fact that activations arrive the hidden layer from both current external input and the hidden layer activations from the previous time step.



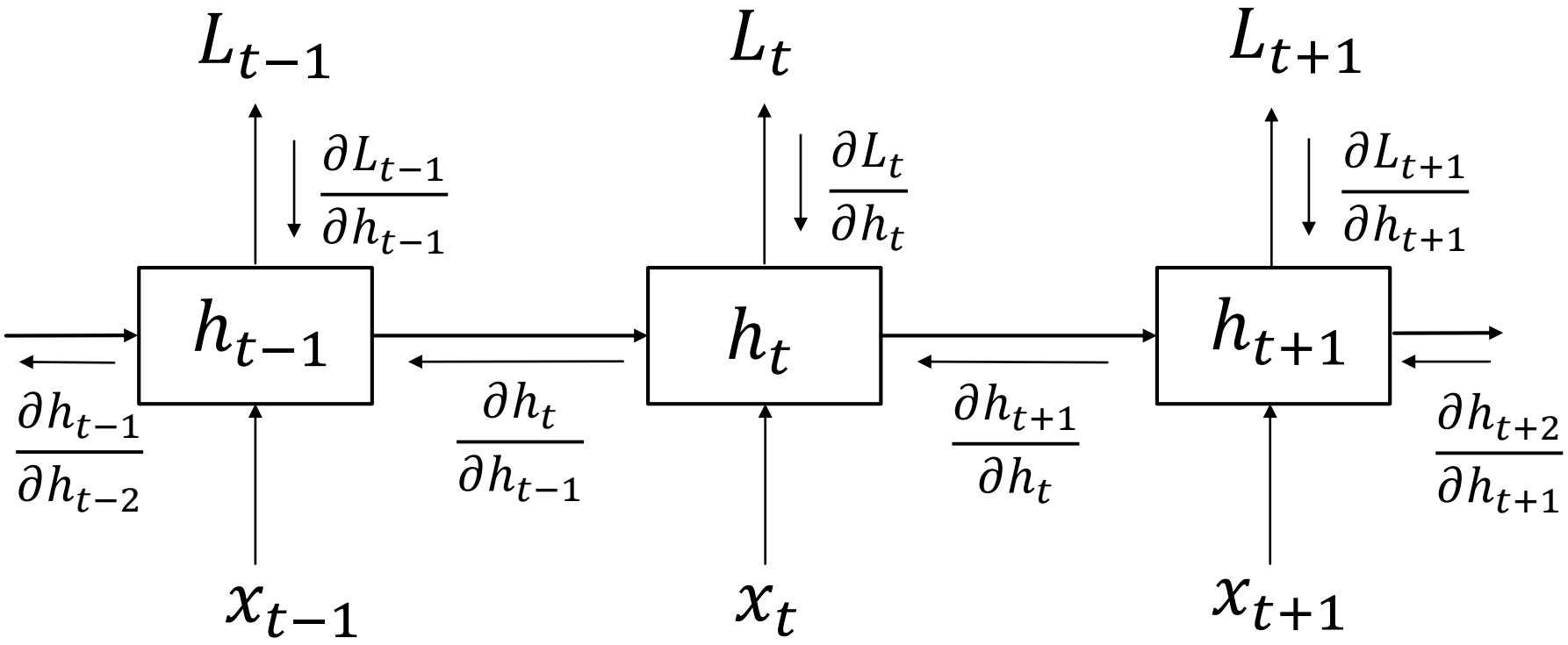
Let I --- i/p neurons, H --- hidden,
K --- o/p neurons
L ---- Loss function

$$\delta_j = \frac{\partial L}{\partial a_j}, \quad a_j = \sum_i w_{ij} b_i$$

Using Backpropagation

$$\delta_h = f'(a_h) \sum_{h' \epsilon H_{l+1}} \delta_{h'} \, w_{hh'}$$

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} = \delta_j \, b_i$$


L_{t-1}
L_t
L_{t+1}
$\frac{\partial L_{t-1}}{\partial h_{t-1}}$
$\frac{\partial L_t}{\partial h_t}$
$\frac{\partial L_{t+1}}{\partial h_{t+1}}$
h_{t-1}
h_t
h_{t+1}
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
$\frac{\partial h_{t+1}}{\partial h_t}$
$\frac{\partial h_{t+2}}{\partial h_{t+1}}$
x_{t-1}
x_t
x_{t+1}

- Let W_{IH} ---- input to hidden weight matrix

  W_{HH} ---- hidden to hidden weight matrix

  x_t ---- input vector at time ‘t’, then

  $h_t = W_{IH}\, x_t + W_{HH}\, h_{t-1}$ ----------(1)

- if T is the length of input sequence, then by applying (1) repeatedly from t=1 to T, we can get the hidden state activations ($h_0 = 0$)

Backward pass:

- Using BPTT, start at t = T till t=1

$$\frac{\partial L}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial L}{\partial a_j{}^t} \frac{\partial a_j{}^t}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j{}^t \; b_i{}^t$$

$$\delta_h^t = \theta'(a_h^t)\left(\sum_{k=1}^{K} \delta_k^t w_{hk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} w_{hh'}\right)$$

$$\delta_j^t \stackrel{\text{def}}{=} \frac{\partial \mathcal{L}}{\partial a_j^t}$$

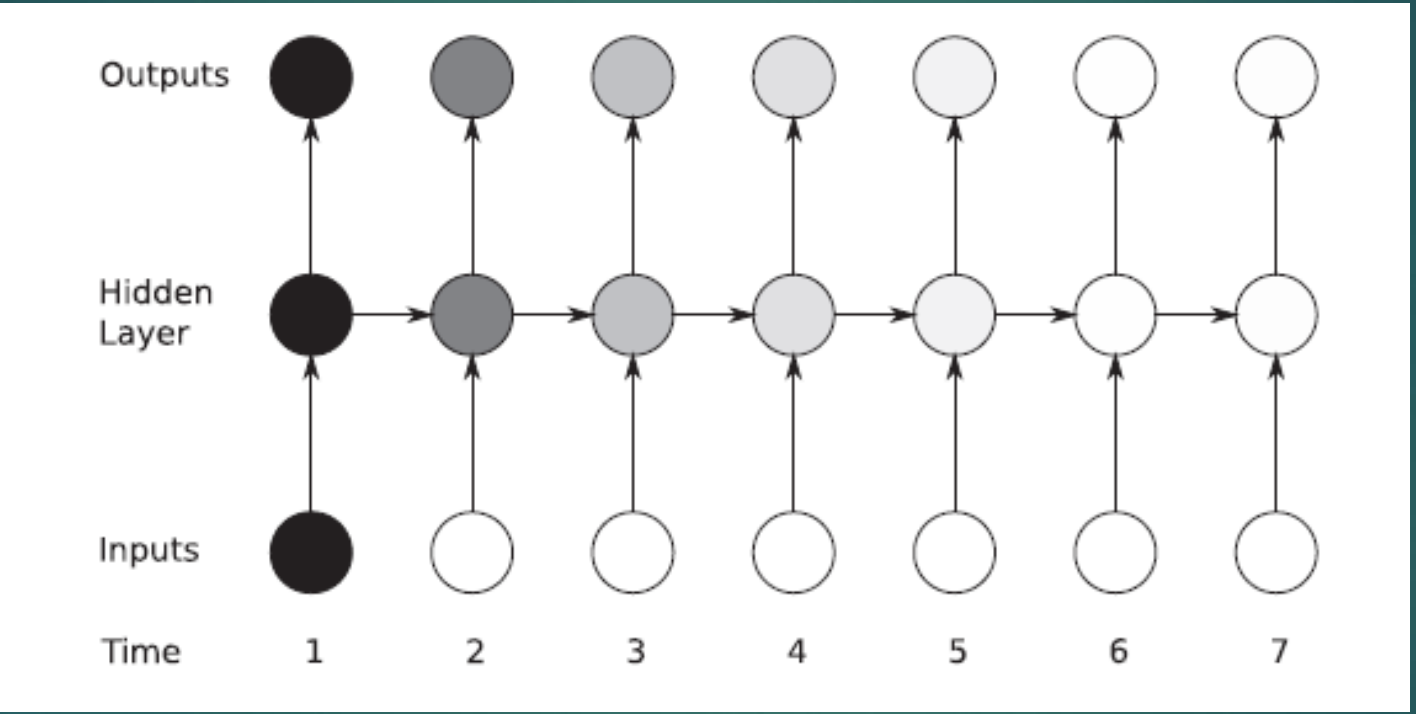
Problem with RNN

- Since the network is unfolded for many no'of time steps, when using BPTT, the gradients either explodes or vanishes. This problem is known as "**Vanishing and exploding gradients problem**"
- Because of this training difficulty, RNNs are not suited for learning long-range temporal dependencies.
- $h_t = W_{IH}\ x_t + W_{HH}\ h_{t-1}$
- $h_t = W_{IH}\ x_t + W_{HH}\ (W_{IH}\ x_{t-1} + W_{HH}\ h_{t-2})$
- $h_t = W_{IH}\ x_t + W_{HH}\ (W_{IH}\ x_{t-1} + W_{HH}(W_{IH}\ x_{t-2} + W_{HH}\ h_{t-3}))$
- $\frac{\partial h_t}{\partial h_k} = \prod_{t>i>k}\ (W_{HH}) = \ (W_{HH})^{t-k}$
- Just like for a real number 'a' ; $a^k \rightarrow \infty$ if a > 1 and $a^k \rightarrow 0$ for a < 1 as k $\rightarrow \infty$

- The term $(W_{HH})^{t-k}$ → 0 as (t-k)→ large, if $\rho < 1$ where ρ is the spectral radius of the matrix W_{HH}
- Similarly $(W_{HH})^{t-k}$ → ∞ as (t-k)→ large, if $\rho > 1$
- $\rho(A) = \max_{\lambda}\{|\lambda_1|, |\lambda_2| \ldots |\lambda_n|\}$ where A is an n x n matrix, λ s are the eigen values
- Gradient of weight Θ , w.r.to loss function L is given by,
- $\frac{\partial L}{\partial \Theta} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial \Theta}$
- $\frac{\partial L_t}{\partial \Theta} = \sum_{1 \le k \le t} \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \Theta}$
- The only term through which error flows backward in time is $\frac{\partial h_t}{\partial h_k}$ and
- $\frac{\partial h_t}{\partial h_k} = (W_{HH})^{t-k}$

- So, the gradients either vanishes to **0** or explodes to ∞ based on the spectral radius ρ
- Even though all weights are initialized between -1 and 1, it doesn't guarantee that the gradients will not explode
- For e.g. Consider, A = $\begin{bmatrix} 2/3 & 2/3 \\ 2/3 & 2/3 \end{bmatrix}$; $A^2 = \begin{bmatrix} 8/9 & 8/9 \\ 8/9 & 8/9 \end{bmatrix}$
- $A^3 = \begin{bmatrix} 128/81 & 128/81 \\ 128/81 & 128/81 \end{bmatrix}$ i.e $A^k \rightarrow \infty$ as k $\rightarrow \infty$ because its eigen values are 4/3 > 1 with multiplicity 2
- Since $\frac{\partial h_t}{\partial h_k} = (W_{HH})^{t-k}$; the problem occurs only for long-term components. i.e. terms for which (t-k) is large.

Sensitivity of inputs over time (RNN)



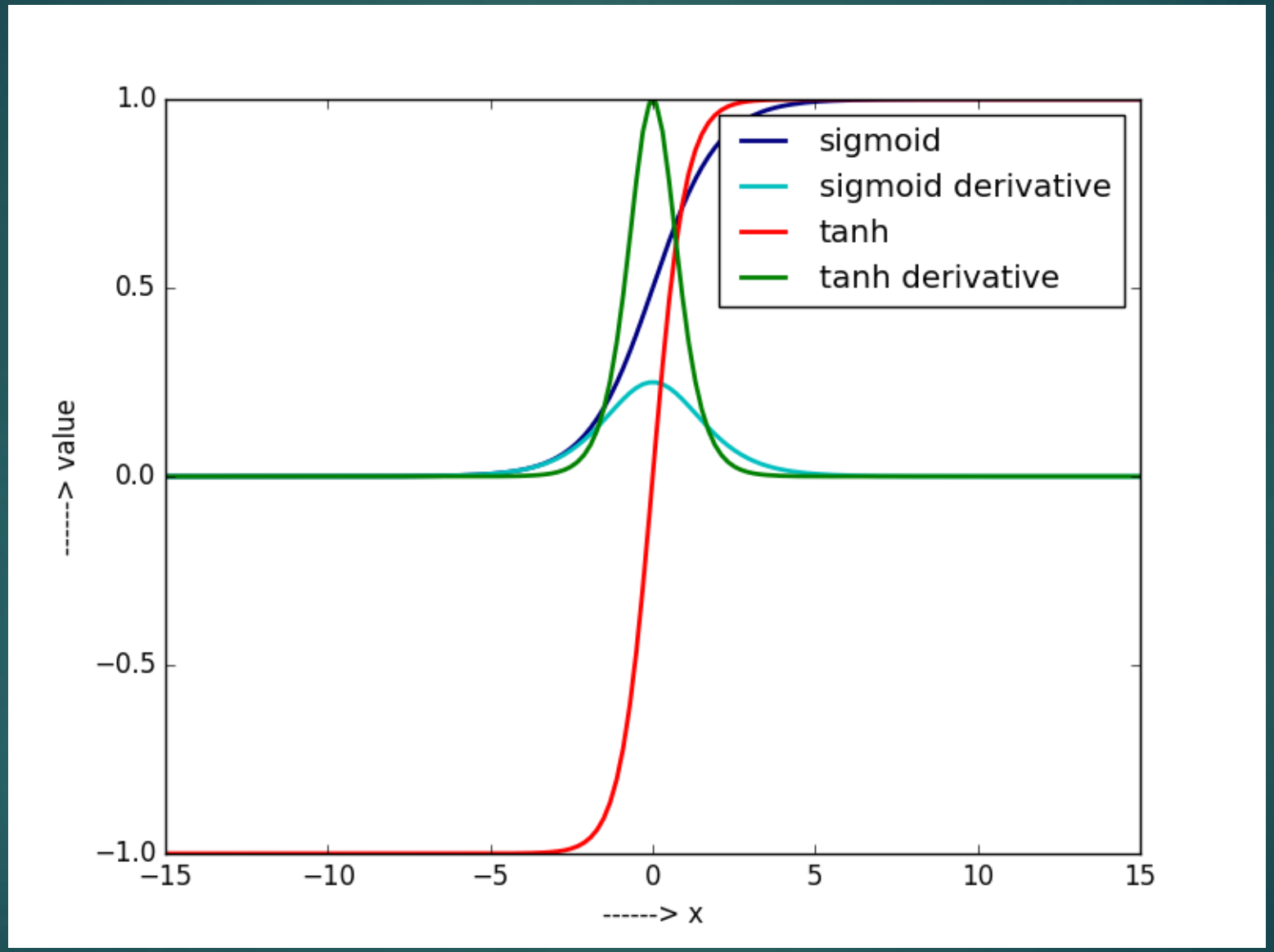
Change the activation function

- Consider a simple NN, where Error ‘E’ depends on weights $\{w_{ij}\}$ only through y_j

$$y_j = f\left(\sum_i w_{ij} x_i\right)$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial y_j}\,\frac{\partial y_j}{\partial w_{ij}} \\ &= \frac{\partial E}{\partial y_j}\, f'\left(\sum_i w_{ij} x_i\right)\, x_i\end{aligned}$$

- If we use sigmoid as an activation function, the max value of f’ is 0.25 and it’s almost zero for very small and very large inputs.


sigmoid
sigmoid derivative
tanh
tanh derivative
1.0
0.5
0.0
−0.5
−1.0
−15
−10
−5
0
5
10
15
------> value
------> x

- Hence, when the output of the cell is near to zero or one, gradients don’t propagate well back. This is called the saturation problem.
- Because of this, it is much more difficult to train the below layers when compared to the outer ones.(gradients keep on decrease in size)
- We can use other activations like tanh, which is better than sigmoid
- One solution is to use ReLU activation function, which has the constant derivative of 1 for positive inputs. Hence the gradients won’t diminish over the layers
- ReLU activation function $$f(x) = \max(0, x)$$

Scale the gradients

- A simple way to manage exploding gradients problem is to rescale the gradients if the norm exceeds a particular value.

Pseudocode:

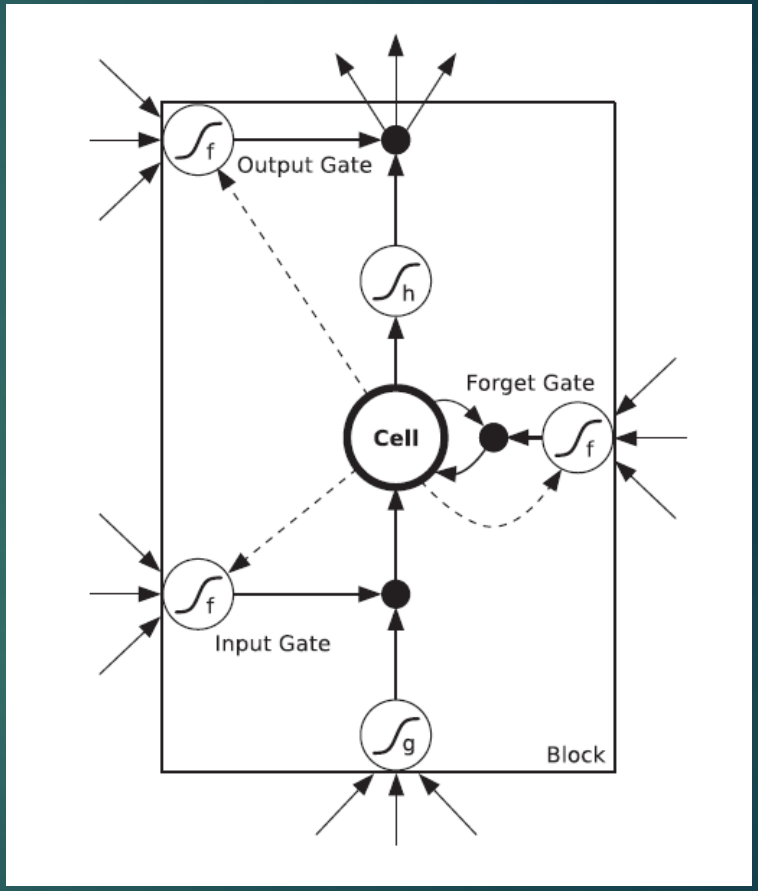
1. Find the gradient $\tilde{w} = \frac{\partial L}{\partial w}$
2. if $\|\tilde{w}\| > threshold$, then
   $\tilde{w} = \frac{threshold}{\|\tilde{w}\|}\tilde{w}$
   endif

LSTM

- LSTM(Long Short Term Memory)s motivated by the problem of capturing long-term dependencies in RNN
- LSTMs are a version of RNN, with specific architecture proved to be effective in learning long-range time dependencies.
- The LSTM architecture consists of a set of recurrently connected subnets, known as memory blocks.
- Each block contains one or more self-connected memory cells and three multiplicative units—the input, output and forget gates
- continuous analogues of write, read and reset

- The multiplicative gates allow LSTM memory cells to store and access information over long periods of time, thereby mitigating the vanishing gradient problem.
- For example, as long as the input gate remains closed the activation of the cell will not be overwritten by the new inputs arriving in the network, and can therefore be made available to the net much later in the sequence, by opening the output gate.
- Total no’of parameters

  I*H*4 + H*H*4 + H*K + H*3



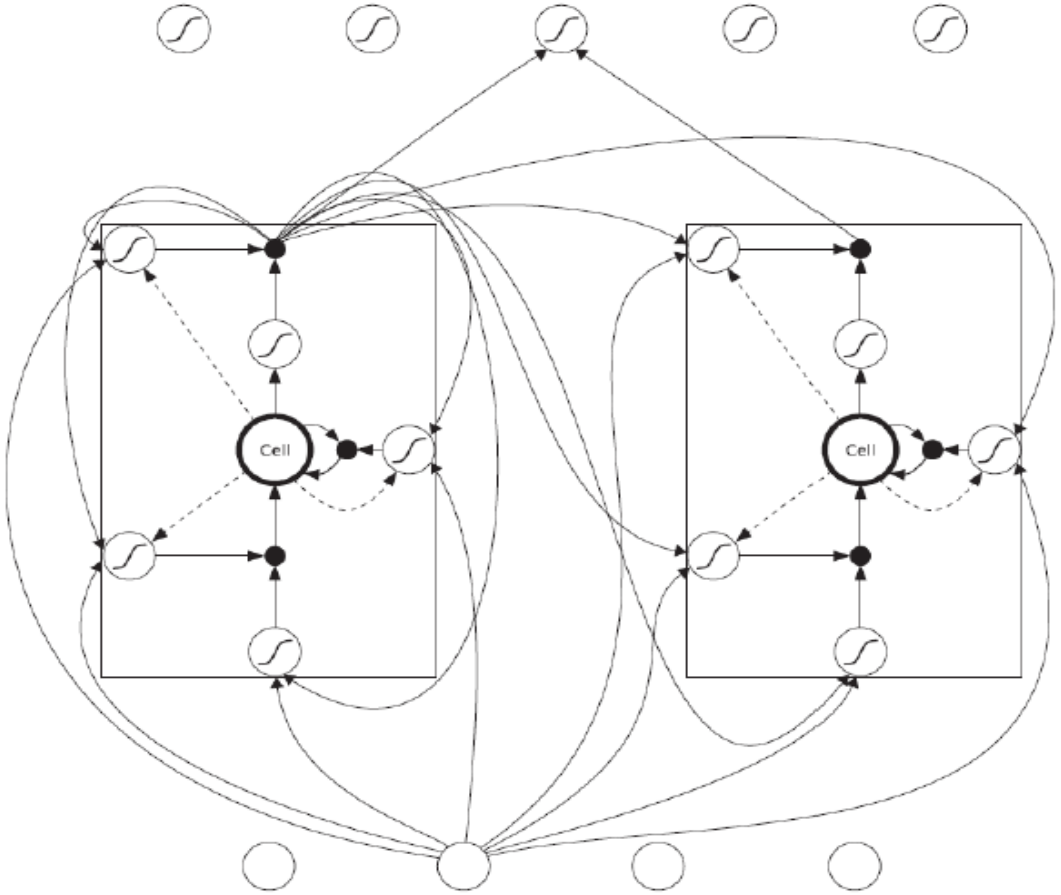


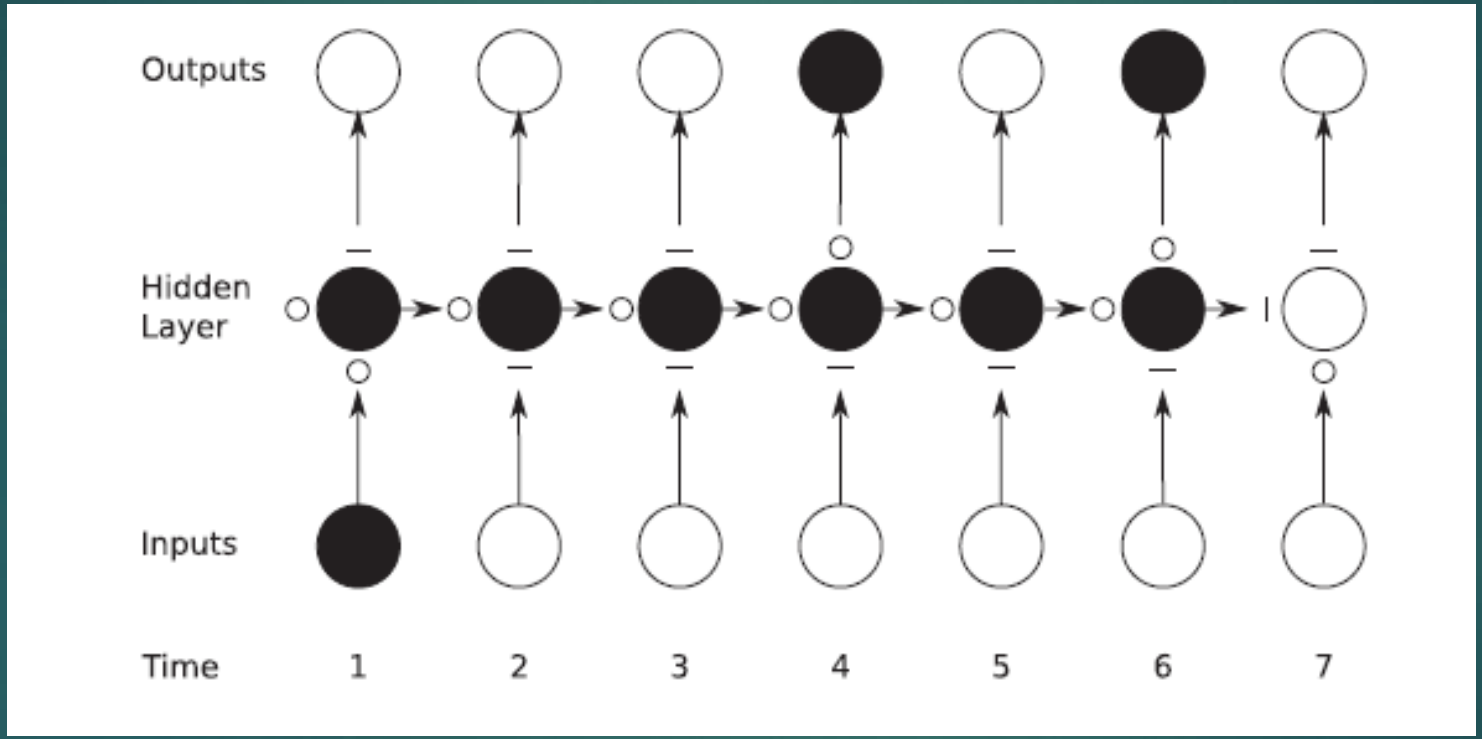
Figure 3: **An LSTM network** The network consists of four input units, a hidden layer of two single-cell LSTM memory blocks and five output units. Not all connections are shown.

LSTM Equations

$\tilde{z}_t = W_{zx} x_t + W_{zm} m_{t-1} + b_z$	net block input
$z_t = g(\tilde{z}_t)$	block input
$\tilde{i}_t = W_{ix} x_t + W_{im} m_{t-1} + W_{ic} c_{t-1} + b_i$	net input to input gate
$i_t = \sigma(\tilde{i}_t)$	output of input gate
$\tilde{f}_t = W_{fx} x_t + W_{fm} m_{t-1} + W_{fc} c_{t-1} + b_f$	net input to forget gate
$f_t = \sigma(\tilde{f}_t)$	output of forget gate
$c_t = f_t \odot c_{t-1} + i_t \odot z_t$	new cell state
$\tilde{o}_t = o_t = W_{ox} x_t + W_{om} m_{t-1} + W_{oc} c_{t-1} + b_o$	net input to output gate
$o_t = \sigma(\tilde{o}_t)$	output of output gate
$m_t = o_t \odot h(c_t)$	block output

- The only term through which error flows backward in time is $\frac{\partial c_t}{\partial c_k}$, $k < t$ and
- $\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial (f_t \odot c_{t-1})}{\partial c_{t-1}} = diag(f_t) = diag(\sigma(\tilde{f}_t))$
- $\frac{\partial c_t}{\partial c_k} = \prod_{t \geq i > k} diag(\sigma(\tilde{f}_i))$
- While this term can approach zero, there is no specific factor in it (i.e. some weight) which will drive the derivative to zero for very large time lags, i.e. for t >> k
- This part is also safe from exploding, since each of the factors is it is bounded by 1 due to the activation function sigmoid.

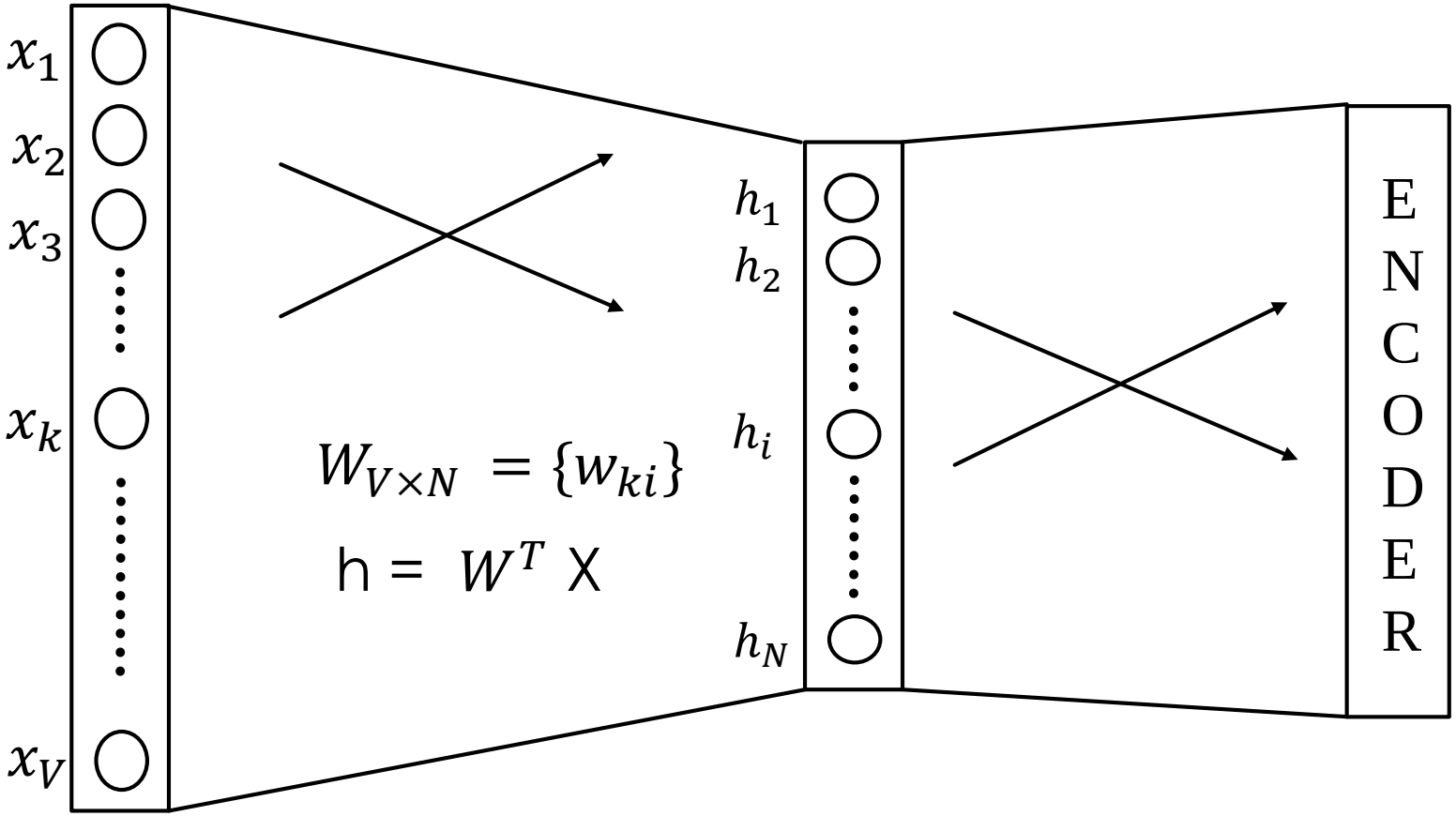
Sensitivity with time(LSTM)



Word embedding

- Since we can't directly input the words into the networks, we have to represent them in vector format
- First form a vocabulary of words in the training set
- Let V is the size of the vocabulary
- Represent each word as using 1-of-V encoding
- Let N be the size of embedding. Then project the V dimensional vector onto N dimensions using V x N weight matrix
- So, the representation of the i^{th} word is given by the i^{th} row of the matrix

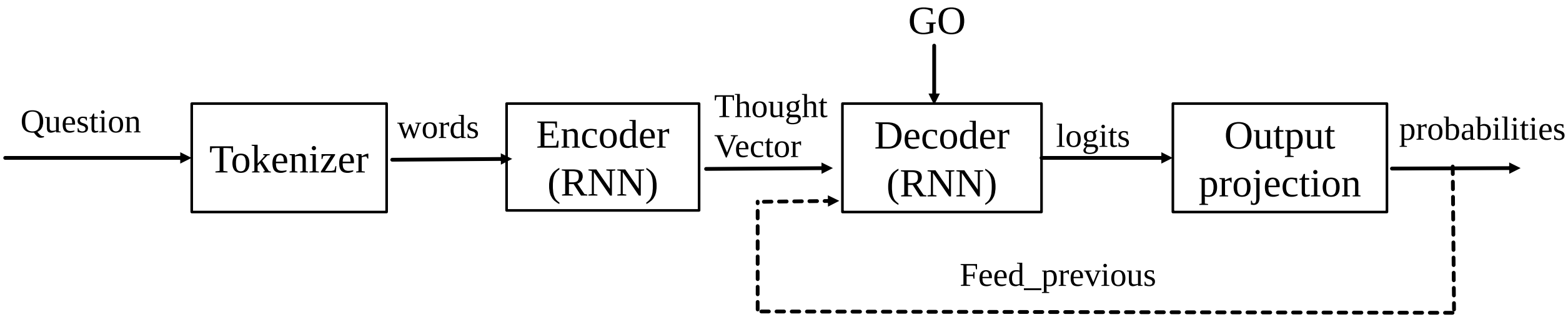
$$h = W^T X$$


x_1
x_2
x_3
x_k
x_V
$W_{V\times N} = \{w_{ki}\}$
h = W^T X
h_1
h_2
h_i
h_N
ENCODER

Architecture

- A simple strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN
- Input $(x_1, x_2, x_3 \ldots\ldots x_T)$, output $(y_1, y_2, y_3 \ldots\ldots y_{T'})$
- $P(y_1, y_2, y_3 \ldots\ldots y_{T'} / x_1, x_2, x_3 \ldots\ldots x_T) = P(y_1, y_2, y_3 \ldots\ldots y_{T'} / \vartheta)$

  $= \prod_{t=1}^{T'} P(y_t / \vartheta, y_1, y_2 \ldots\ldots y_{t-1})$
- Encoder $(x_1, x_2, x_3 \ldots\ldots x_T)$ -------- ϑ
- Decoder -------- predicts $P(y_t / \vartheta, y_1, y_2 \ldots\ldots y_{t-1})$



While training, Feed_previous = False
While testing, Feed_previous = True

Basic Model

- Encoder is modelled using an RNN/LSTM, such that $(x_1, x_2 \ldots . x_T)$ --------- > ϑ

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

$$\vartheta = q(h_1, h_2 \ldots h_T)$$

- Here ϑ = q(h_1, $h_2 \ldots \ldots h_T$) = h_T
- With decoder RNN/LSTM, this conditional probability distribution can be modeled as

$$p(y_t/\vartheta, y_1, y_2 \ldots y_{t-1}) = g(y_{t-1}, s_t, \vartheta)$$

- where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t , s_t is the hidden state of decoder at time 't', given by

$$s_t = \begin{cases} f(\vartheta) & \text{if } t = 1 \\ f(s_{t-1}, y_{t-1}) & \text{otherwise} \end{cases}$$

Model with attention

- Includes the attention mechanism.
- Allows the decoder to peek at input at every time step while decoding.
- Here the conditional probability is modeled as

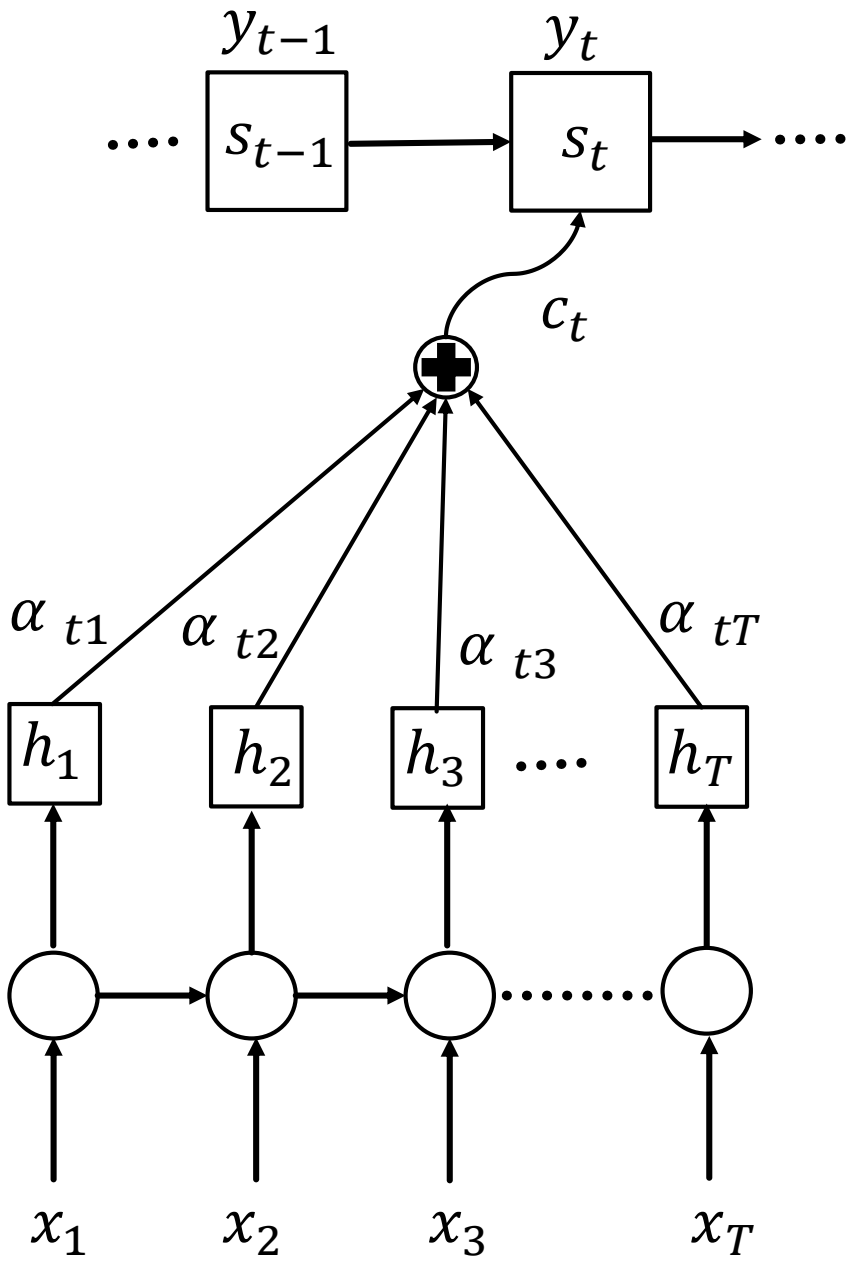
$$p(y_t/y_1, y_2 \ldots y_{t-1}, x) = g(y_{t-1}, s_t, c_t)$$

- where s_t is the hidden state of the decoder at time t, given by

$$s_t = f(y_{t-1}, s_{t-1}, c_t)$$

- where c_t is the context vector for symbol y_t. Unlike the basic model, here the probability is conditioned on different context vector for each symbol.
- Context vector depends on the sequence of encoder hidden states ($h_1, h_2 \ldots\ldots h_T$)

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


y_{t-1}
y_t
s_{t-1}
s_t
c_t
α_{t1}
α_{t2}
α_{t3}
α_{tT}
h_1
h_2
h_3
h_T
x_1
x_2
x_3
x_T

- weights α_{tj} is given by

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

- where

$$e_{tj} = a(s_{t-1}, h_j)$$

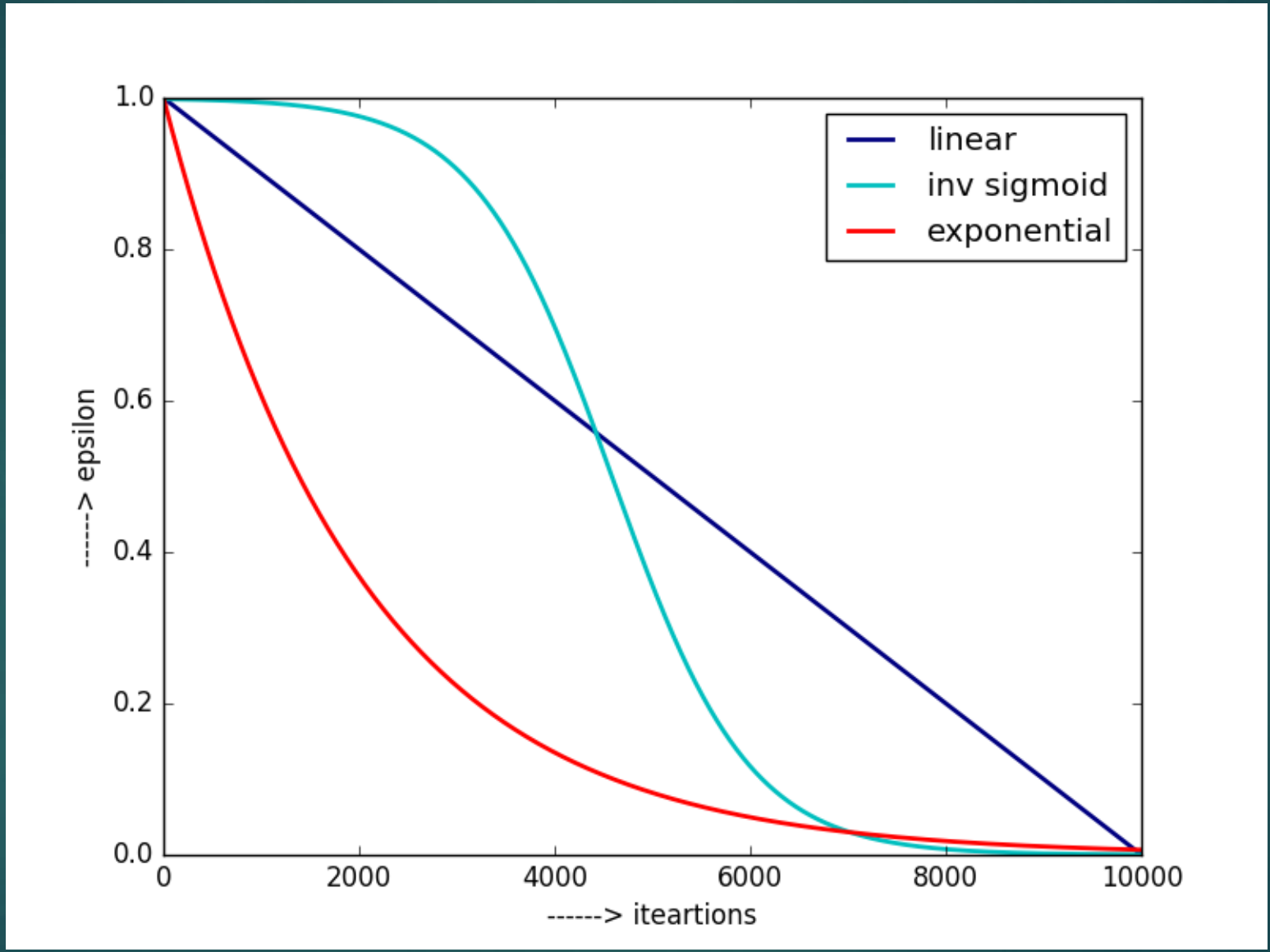
- e_{tj} is a score of how well the input at position 'j' and output at position 't' match.
- Alignment model *a* can be modeled as a feed forward neural network.

$$e_{tj} = v^T \tanh(W_1 h_j + W_2 s_{t-1})$$

- The vector *v*, matrices *W*1, *W*2 are the learn-able parameters of the model.

Scheduled sampling

- While training, we provide the current token to the decoder to predict the next token
- While inference, current token is considered to be the one generated by the model itself.
- So, if the model makes a mistake early in the sequence, there is a chance that the error is propagated to the next time steps and it gets accumulated.
- This happens mainly because of the discrepancy between training and testing methods.
- To make the model robust to its errors, proposed a strategy which is scheduled sampling.


linear
inv sigmoid
exponential
1.0
0.8
0.6
0.4
0.2
0.0
-----> epsilon
0
2000
4000
6000
8000
10000
------> iteartions

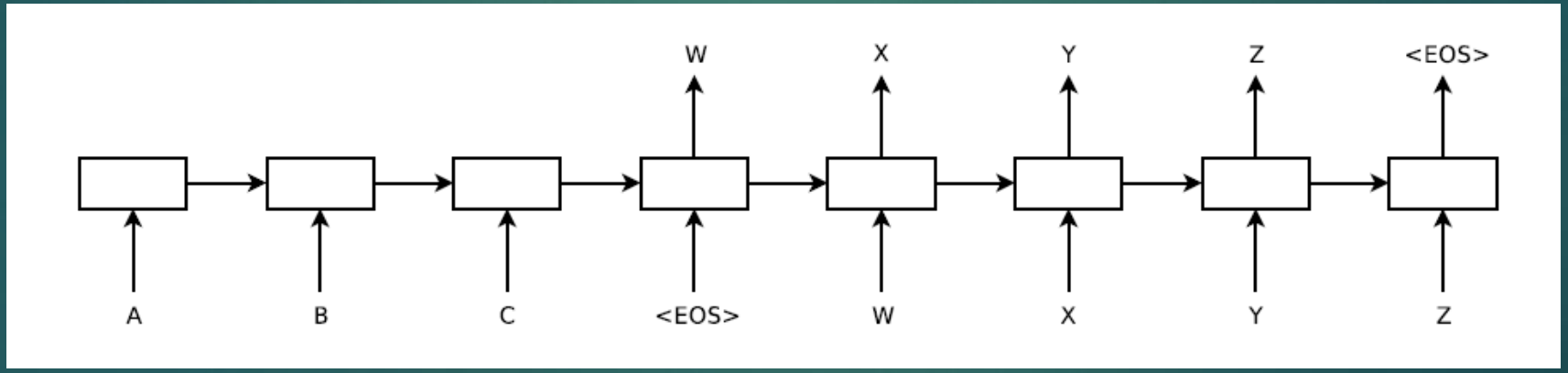
- According to this strategy, choose the true token at the i^{th} iteration with probability ϵ_i, and the estimate coming out of model with probability (1-ϵ_i)
- Where estimate at time t-1 is

$$\arg\max_{s} P(y_{t-1} = s\,;\, s_{t-1})$$

- And the strategy is to reduce it as ‘i’ increases. In this way, we can sample more from the estimates as the training progresses, which makes the model ready for the inference.
- Some of the strategies are
  - Linear ϵ_i = max (ϵ, k- c.i) ; $0 \leq \epsilon < 1$; k, c decides the rate of convergence
  - Exponential $\epsilon_i = k^i$; k < 1

Dataset

- The training data consists of source sentence and the corresponding response.
- answer ------- ‘GO’ + answer + ‘EOS’
- Each question and answer lengths are limited to certain length. for e.g. (20,20)
- Since, it allows us for batch processing and we can use matrix operations.



Training

- Consider the translation *ABC* → *WXYZ*.
- The encoder inputs would be A, B, C in sequence.
- Decoder inputs would be GO, W, X, Y, Z.
- Outputs of the decoder expected at each time step would be W, X, Y, Z, EOS.
- The output of the decoder at time ’t’ is fed back to the decoder, to predict the word at time ’t+1’.
- provide the correct input to the decoder at every time-step, even if the decoder made a mistake before.
- Prepare the vocabulary of most frequent words from both the source and target files.
- Embed the words before feeding into encoder.

- Generate the ‘thought vector’ (final encoder state).
- Thought vector is considered as the initial hidden state of the decoder.
- Now, feed the decoder inputs one after the other.
- Decoder generates an array of V’ values, each corresponds to the logits of the vocabulary on the target side.
- Output of the decoder gives logits (range $(-\infty, \infty)$)
- After output projection

  probabilities = softmax(logits*W + b)
- Cross entropy error (Perplexity) = $-\sum_{k \in V}(z_k \log y_k)$
- Back propagate the error to jointly make changes in the encoder and decoder parameters

- During testing, start with ‘GO’ until we generate ‘EOS’
- Encoder and decoder jointly work together to maximize the probability of output sequence given the input sequence
- Training objective is to maximize

$$\frac{1}{|S|}\sum_{(t,s)\,\in\,S}\log P(t/s)$$

- $\mathrm{P}(y_1, y_2, y_3 \ldots. y_{T'} / x_1, x_2, x_3 \ldots. x_T) = \mathrm{P}(y_1, y_2, y_3 \ldots. y_{T'} / \vartheta)$

$$= \prod_{t=1}^{T'} \mathrm{P}(y_t / \vartheta, y_1, y_2 \ldots. y_{t-1})$$

- $\log \mathrm{P}(y_1, y_2, y_3 \ldots. y_{T'} / x_1, x_2, x_3 \ldots. x_T) = \sum_{t=1}^{T'} \log \mathrm{P}(y_t / \vartheta, y_1 \ldots. y_{t-1})$
- Most likely o/p sequence = $\arg\max_t \log P(t/s)$

Evaluation measures

- **Perplexity**
- Let N be the length of sequence, then

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \log p_{target_i}$$

- Where, $target_i$ is the target word at the i^{th} position.
- It is nothing but the average of cross entropy loss over the sequence
- Perplexity = exp(loss)

Experiments with cornell movie dataset

- Has conversations from 617 movies.
- Conversational exchanges between 10,292 movie characters.
- After pre-processing, able to extract 1,30,000 one-to-one (Q,A) pairs.
- For simplicity, (Q,A) pair length is limited to (10, 15)

- Here are the model parameters used
  - vocabulary side ----------> 25,000 (both source and target sides)
  - word embedding size ----------> 32
  - num of layers ----------> 2
  - num of cells in each layer -----> 512
  - batch size ----------> 10
  - num of iterations ----------> 47,500
  - learning rate (initial) ----------> 0.5

Sample conversation

- *human*: hello
- *machine*: hello
- *human*: how are you?
- *machine*: i 'm fine
- *human*: who made you?
- *machine*: you
- *human*: are you human?
- *machine*: no
- *human*: are you a robot?
- *machine*: yes
- *human*: are you male?
- *machine*: no
- *human*: are you female?
- *machine*: yes
- *human*: will you come to movie with me?
- *machine*: i will
- *human*: what time is okay with you?
- *machine*: eight
- *human*: where do you live?
- *machine*: i do n't know
- *human*: what's your age?
- *machine*: you know
- *human*: how old are you?
- *machine*: i 'm fine
- *human*: are you woman?
- *machine*: no
- *human*: are you man?
- *machine*: yes

- Not much change in perplexity is observed using the attention mechanism.
- The conversation above suggests that the model can converse with a human fluently.
- Since the model is trained for one-to-one mapping, it is unable to take into account the previous sentences while conversing. Hence, the model is not good at maintaining the context.
- It can also be seen from the conversation that the model lacks the personality and its responses are inconsistent.
- This is because of using the noisy, open-domain dataset that is used to train the model.
- This lack of consistent personality makes it difficult to pass the Turing test.

SNE

- Stochastic Neighbor Embedding
- Dimensionality Reduction Technique
- Objective is to model similar objects by nearby points and dissimilar objects by distant points.
- X = $\{x_1, x_2 \ldots x_n\}$ are high dimensional data points and Y = $\{y_1, y_2 \ldots y_n\}$ are low-dimensional data points
- It first coverts the distances between data points in high dimensional space to joint probabilities

$$p_{j/i} = \frac{exp(-\|x_i - x_j\|^2 / 2\sigma_i^2}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

- Where σ_i is the variance of the Gaussian centered at x_i, $p_{j/i}$ is the probability of x_i picking x_j as neighbor.

- σ_i will vary according to the density of data points around x_i (if the region around x_i is dense σ_i will be low and vice versa).

$$q_{j/i} = \frac{exp(-\|y_i - y_j\|^2}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)}$$

- We are only interested in modelling pairwise similarity, so set $p_{i/i} = q_{i/i} = 0$
- Our aim is to make the conditional probabilities $p_{j/i}$, $q_{j/i}$ similar. i.e. we have to reduce the difference between them, using KL divergence.

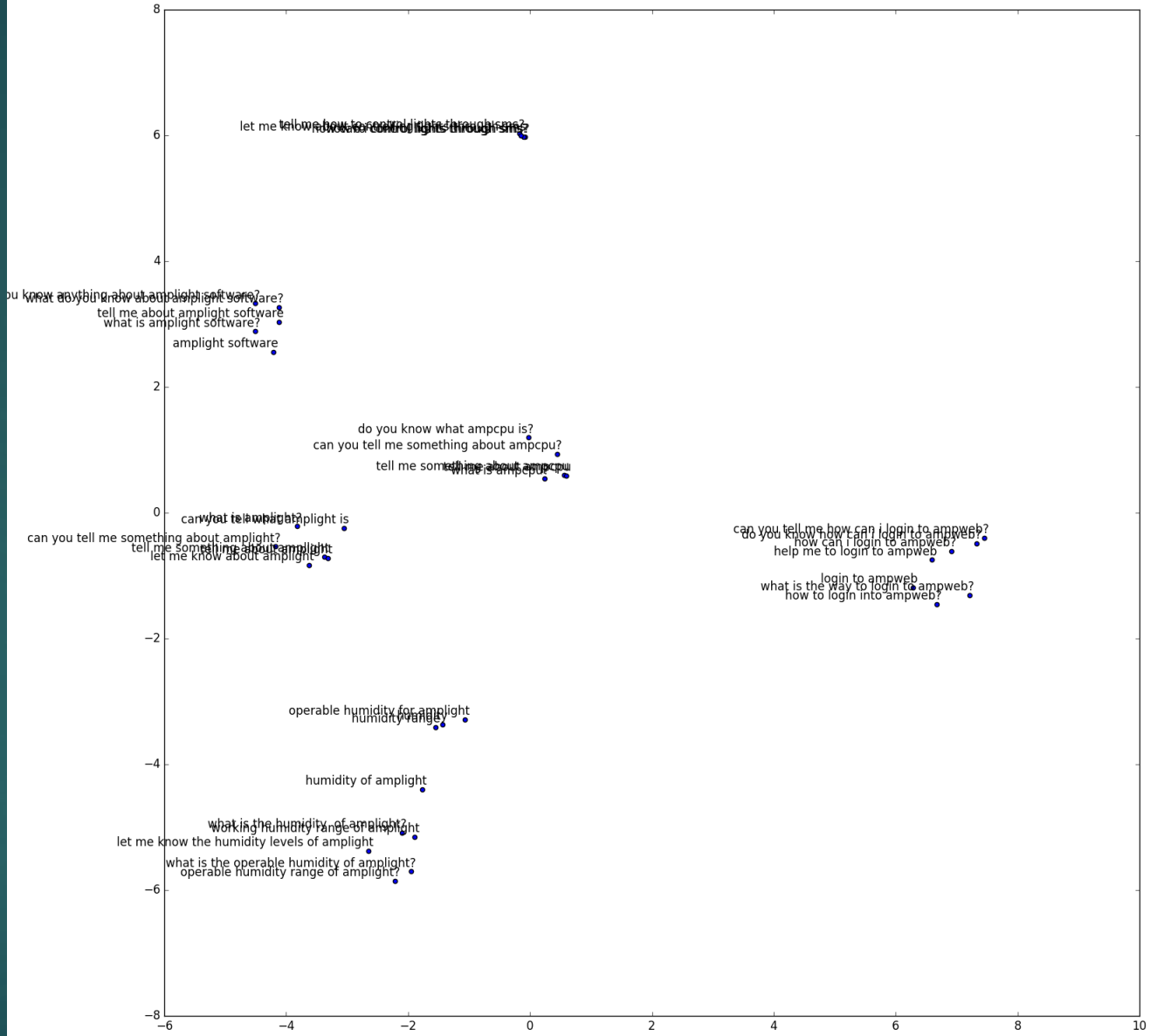
$$C = \sum_i \mathrm{KL}(P_i||Q_i) = \sum_i \sum_j p_{j/i} log \frac{p_{j/i}}{q_{j/i}}$$

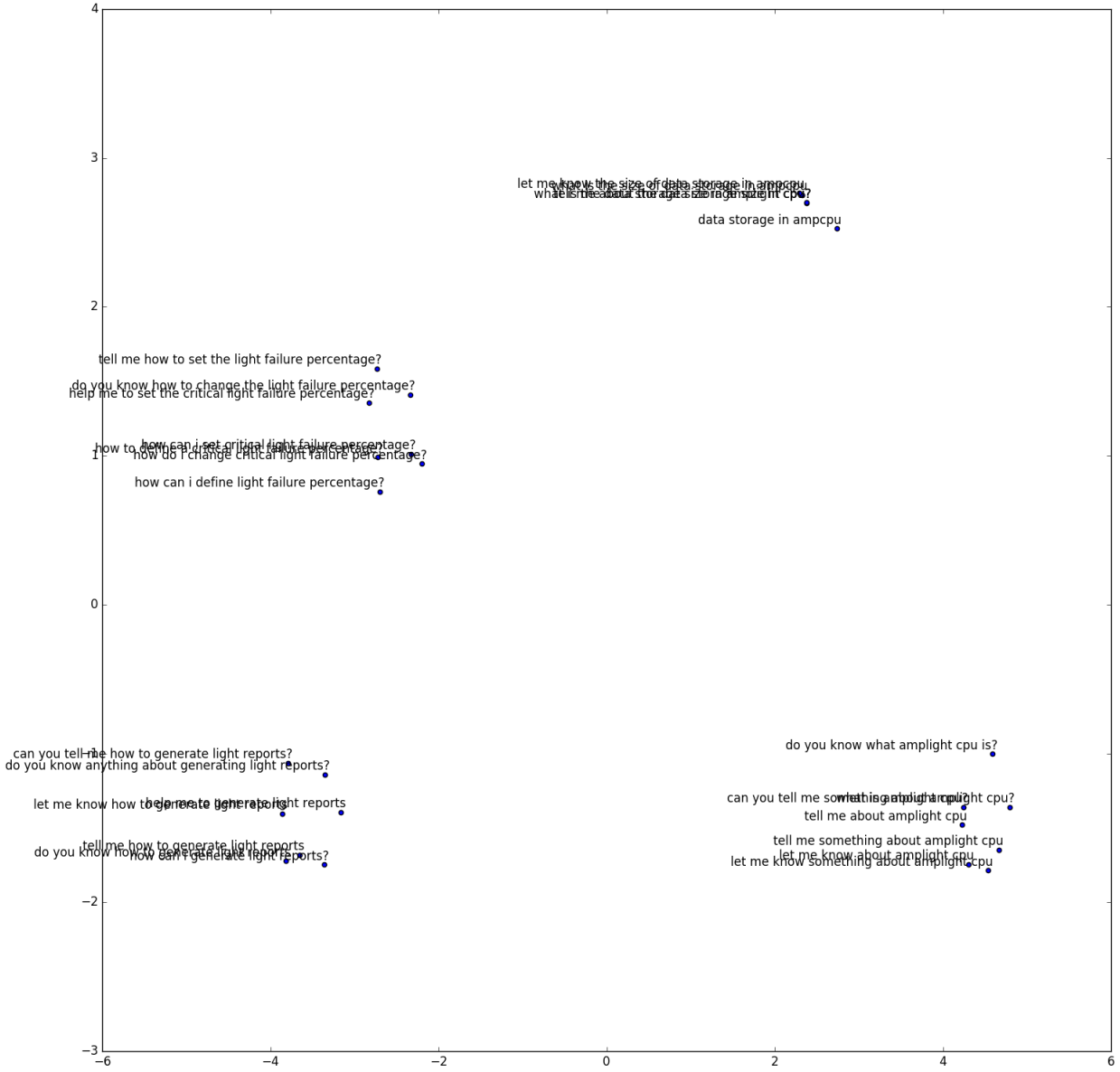
- Where P_i is the conditional probability distribution over all data points given x_i

- The gradient is computed to be

$$\frac{\partial C}{\partial y_i} = 2\sum_j (p_{j/i} - q_{j/i} + p_{i/j} - q_{i/j})(y_i - y_j)$$

- Gradient is proportional to the mismatch of the similarities between data points and map points.
- Using Gradient descent, we can compute y_i over iterations.
- This technique was used to project 128 dimensional vectors that are obtained out of encoder RNN, onto a 2 dimensional space.
- Similar sentences are supposed to be clustered on the 2d space because their vector similarity is more


8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8
10
let me know how to control lights through sms?
tell me how to control lights through sms?
how to control lights through sms?
what do you know about amplight software?
tell me about amplight software
what is amplight software?
amplight software
do you know what ampcpu is?
can you tell me something about ampcpu?
tell me something about ampcpu
what is ampcpu?
can you tell what amplight is
can you tell me something about amplight?
tell me something about amplight
let me know about amplight
can you tell me how can i login to ampweb?
do you know how can i login to ampweb?
how can i login to ampweb?
help me to login to ampweb
login to ampweb
what is the way to login to ampweb?
how to login into ampweb?
operable humidity for amplight
humidity range?
humidity of amplight
what is the humidity of amplight?
working humidity range of amplight
let me know the humidity levels of amplight
what is the operable humidity of amplight?
operable humidity range of amplight?


4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
6
data storage in ampcpu
tell me how to set the light failure percentage?
do you know how to change the light failure percentage?
help me to set the critical light failure percentage?
how can i define light failure percentage?
do you know what amplight cpu is?
tell me about amplight cpu
tell me something about amplight cpu
let me know about amplight cpu
can you tell me how to generate light reports?
do you know anything about generating light reports?
tell me how to generate light reports
how can i generate light reports?

BLEU Score

- Bi Lingual Evaluation Understudy
- Evaluation metric for Machine translation.
- Based on modified n-gram precision.
- Language independent,
- Operate on local level , doesn’t take grammatical correctness into account

$$BLEU = B.\ \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right)$$

- Where

$$B = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } c < r \end{cases}$$

- p_n is the modified n-gram precision, which is the number of n-grams in the candidate translation that occur in any of the reference translations, out of total number of n-grams in the candidate translation.
- For N=4, the score is found to be in much correlation with the human judgement.

SYSTEM A: Israeli officials (2-GRAM MATCH) responsibility of airport (1-GRAM MATCH) safety

REFERENCE: Israeli officials are responsible for airport security

SYSTEM B: airport security (2-GRAM MATCH) Israeli officials are responsible (4-GRAM MATCH)

Metric	System A	System B
precision (1gram)	3/6	6/6
precision (2gram)	1/5	4/5
precision (3gram)	0/4	2/4
precision (4gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

TIDES-IIIT Eng-Hin parallel corpus

- DARPA-TIDES released in 2002 for SMT
- Later refined by (Venkatapathy, 2008) for NLP tools Contest.
- “ the translations are not faithful. They are paraphrased in such a way that they convey the meaning in the best possible way. i.e they are not the exact translations ”
- 50k training, 1k validation
- Limited the length to 30
- 33k used for training, 716 for validation

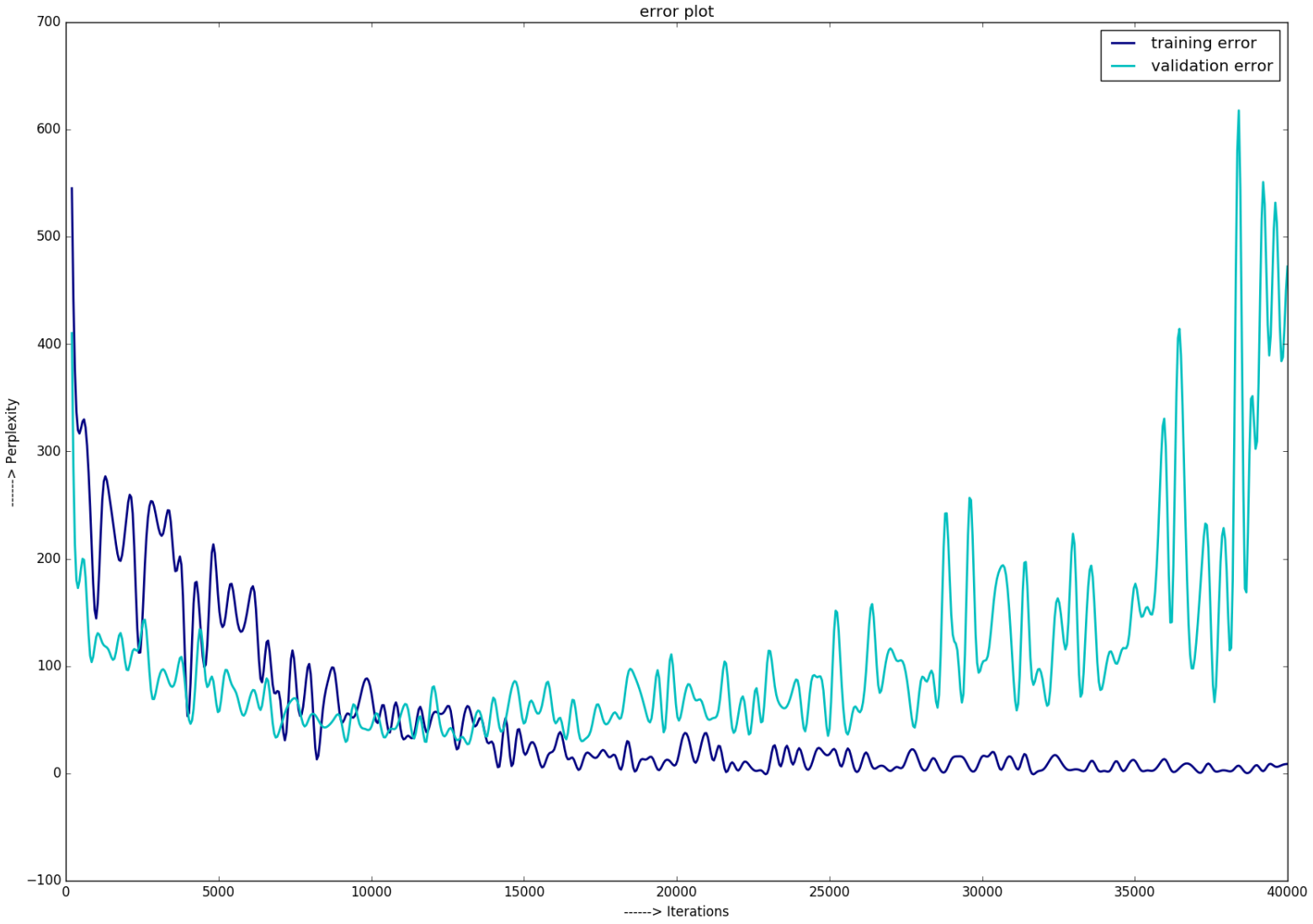
- Number of layers 2
- Number of cells(LSTM) 512
- Batch size 32

Model used	BLEU score	
	Vocabulary 8k	Vocabulary 12k
Basic Seq2Seq	3.48	3.66
Seq2Seq with attention	8.05	21.47

TABLE 5.1: Experimental Results on TIDES-IIIT parallel corpus.

- Scheduled sampling with exponential decay after 50k iterations resulted a BLEU score of 23.13 on training set

- Seqseq model with attention mechanism has boosted the performance of the translation model.
- Even though scheduled sampling achieved an improvement in the model performance, the choice of decay schedule has proved to be crucial.
- Using the scheduled sampling from the beginning of the training has deteriorated the performance of the model.
- Hence it is suggested to use a decay schedule which decreases slowly and is better to use it after certain number of iterations.



Overfitting of the model to the training data

Conclusions

- Seq2Seq models learn the semantic and syntactic relations between the source and target sentence pairs by maximizing the likelihood of the target sentence given the source.
- Attention mechanism is useful in learning the alignment between the source and target words in MT task rather than on a mere question-answering task.
- Scheduled sampling has resulted in the performance improvement of the model, but one should choose a proper decay schedule, otherwise it could lead to a bad model.
- The projections of the thought vectors from an encoder network onto 2D space indicates that the model generates similar thought vectors for similar questions

Future works

- Instead of modelling only uni-directional source-to-target dependency P(Y/X), we can model source-to-target dependency P(X/Y).
- We can use Bi-directional RNN or Bi-directional LSTMs.
- Evaluation metrics doesn’t take grammatical correctness into account. Doesn’t correlate much with human judgement. New evaluation metrics can be explored.
- Instead of using the random word embeddings, we can use the word vectors obtained from a language model so that it can speed-up the training process.
- Multi-tasking can be mixed with Seq2Seq models like using a shared encoder to translate into multiple languages from the source language.

THANK YOU