Week 1 : Recurrent neural Networks. (v.v.y week for me).

L-1 why sequence models. [10.20 Am 09.01.19]

Example of sequence data and application.

1. Speech recognition. (can be useful.
2. Music generation.
3. Sentiment classification. (v.v.y)
4. DNA sequence Analysis.
5. machine translation.
6. video activity recognition. [video frame থেকে বের করে নিয়ে হবে ...]
7. Name entity recognition.

L-2 Notation [10.30 Am 09.01.19]

motivating example.

Name entity recognition :- (where the peoples name in a sentence used in search engines).

→ Indexing different types of names from text (people, name, currency Name, city name, company name etc)

x: Harry potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ . . . . $x^{<t>}$ . . . . $x^{<9>}$

y: $y^{<1>}$ . . $y^{<2>}$ . . . . . . $y^{<t>}$ . . . . . $y^{<9>}$

T_x = length of input sequence.

T_y = length of output sequence.

$X^{(i)<t>}$ = t^{th} element of i^{th} training example i.

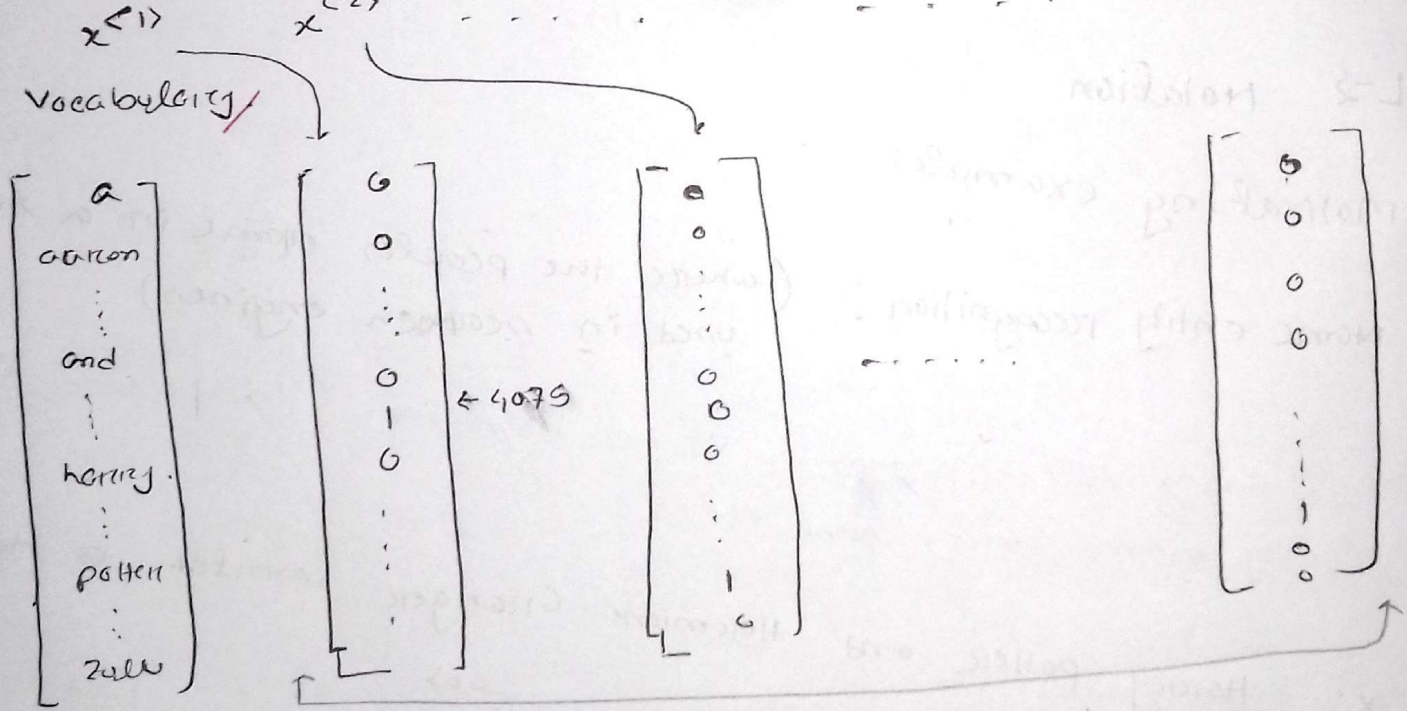
$T_x^{(i)}$ = input sequence length of i^{th} example.

$Y^{(i)<t>}$ = t^{th} element of i^{th} training example in output

$T_y^{(i)}$ = output sequence length of i^{th} example.

How to represent words of a sentence.

X: Harry Potter and Hermoine Granger invented a new spell.



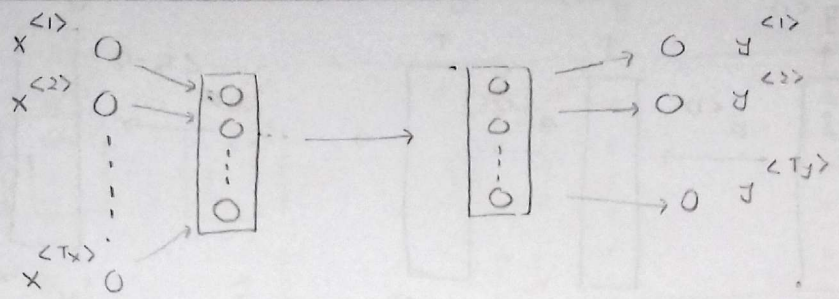
** one hot vector to represent words.

[your word is 1 vocabulary size is 10000 vector create 10000].

** we can create a vocabulary or dictionary by taking 10000/20000 common words in our training set.

L-3 Recurrent neural network model. [10.55 Am 04.02.19]

Why not a standard network?



Problems :-

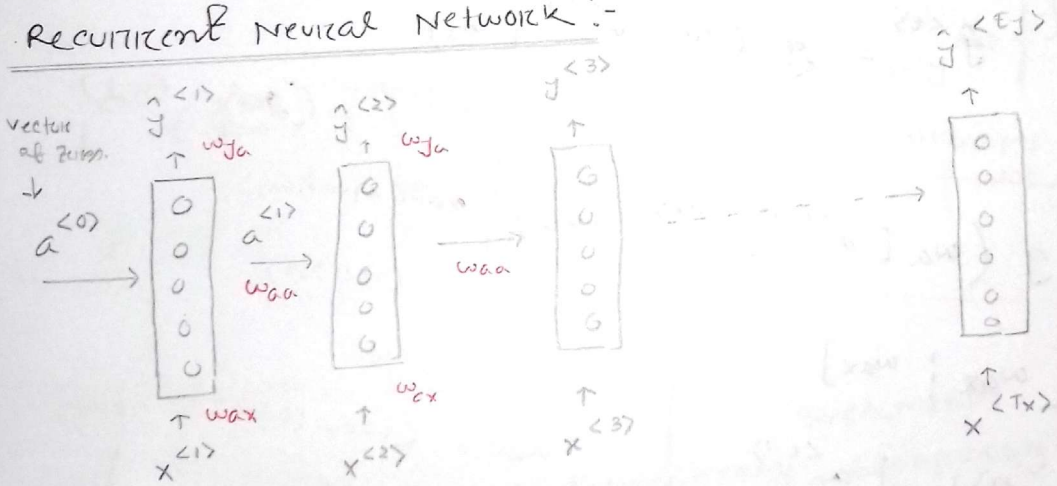
1. Inputs, outputs can be different lengths in different examples.

main disadvantage → 2. Doesn't share features learned across different positions of a text.

(If we have a vocabulary size of 20,000 and max word size = 100 then number of parameters will be very large);

→ This problem can be solved by recurrent neural network.

Recurrent Neural Network :-



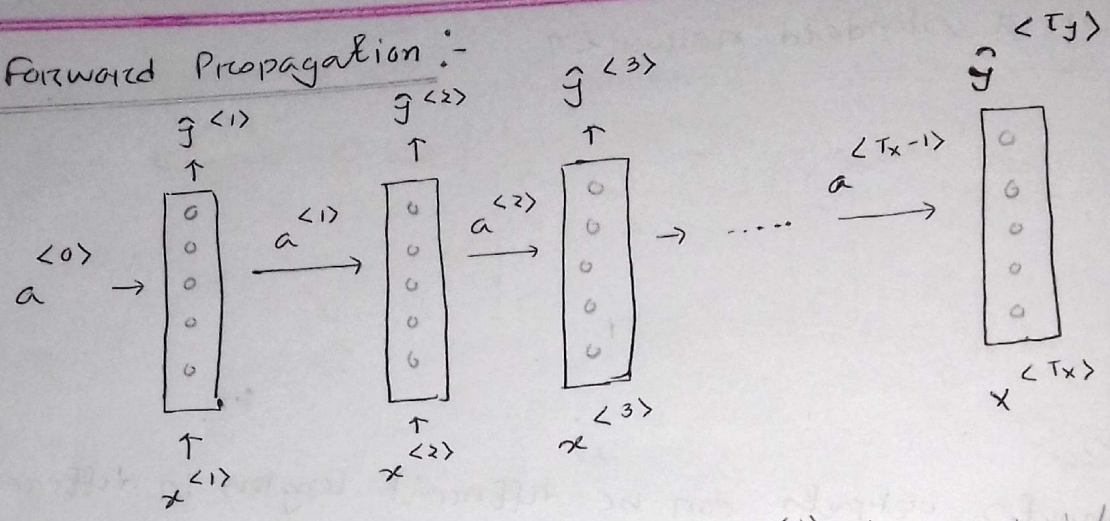
[** one weakness of RNN is that it uses only the ealier information

[sometimes we need to features from earlier as well as later words in that case performance of RNN falls.]

(slide 3 example next one)

(Cont)

Forward Propagation :-



$a^{<0>} = \vec{0}$

$$a^{<1>} = g(w_{aa} a^{<0>} + w_{ax} x^{<1>} + b_a) \leftarrow \tanh/\text{Relu}$$

$$\hat{y}^{<1>} = g(w_{ya} a^{<1>} + b_y) \leftarrow \text{sigmoid}$$

$$\vdots$$

$$a^{<t>} = g(w_{aa} a^{<t-1>} + w_{ax} x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(w_{ya} a^{<t>} + b_y).$$

→ This equation can be changed slightly : (Short form)

$$a^{<t>} = g(w_a [a^{<t-1>}, x^{<t>}] + b_a)$$ main equation.

Some

$$w_a = [w_{aa} \vdots w_{ax}].$$

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

$$\therefore [w_{aa} \vdots w_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

$$= w_{aa} a^{<t-1>} + w_{ax} x^{<t>}$$

Let $w_{aa} = (100, 100)$

$a^{<t-1>} = 100$

$w_{ax} = (100, 10000)$

$x^{<t>} = 10000$

So $= w_a = (100, 10100)$

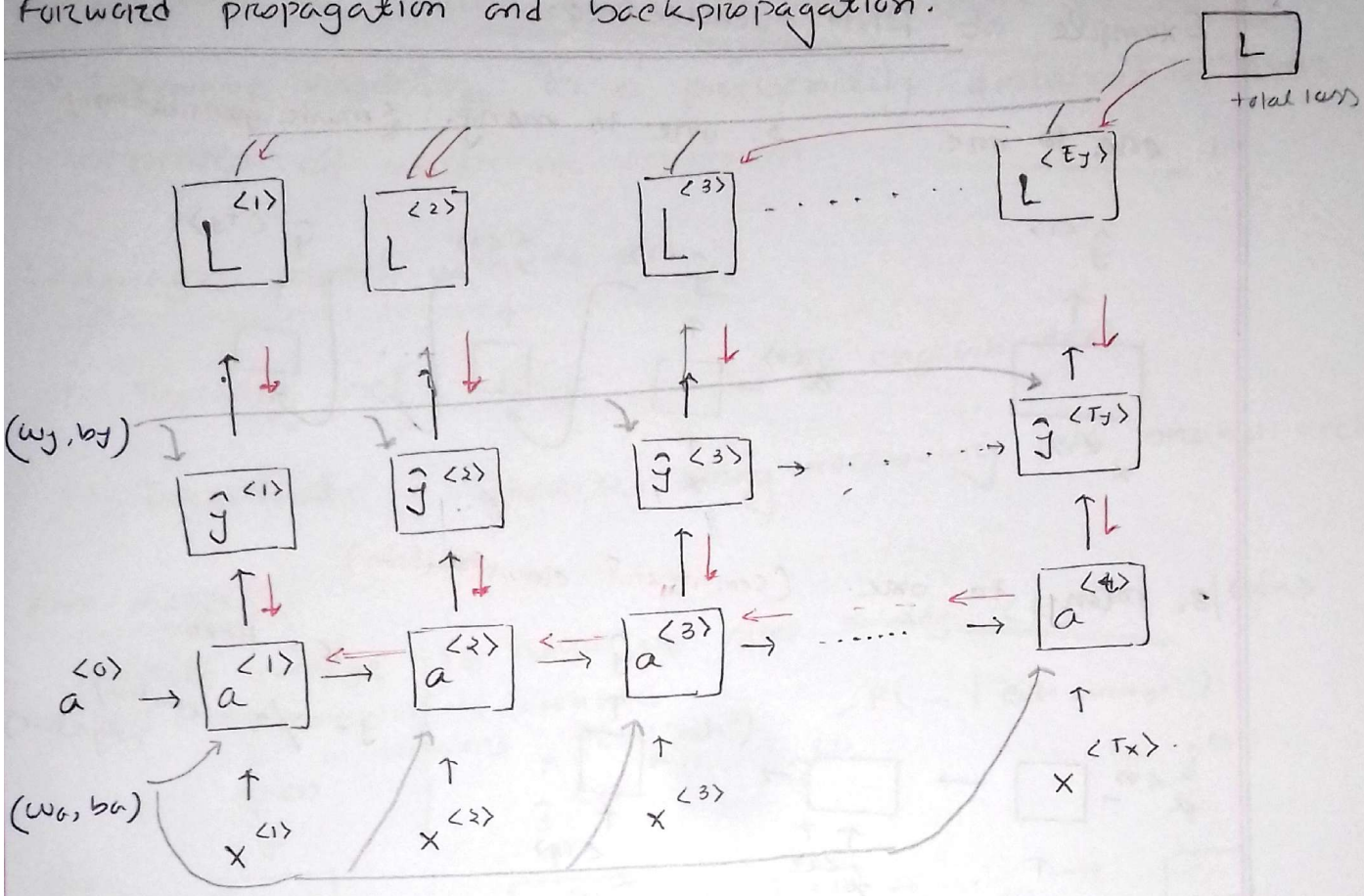
$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}_{10000} = 10100$

calculation

$$\hat{y}^{<t>} = g(w_y a^{<t>} + b_y)$$ m.E.

L-64 Backpropagation through time. [11:35 Am 04.02.19]

Forward propagation and backpropagation.



$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1-y^{<t>}) \log (1-\hat{y}^{<t>})$$

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>}) \quad \text{[total loss]}$$

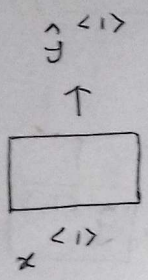
Backpropagation through time.

[Programming Framework automatically implements backpropagation so we do not have to worry much about it]

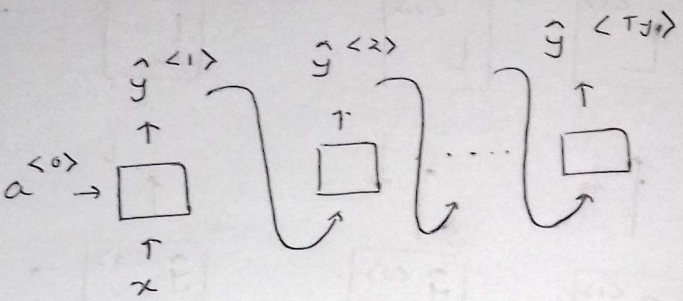
L-5 Different types of RNN. [11.50 Am 04.01.19]

Example of RNN architectures.

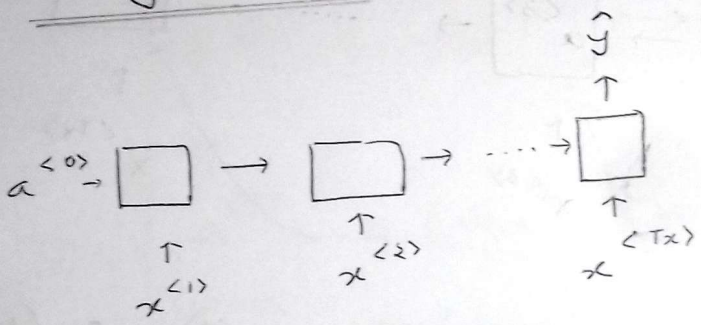
1. one to one



2. one to many. (music generation)



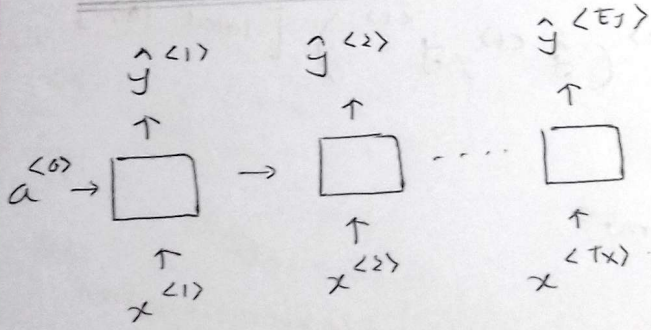
3. many to one. (sentiment classification)



x = text

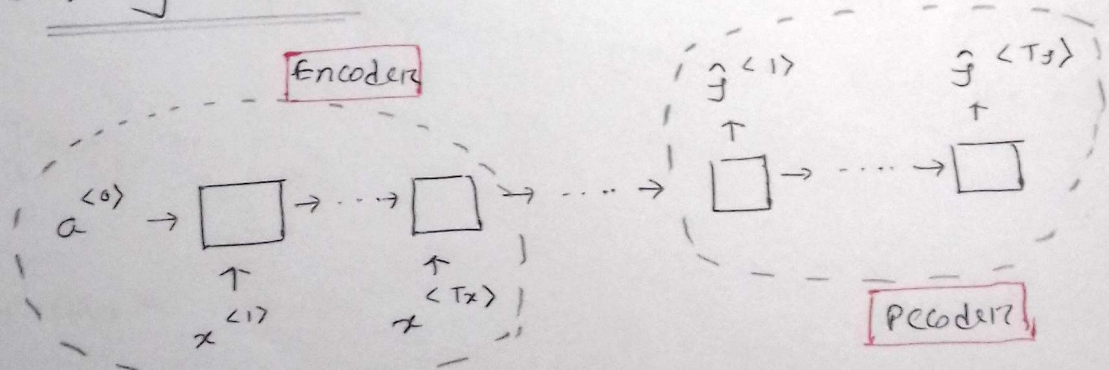
y = 0/1 (positive/negative)

4. many to many (Name entity recognition) ($T_x = T_y$)



$T_x = T_y$.

5. many to many ($T_x \neq T_y$) (machine translation)



L-6 Language model and sequence generation. [12.15 PM 04.01.19]

[Most basic and important task in natural language Processing]

** Language modeling is a probability distribution over sequences of words or sentences.

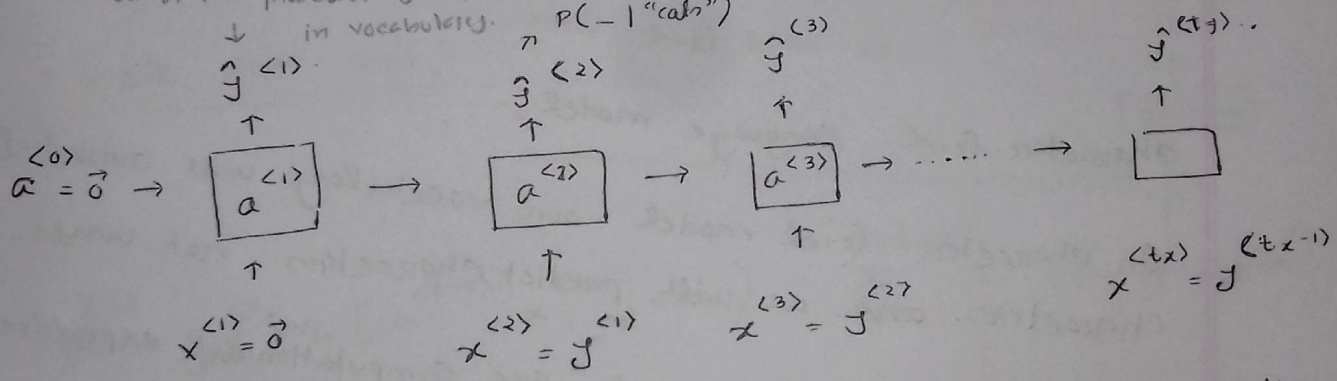
Language model with an RNN :-

1. Training set : large corpus of english text.
2. Tokenize : Tokenize using vocabulary into one-hot vectors.

RNN model :-

Cats average 15 hours of sleep a day. <EOS> | unk>

what is the probability of each word in vocabulary.



[** Each step of RNN will look at some set of preceding word. and try to predict next word.]

Loss Function : $= \mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$.

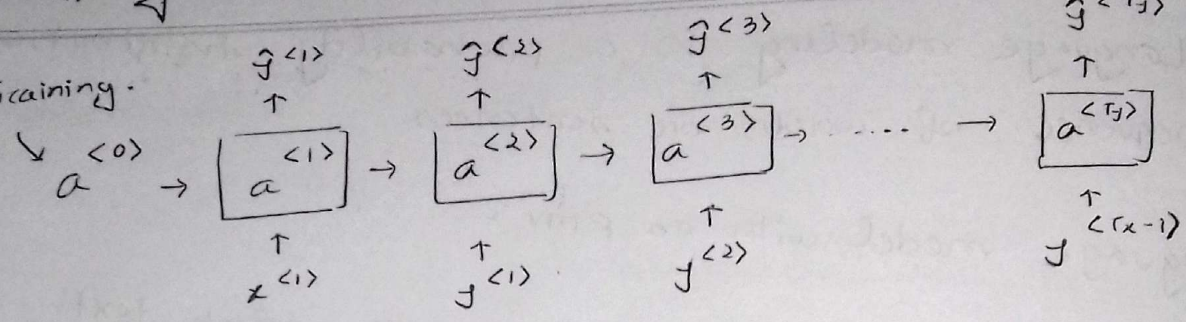
$$\mathcal{L} = \sum_{t=1} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

(Programming exercise - I will implement this)

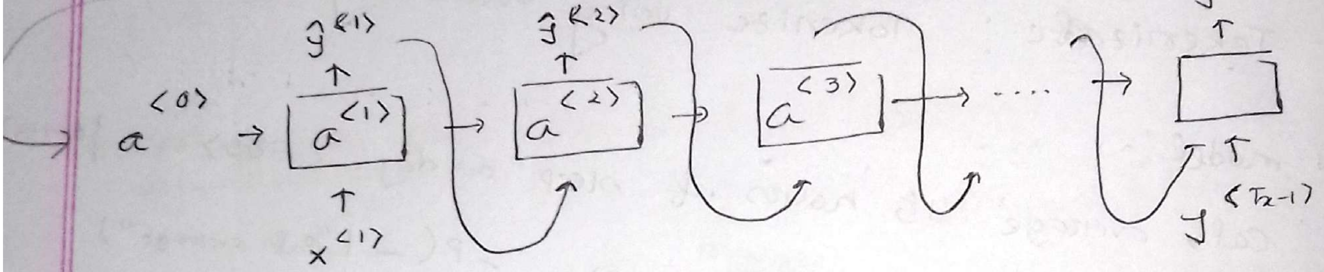
L-7 Sampling Novel sequences. [4.30 pm 04.02.19]

Sampling a sequence from a trained RNN:-

Training.



** Generate randomly chosen sentence form RNN model.



** word level RNN

Character-level language model:-

In character level model our vocabulary will consist of characters. and we will predict characters not words.

** [character-level models are computational expensive and they are not as good as word level]

[Programming exercise માં language model implement કરવું] (slide માં જોયું don't example ત્યાં જોવું)

L-8 Vanishing gradients with RNNs [4.50 pm 04.02.19]

[Basic RNN has vanishing gradients problem].

** [In language model there can be very long term dependencies]

(slide ও RNN এর vanishing gradient দেখানো হয়েছে) [ছবি]

(এই problem টা কিভাবে solve করা যায় তার method আলোচনা করা হবে.

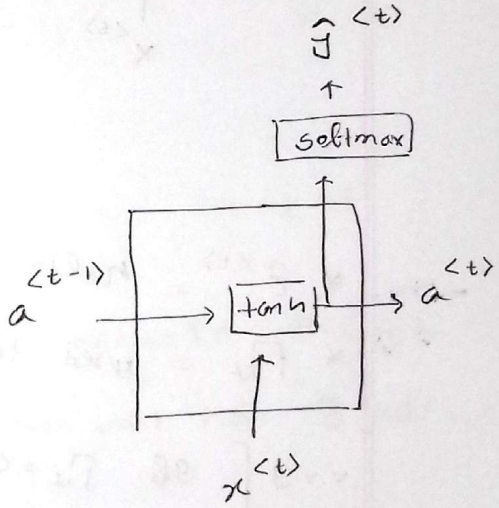
v.v.g

L-9 ~~Gated~~ Gated Recurrent Unit (GRU) [7.00 pm 04.02.19

[Much better in capturing long range connections and helps in vanishing gradient problem]

Simple RNN Unit :-

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



এই paper এটা.

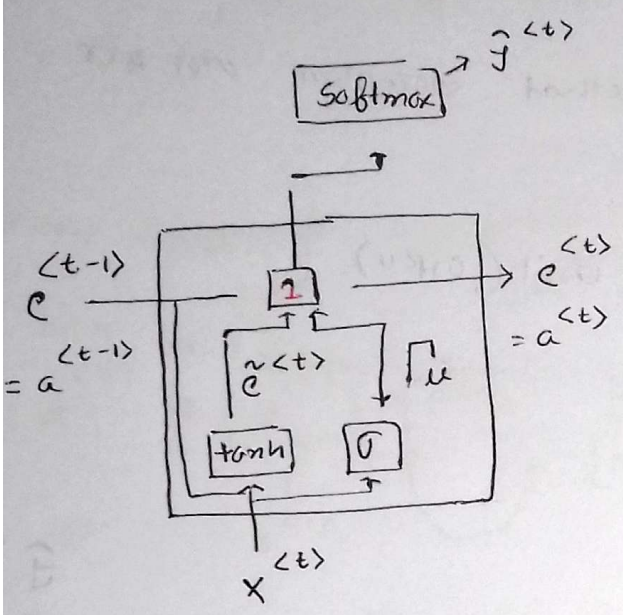
[1] On the properties of Neural Machine Translation.

[2] Empirical Evaluation of Gated Recurrent Neural Networks on Sequence modeling.

* এই paper দুটো পড়তে হবে.

GRU (simplified).

The cat which already ate ...., was full.



c = memory cell.

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

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

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

↓ "update" (gate)

elementwise.

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

** gate is where update value.

* $\tilde{c}^{<t>}$ = holds the value of the memory.

* Γ_u = used to update a particular memory cell.

v.v.y [If $\Gamma_u = 0$ then $c^{<t>} = c^{<t-1>}$ that helps to reduce vanishing gradient problem and learn dependies through longer network]

** dimension of $c^{<t>}$, $\tilde{c}^{<t>}$, Γ_u are same.

Full GRU :-

$\tilde{h} \leftarrow \tilde{c}^{<t>} = \tanh\left(w_c\left[\Gamma_r * c^{<t-1>}, x^{<t>}\right] + b_c\right)$ [new candidate for memory cell]

$u \leftarrow \Gamma_u = \sigma\left(w_u\left[c^{<t-1>}, x^{<t>}\right] + b_u\right)$

$r \leftarrow \Gamma_r = \sigma\left(w_r\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$

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

** This helps to catch long range dependencies and makes RNN better.

v.v.v.g

L-10. Long short Term memory. [7.45 pm 04.02.19

[LSTM allows the error to back propagate through unlimited number of time steps. A LSTM unit has 3 gate,

1. input gate.
2. output gate.
3. forget gate.

It calculates the hidden state by taking combination of these 3 gates.]

[It's unique mechanism helps to overcome vanishing and exploding gradient problem]

[LSTM is more powerfull than GRU]

GRU or LSTM
can we over our
choice are ours.

GRU

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

$$\Gamma_r = \sigma\left(W_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$$

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

$$a^{\langle t \rangle} = c^{\langle t \rangle}.$$

LSTM.

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

$$\Gamma_f = \sigma\left(W_f\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right)$$

$$\Gamma_o = \sigma\left(W_o\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = \Gamma_o * \tanh\left(c^{\langle t \rangle}\right)$$

LSTM in pictures :- (understanding LSTM blog post reader (RNN)).

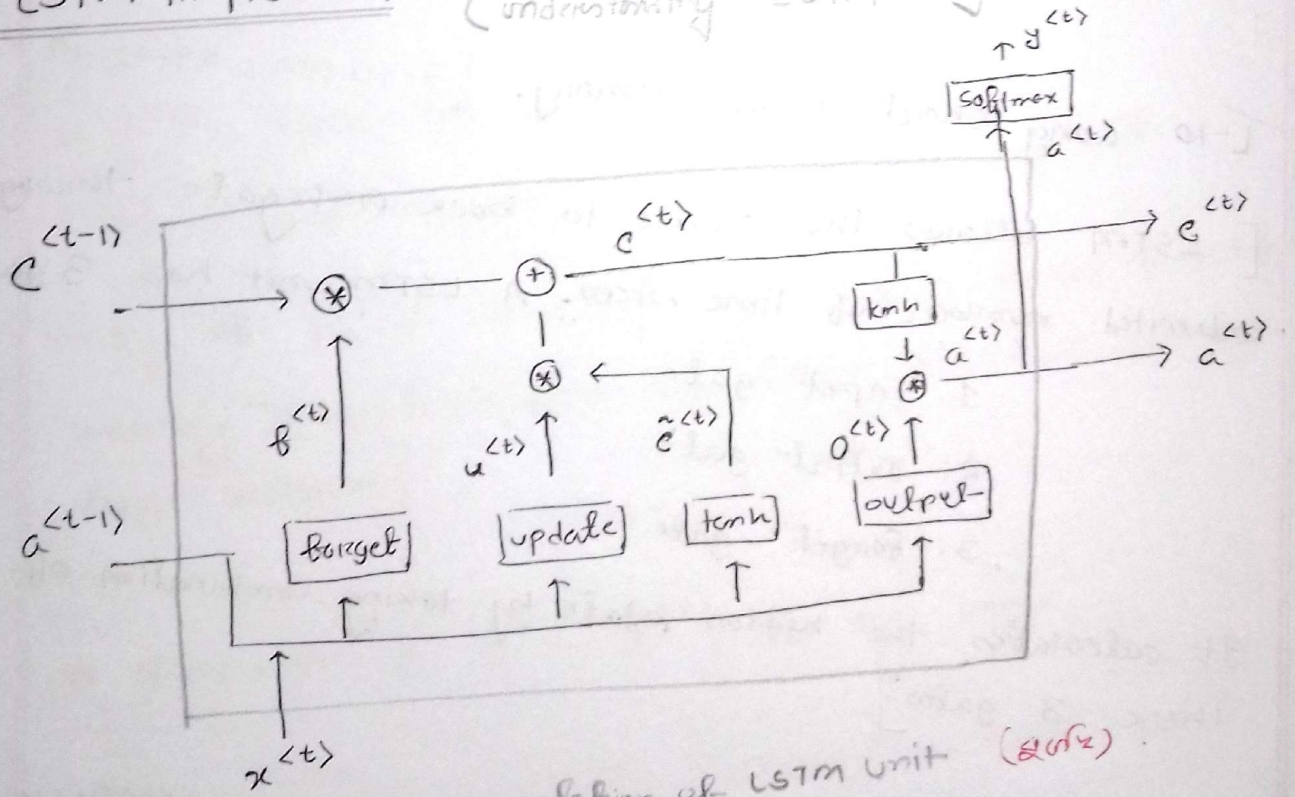


Fig :- Graphical representation of LSTM unit (একটি).

** LSTM unit is good in memorizing previous unit values.

** peephole connection) → used to effect gate output.

L-11 Bidirectional RNN. [8.13 pm 04.01.19]

(BRNN)

[Bidirectional RNN takes information from previous as well as future words]

$$\hat{y}^{<t>} = g\left(w_y\left[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}\right] + b_y\right).$$

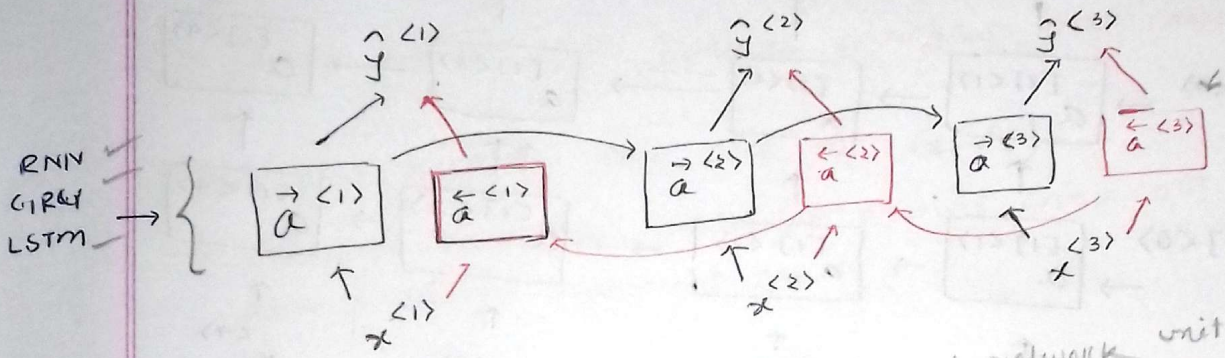


Fig:- Bidirectional Recurrent Neural Network unit.

** [For natural language processing a BRNN with LSTM in a pretty reasonable thing to try.]

Disadvantages:-

1. Need entire sequence of data before prediction anywhere.

(For real time speech recognition it is not very good).

→ But for natural Language Processing where we have the whole data it is really good.

L-12 Deep RNNs. [8:30 pm 04.01.19].

Deep RNN example.

(may have deep ~~recurrent~~ network).

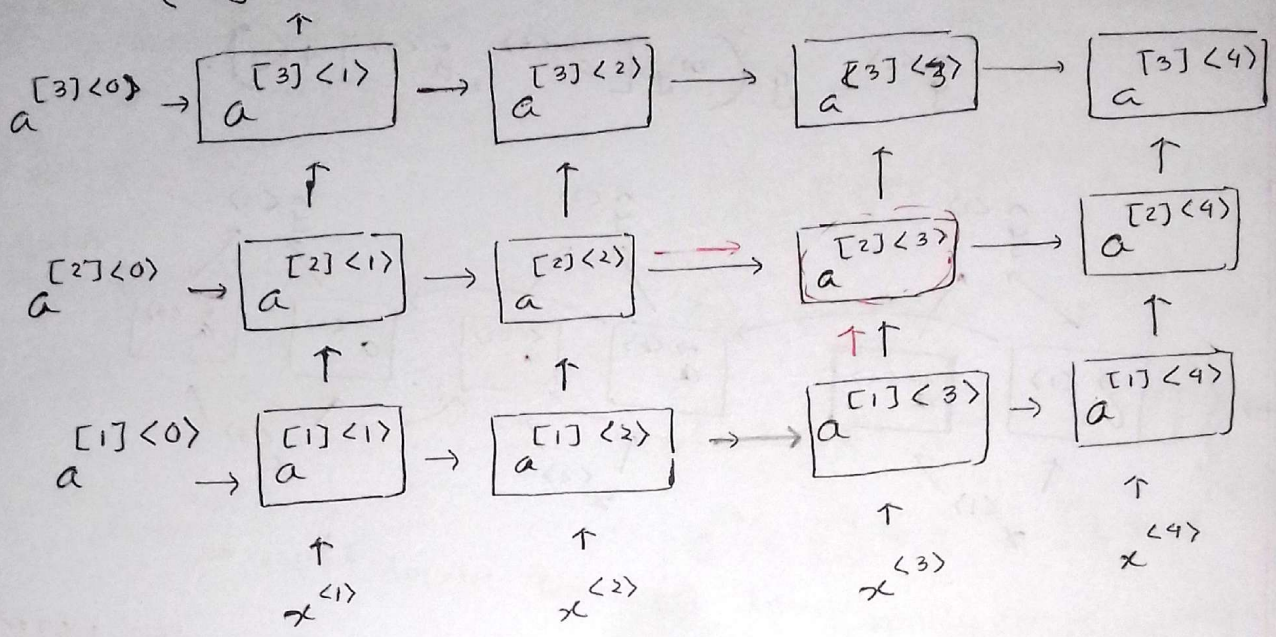


Fig:- A deep RNN architecture.

Left,

$$a^{[2]\langle 3\rangle} = g\left(w_a^{[2]}\left[a^{[2]\langle 2\rangle}, a^{[1]\langle 3\rangle}\right], + b_a^{[2]}\right).$$

** [This week was very much important, learn about Basic RNN, GRU, LSTM, BRNN, and deeper RNNs].

→ ~~এই~~ related papers ~~গুলো~~ insha-Allah ~~করে~~ নিবে.

Programming Assignment :-

1. Building Recurrent Neural Network - step by step.

** আমার এখানে main target হবে input preprocessing করা. তারপর অংশগুলো smaller easily process করতে পারব.

** [paper লেখার সময় course এর লেখাগুলো ব্যবহার করার try করব.]

[Basic RNN ও LSTM block তৈরি করা (আমরাই)]

** (Backward propagation implement করি নাই কেননা এই Programming framework দিয়ে implement করা হয়).

2. Character level language model. (what we will learn).

- How to store text data for processing using an RNN.
- How to synthesize data by sampling predictions at each time step and passing it to the next RNN-cell unit.
- How to build a character level text generation recurrent neural network.
- Why clipping the gradients is important.

** এখান থেকে আমার code কিভাবে implement করব এই ধারণা নিতে হবে.

1. [LSTM text generation by keras team]. ✓

2. [Karpathy's blog post]. ✓

[Shakespeare এর মত sonnet generation হইছে]

[এই idea থেকে suspicious text generation এ use করতে পারি]

Week-2 Natural Language Processing and word embeddings. [07-02-19]

P1: Introduction to word embeddings.

L-1 Word Representation. [12:30 PM 07.02.19]

V = [a, aaron, ..., zulu, <UNK>] |V| = 10,000

1-hot representation:

Man (5391)	woman (9853)	king (4914)	Queen (7157)	apple (456)	Orange (6257)
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	1	1	0
0	0	0	⋮	⋮	0
1	0	1	1	⋮	1
⋮	⋮	⋮	⋮	0	⋮
0	1	0	0	0	

If our algorithm learns, I want a glass of apple Juice.

It can not generalize, I want a glass of orange Juice.

[Because Product of apple and orange is zero]

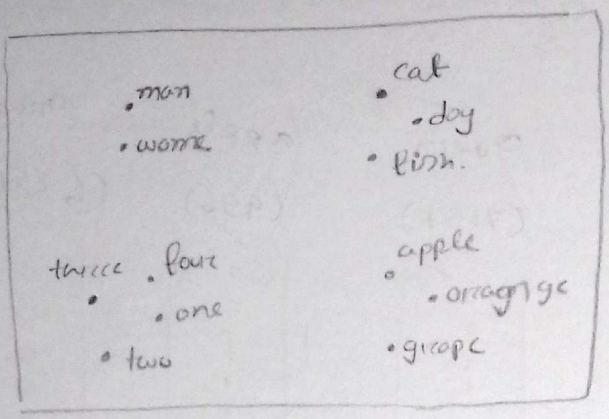
Featurized representation: word embedding:

Features	Man (5391)	woman (9853)	king (4914)	Queen (7157)	apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.95	0.95	-0.001	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
⋮	↓	e_{9853}				
	e_{5391}					

[It helps us to generalize better over different things]

Visualizing word embeddings.

[ ] visualizing Data using t SNE.

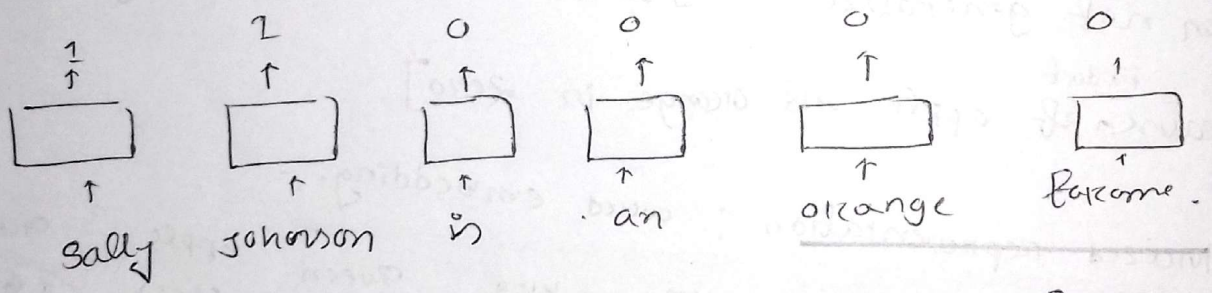


300D
↓
2D

[most important idea in Natural Language processing]

L-2 Using word embeddings. [12.55 PM 07.02.19]

Named entity recognition example.



Robert Lin is an apple farmer.

→ we can apply transfer learning there.

(1B - 100B words) → to learn similar word.

100k → words (small training set)

Transfer learning and word embeddings.

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 (say, 100k words) [may use dense vector instead of one-hot vector]
3. Optional:- Continue to finetune the word embeddings with new data.

word embedding relation to face encoding.

[It is quite similar which creates encoding for the input].

[one difference is that in face encoding neural network ~~may~~ can have unknown face,
But in word embedding we had a vocabulary and has a fixed encoding for the words in vocabulary.]

L-3 Properties of word embeddings. [1:25 pm 07.02.19]

word embeddings are useful for analogy reasoning.

Such as man : woman, king → ?

$$e_{man} - e_{woman} \approx e_{king} - e_{w}$$

Find the word w :- $\arg\max_{w} \text{sim}(e_w, e_{king} - e_{man} + e_{woman})$

[Choose the word which maximizes similarity]

Cosine Similarity (most useful)

$$\text{Sim}(u, v) = \frac{u^T v}{\|u\|_2 \, \|v\|_2}$$

$$\text{Sim}(e_w, e_{king} - e_{man} + e_{woman}).$$

[Slide এ এগুলো লেখা ছিল, আলাদা করে slide দেখানো লাগে]

L-4 Embedding matrix. [1:40 pm 07.2.19]

[matrix নিয়ে. Easy] (আগেরটা না বুঝলেও চলবে).

E . O_j . $\therefore E \cdot O_j$.

$[300, 10K]$ $[\;]$ $[300, 10K]$ $[10K, 1]$ $= [300, 1]$

$[300, 10K]$ → Feature, vocab-size.

$[\;]$ → one-hot vector for j-th element in vocabulary.

$[300, 1]$ → Feature matrix for a word.

[In practice, use specialized function to look up an embedding].

P2: Learning Word Embeddings : Word2Vec & Glove.

L1: Learning word embeddings. [7.05 pm 07.02.19]

I	want	a	glass	of	orange	Juice.
4343	9665	1	3852	6163	6257	

I O_{4343} → E → e_{4343}

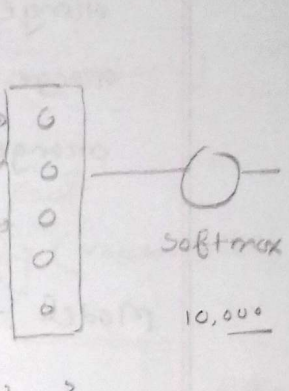
want O_{9665} → E → e_{9665}

a O_1 → E → e_1

glass O_{3852} → E → e_{3852}

of O_{6163} → E → e_{6163}

orange O_{6257} → E → e_{6257}



one-hot vector

Embedding vector

word embedding

Context / target pairs.

[Context is embedding dimension that helps to learn embedding vectors]

→ Last 4 words. [Look at the past 4 word and predict the next word]

→ 4 words on left & right [Look at 4 words on both side of Predicting word].

→ Last 1 word.

[Neural 1 word] – Skip gram model.

Here we use context to predict target word and eventually learn word embeddings.

L-2 Word2vec [7.30 pm 07.04.19]

Skip-grams :-

I want a glass of orange juice to along with my cereal.

Context	target
orange	juice
orange	glass
orange	my.

** Here we randomly chose a word as context word.

** Randomly chose another word as target word within a context window

Model :-

Let, vocabulary size = ~~10,000~~ 10,000

Context c ("orange") → Target t ("juice")

6257 - position 4834 - position.

$$O_c \longrightarrow E \longrightarrow e_c \longrightarrow O \longrightarrow \hat{y}$$

O_c [one-hot vector for context word] → E [embedding matrix] = $E \cdot O_c$ [embedding vector for context word] → O softmax → $\hat{y}$

$$\text{Softmax}: \quad p(t|c) = \frac{e^{\theta_t^T \cdot e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T \cdot e_c}}$$

[θ_t = Probability associated with output target word t

e_c = embedding vector for context word c]

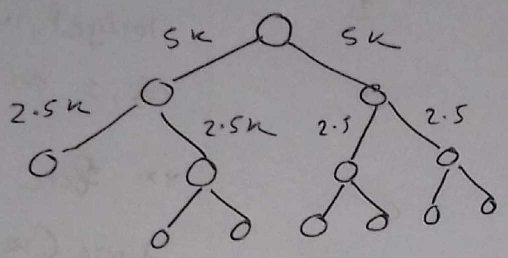
$$\text{Loss}: \quad L(\hat{y}, y) = -\sum_{i=1}^{10000} y_i \cdot \log \hat{y}_i \qquad y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \text{ - one-hot vector}$$

[This is called skip-gram model.]

Problems with softmax classification.

$$P(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$ → High computational to compute this sum value.

One solution is Hierarchical softmax :- (similar to binary search)



Computation cost becomes = log (vocab-size)

[To build a hierarchical softmax classifier we can use different Heuristics.] (Paper [illegible])

How to sample context c ?

** we can sample context c uniformly randomly but then the frequent words (the, of, a, and, to ....) will be update and words (orange, juice, ...) will not update much.

** So we use some heuristics to sample some common and uncommon words.

[The key problem of this algorithm is that computational cost of softmax function]

[3] mikolov paper

v.v.g

L-3. Negative Sampling. [8.10 pm 07.02.19]

[This algorithm is used to solve the problem of skip-gram model]
[paper পড়তে হবে].

Defining a new learning problem.

I want a glass of orange juice to go along with my cereal.

v.v.g.

Context	word	target
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

Number of random words from vocabulary (k) — rows orange/king to orange/of

Context and word → x; target → y

** pick a context word, pick a target word and give it a label of 1 as it is positive.

** For some number of time (k) for the context word chose random word from vocabulary and label as 0 to negative

** → Now we have supervised learning algorithm where we input x (pair of words) and predict y (target label)

$k = 5-20$ smaller dataset

$k = 2-5$ large dataset.

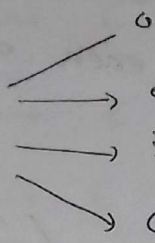
(orange and purple)

Model:-

Softmax: $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$

$P(y=1 \mid c,t) = \sigma(\theta_t^T e_c)$ ← logistic regression.

orange 6257 → O_{6257} → E → e_{6257}



vocab-size ↓

10000 binary classification Problem using logistic regression.

Selecting negative examples.

[After selecting positive word for the context word, How do we select negative samples]

→ Probabilities of words can be counted, (unigram distribution)

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_i)^{3/4}}$$

[$f(w_i)$ = Frequency of word w_i in your training set]

[English - If not Pre-trained word-embeddings vector that are, you can download and use it]

L-4 Glove [8.45 pm 07.02.19]

Glove (global vectors for word representation)

A word-embedding model which is "count-based".

X_{ij} = # times i (t) appears in context of j (c).

(t, c ; c, t)

** X_{ij} is the count which captures how often the word i, j appears close to each other.

Model:-

$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij}) \left(\theta_i^T \cdot e_c + b_i + b_j - \log X_{ij} \right)^2$$

($i \to t$, $j \to c$; $\theta_i^T \cdot e_c \to \theta_t^T \cdot e_c$)

$f(X_{ij})$ → weighting term; $f(X_{ij}) = 0$ if $X_{ij} = 0$.

[This weighting term ensures that most frequent words (less weight) and infrequent words do not get much weight

** (paper পড়তে হবে).

[Slide এ দেওয়া [illegible] আছে]

P3:- Applications using word-embeddings.

L-1 Sentiment classification. [9.35 pm 07.02.19]

[one of the challenges is that we may not have large training dataset. but word embedding can be useful with less data.]

Simple sentiment classification model.

The	dessert	is	excellent
8928	2468	4694	3180

The → O_{8928} → E → e_{8928}

dessert → O_{2468} → E → e_{2468}

is → O_{4694} → E → e_{4694}

excellent → O_{3180} → E → e_{3180} (300)

(e_{8928}, e_{2468}, e_{4694}, e_{3180}) → Avg → O → $\hat{y}$ softmax

[** The problem is that it does not consider orders of words. It does not take in account semantic meaning.

RNN for sentiment classification:-

$a^{<0>}$ → $a^{<1>}$ → $a^{<2>}$ → $a^{<3>}$ → ..... → $a^{<10>}$ → softmax → $\hat{y}$

$a^{<1>}$: e_{1852} ← E ← O_{1852} (Completely)

$a^{<2>}$: e_{4966} ← E ← O_{4966} (Receiving)

$a^{<3>}$: e_{4427} ← E ← O_{4427} (in)

$a^{<10>}$: e_{336} ← E ← O_{330} (ambience.)

[This does help in couting semantic meaning].

(BLP ৫ গিট নিয়ে future এ কাজ করব)

L-2 Debiasing word embeddings [11:10 pm 07.02.19]

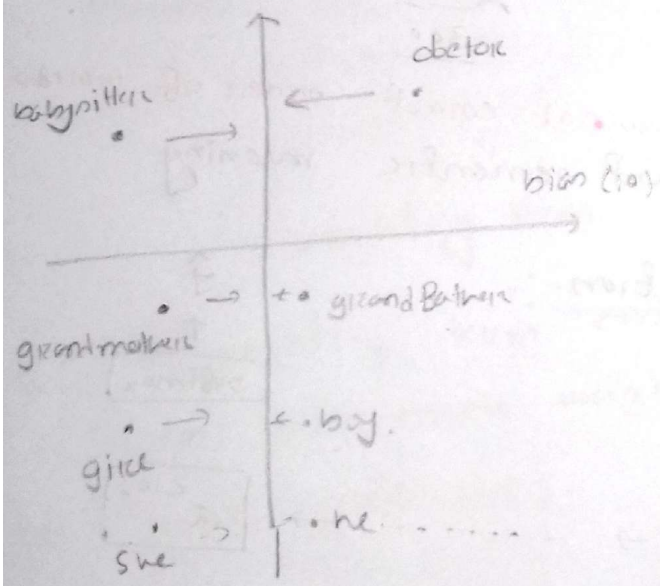
word embeddings can reflect gender ethnicity, age, sexual orientation and other biases of the text used to train the model.

[we must try to diminish this type of biases as machine learning takes really much important decisions].

[BIT নিয়ে paper আছে] [Bolukbasi et. al. 2016]

Addressing bias in word embeddings.

1. Identify bias direction.



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

3. Equalize pairs.

(Future idea নিয়ে কাজ করব).

Programming Assignment :-

Operations on word vectors.

- Cosine similarity a good way to compare similarity between pairs of word vectors.
- For NLP applications using a pre-trained set of word vectors from the internet is often a good way to start.

Emojify ! (Same as sentiment classification)

([illegible] sentiment classification [illegible] idea [illegible])

[** [illegible] word embedding glove vector [illegible] [illegible] [illegible]
[illegible] [illegible] [illegible] [illegible]]

[[illegible]]

r.r.g:
** [word embedding [illegible] Blog [illegible] implement [illegible]]
[illegible] Blog [illegible]].

** Coursera Course [illegible] embedding layer implement [illegible].

** Important points.

→ In a NLP task where training set is small, word embeddings can be used to help your algorithm.

→ Word embeddings allow a model to work on words in the test set that may not even appeared in training set.

→ To use mini-batches, the sequences need to be padded so that all the examples in mini-batch have same-length.

→ Design your embedding matrix carefully.
  - Can use pretrained values or initialize in your training set

→ After embedding use your Deep learning architecture.

viva [আমার কাছে এই সপ্তাহ এর exercise really important].

week-3 : Sequence models & Attention mechanism.

P1 : Various Sequence to Sequence Architectures.

L-1 Basic Models. [10:35 Am 09.02.19]

Basic Machine translation model.

→ देखें paper जो reference दिए गए हैं।

Automatic image captioning.

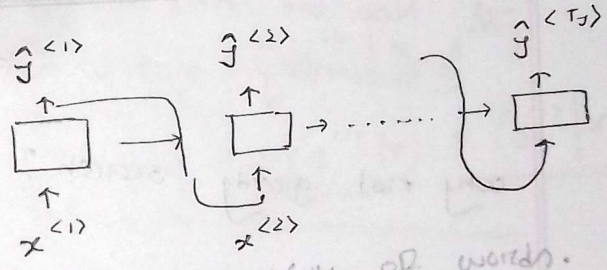
→ AlexNet आदि जैसे जो Decoder use करें automatic image caption करने के लिए

→ Decoder circuit जो Normally RNN आदि के हैं।

L-2 Picking the most likely sentence. [10:50 Am 09.02.19]

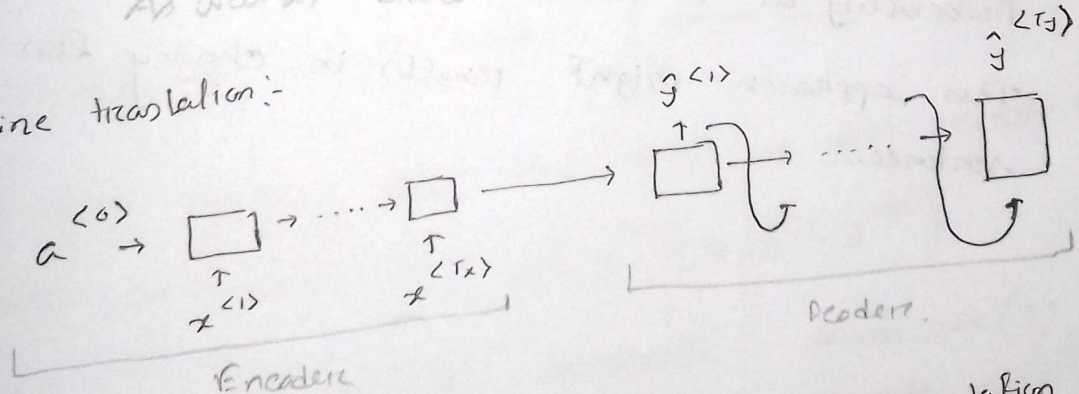
Machine translation as building a Conditional Language model.

Language model :-



[Language model helps to calculate probability of words.
As well as create make sentences.

Machine translation :-



** Encoder circuit figures out some representation of the input sentence and feed it to decoder circuit.

"Condition Language model" $P(y^{<1>}, \cdots y^{<T_y>} \mid x^{<1>}, \cdots x^{<T_x>})$ Conditioned

↳ Probability of an English sentence one some French sentence.

Finding the most likely translation.

x = a french sentence.

$P(y^{<1>}, \dots y^{<T_y>} | x)$ = Probability of corresponding english sentence.

we will not sample randomly. we will take the english sentence that maximizes the the probability.

$$\arg\max_{y^{<1>}, \dots y^{<T_y>}} P(y^{<1>}, \dots y^{<T_y>} | x)$$

→ Now we need an algorithm which calculates this probability efficiently, that is (Beam Search).

why not greedy search?

→ In greedy search it taken the word which maximize probability at the current step.

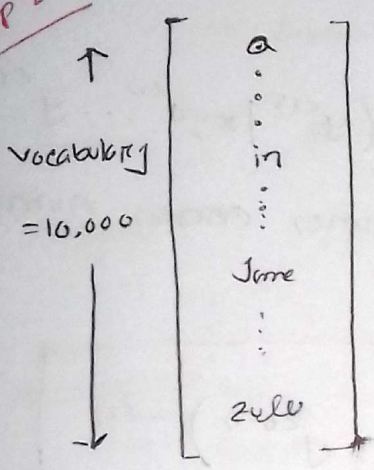
This approach might results in chosing less optimal sentence.

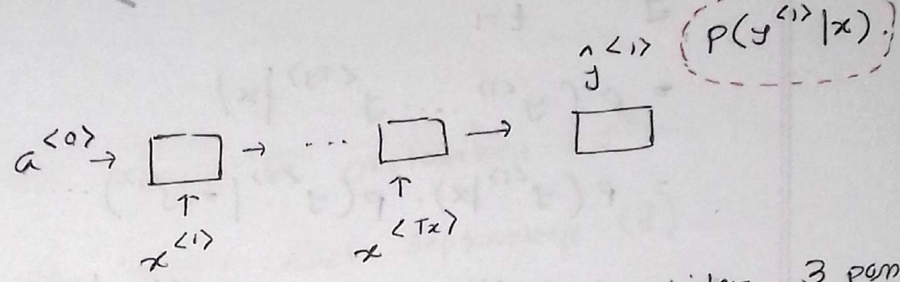
L-3 Beam Search [11.10 Am 09.02.19]

[Beam search is a algorithm which try to maximize conditional Probability in machine translation]

<u>Beam Search algorithm.</u> (Slide এ ছবি দেখে বুঝতে হবে)

Step 1





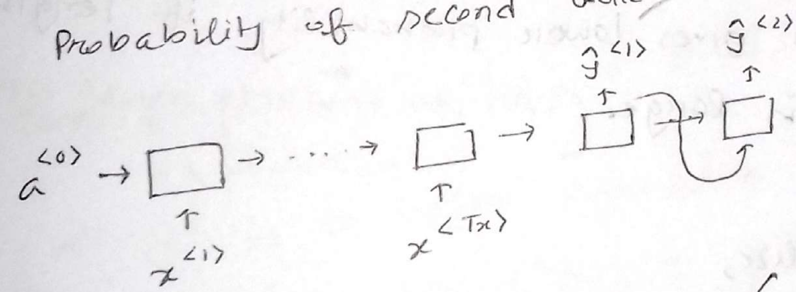
$B = 3$ (beam width), it will consider 3 possible output at a time.

Step-2

→ Now after selecting first 3 words we will calculate the Probability of second word,

$P(y^{<2>} | x, y^{<1>})$

$= P(y^{<1>} | x) \cdot P(y^{<2>} | x, y^{<1>})$



Number of possible words = (Beam size × vocab_size).

** After selecting first two word we will select the 3rd word which maximize $P(y^{<3>} | x, y^{<1>}, y^{<2>})$.

in each step we will memorize beam size (3) possible output which is ~~calculat~~ used to calculate next step.

[পুরোটা ব্যাপারটা - ভালো বুঝিনি না].

[Slide এ আরও একটু - ভালো করে দেখা লাগবে]

L-4 Refinements to Beam search. [11:35 Am 09.02.19]

Length normalization. (garlic)

$$\underset{y}{\text{argmax}} \prod_{t=1}^{T_y} p\left(y^{<t>} \mid x, y^{<1>}, \ldots y^{<t-1>}\right) \quad -(1)$$

$$= p\left(y^{<1>}, \ldots y^{<T_y>} \mid x\right)$$

$$= p\left(y^{<1>} \mid x\right) \cdot p\left(y^{<2>} \mid x, y^{<1>}\right) \cdots \cdots p\left(y^{<t>} \mid x, y^{<1>} \ldots y^{<T_y-1>}\right)$$

As probabilities are small this sometimes causes numerical underflow. we can avoid this by,

$$\underset{y}{\text{argmax}} \sum_{y=1}^{T_y} \log p\left(y^{<t>} \mid x, y^{<1>}, \ldots y^{<t-1>}\right) \quad -(2).$$

The objective function given lower probability if length of the sentence (T_y) is large. (As we multiply more small probabilities).

This can be normalize,

$$\left[\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log p\left(y^{<t>} \mid x, y^{<1>}, \ldots, y^{<t-1>}\right)\right]$$

hyper parameter

$\alpha = 0.7$

$\alpha = 0.$

$\alpha = 1$

Beam width B (parameter)

- large B: better result, slower.
- Small B: worst result, faster.

** Beam search runs faster but is not guranteed to find exact maximum for $\underset{y}{\text{argmax}}\, p(y|x)$.

L-5 Error analysis in beam search. [11.50 Am , 09.02.19]

[Slide এ কিছু technique দিয়ে আলোচনা করেছে] (বুঝিনি).

In machine translation, we have two main components,

→ RNN. [Computes $p(y|x)$]

→ Beam search.

Human :- Jane visits africa in september. (y^*)

Algorithm :- Jane visited africa last september ($\hat{y}$).

Case 1 :- $p(y^*|x) >> p(\hat{y}|x)$.

→ Beam search chose $\hat{y}$. But y^* attains higher $p(y|x)$.

Conclusion :- Beam search is a fault.

Case 2 :- $p(y^*|x) \leq p(\hat{y}|x)$

→ y^* is a better translation than $\hat{y}$. But RNN predicted

predicted $p(y^*|x) < p(\hat{y}|x)$

Conclusion :- RNN model is a fault.

(বুঝিনি easy লাগছে) [error analysis কেন? কেন করা হয় for prob].

L-6 Bleu Score. [12:05 pm 09.02.19]

[After automatic generation of machine translation Bleu score helps to determine how good the translation is]

Bleu score is calculated on machine translation (MT) output and human references.

Example :- Ref 1 :- The cat is on the mat

Ref 2 :- There is a cat on the mat.

MT out : the the the the the The the.

[বিভিন্ন ধরনের Bleu score calculation method আছে] (easy ones).

** Bleu score on bigram. (slide এ দেখানো হয়েছে)

** Bleu score on unigram :-

$$P_1 = \frac{\sum_{\text{unigram} \in \hat{y}} \text{Count}_{clip}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})}$$

$$P_n = \frac{\sum_{\text{n-gram} \in \hat{y}} \text{Count}_{clip}(\text{n-gram})}{\sum_{\text{n gram} \in \hat{y}} \text{Count}(\text{n-gram})}$$

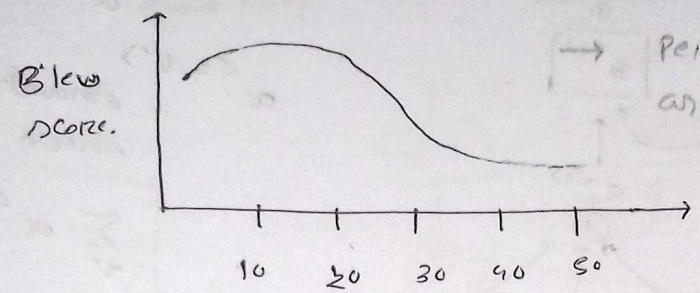
P_n = Bleu score on n-gram only.

Combined Bleu score : $\exp\left(\frac{1}{m}\sum_{n=1}^{m} P_n\right)$

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

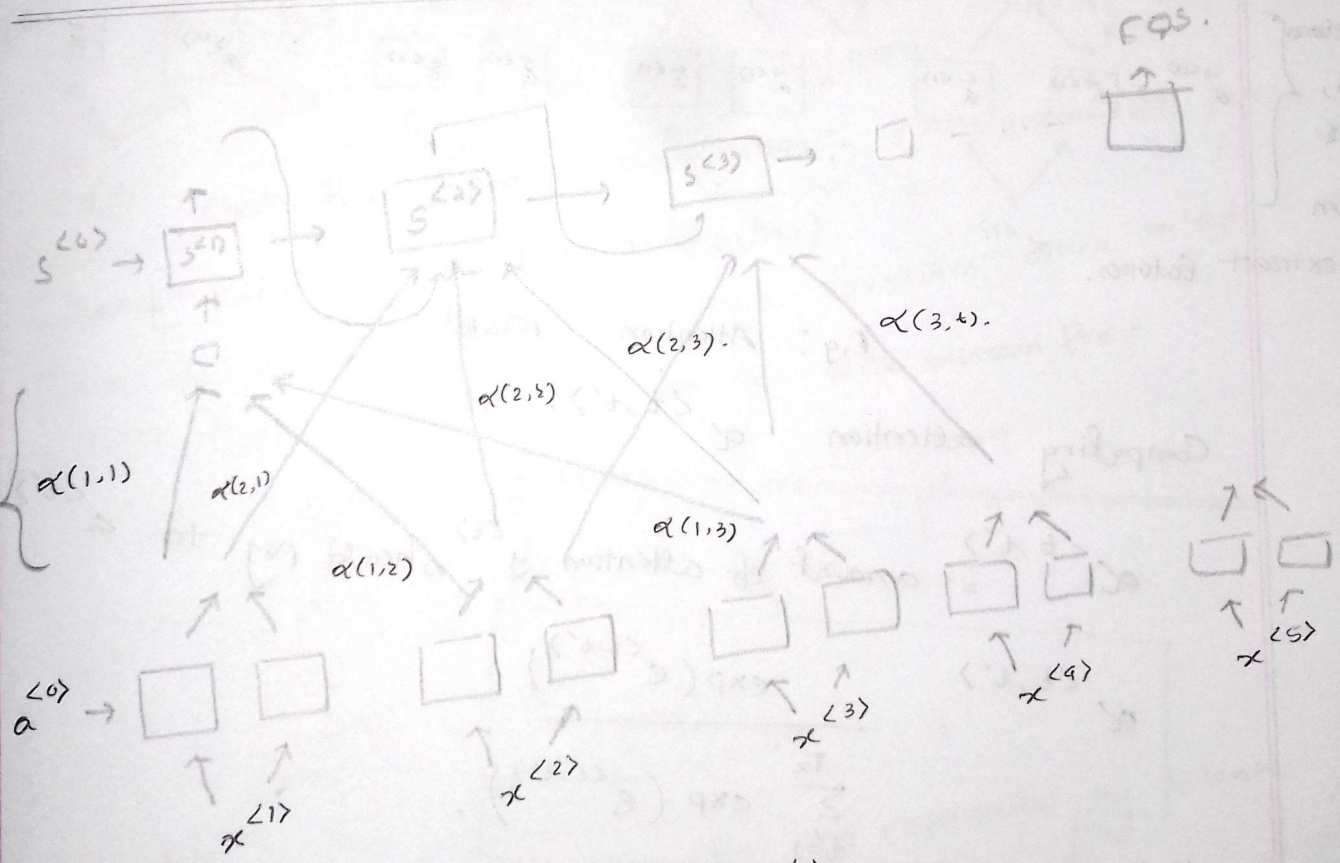
L-7 Attention model intuition [1.00 pm 09.02.19].

[Encoder-Decoder model (Previous model) performs well on short sentences but do not perform much well on long sentence]



** Attention model helps to perform well on long sentence as well.

Attention model intuition.



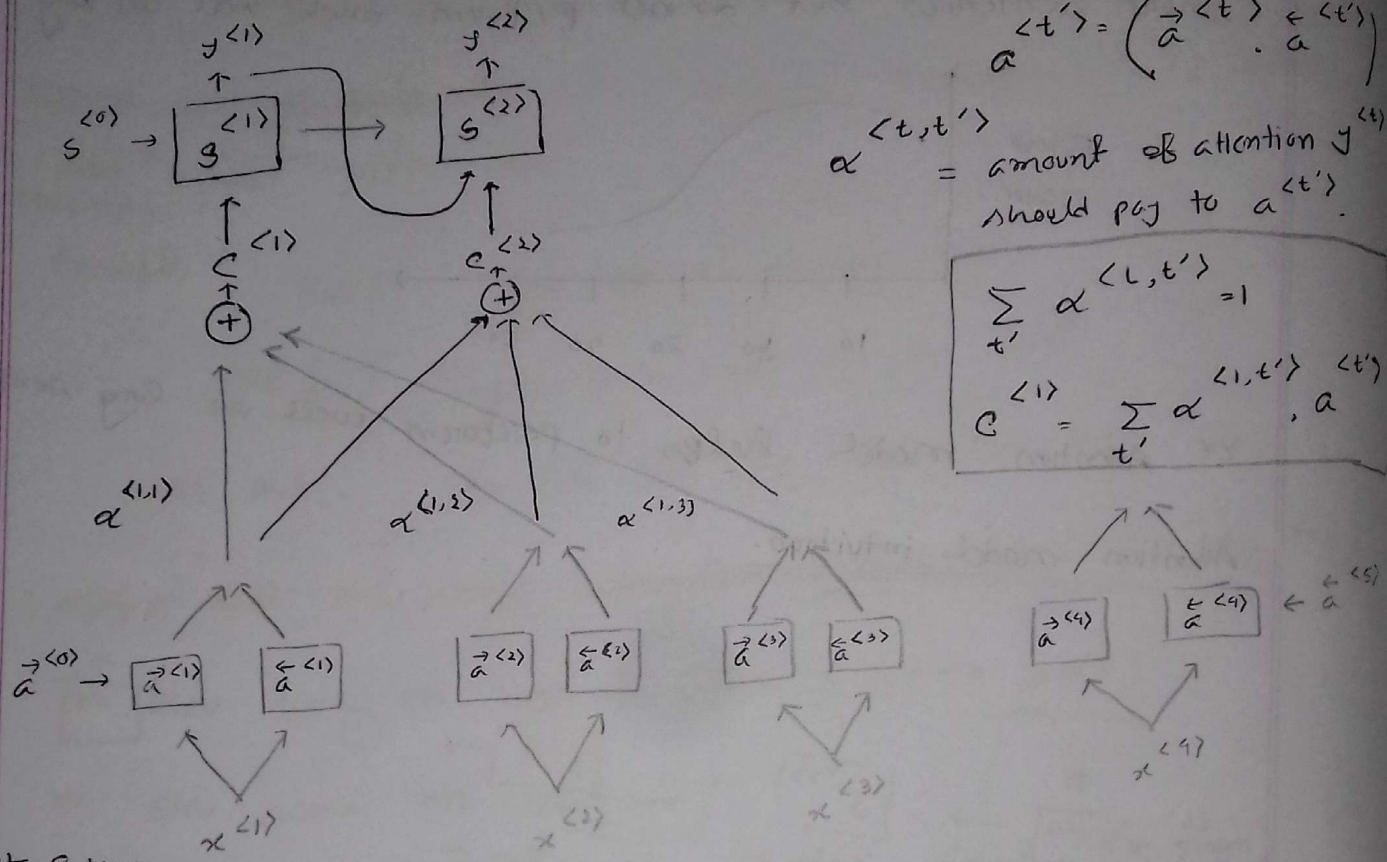
Attention weight = $\alpha(t, t')$

→ when you generating t english word on how much you should give attention to t' th french word.

L-8 Attention model. [1:20 pm 09.02.19]

[Recent সময়ে খুব Attention model এর paper টা অনেক হচ্ছে].

(slide এ যা যা লেখা আছে)



$$a^{<t'>} = \left(\overrightarrow{a}^{<t'>}, \overleftarrow{a}^{<t'>}\right)$$

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$.

$$\sum_{t'} \alpha^{<1,t'>} = 1$$

$$c^{<1>} = \sum_{t'} \alpha^{<1,t'>} a^{<t'>}$$

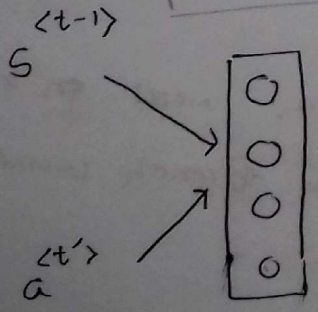
Bidirectional RNN, GRU or LSTM to extract features.

Fig: Attention model.

Computing attention $\alpha^{<t,t'>}$.

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$



** visualizing $\alpha^{<t,t'>}$

[cost is high]

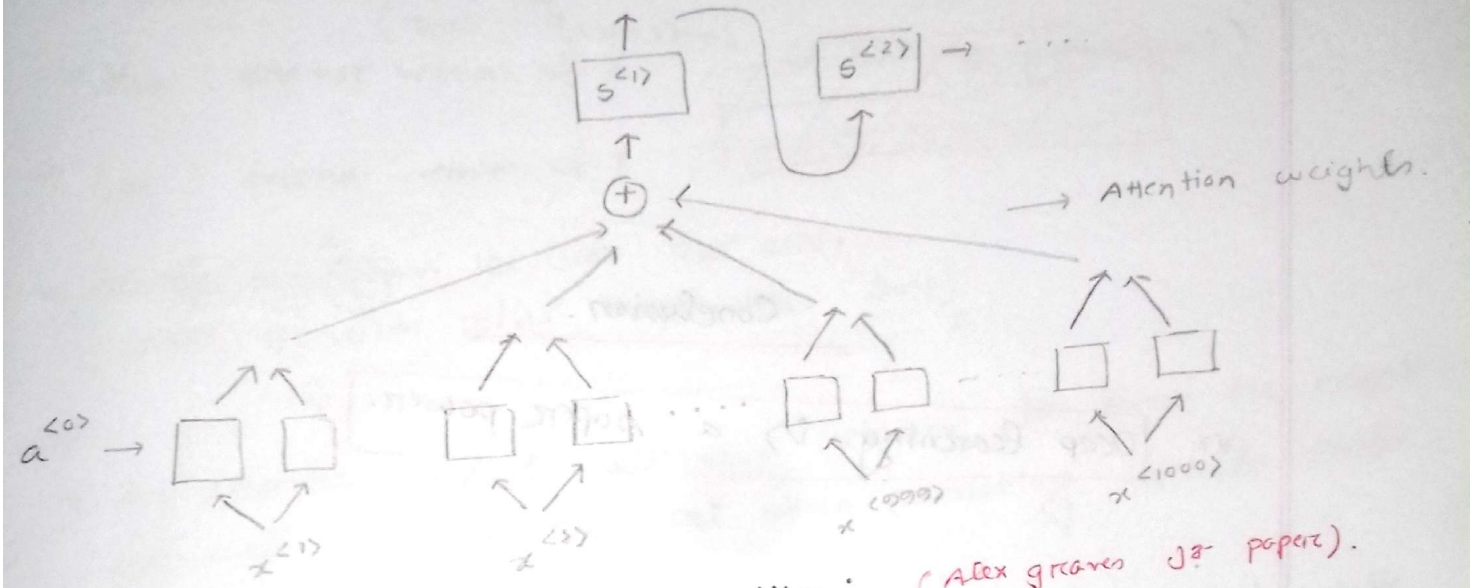
P2: Speech recognition. Audio data.

L-1 Speech recognition. [4.40 pm 09.02.19]

Attention model for speech recognition :-

x → y

audio clip → transcript

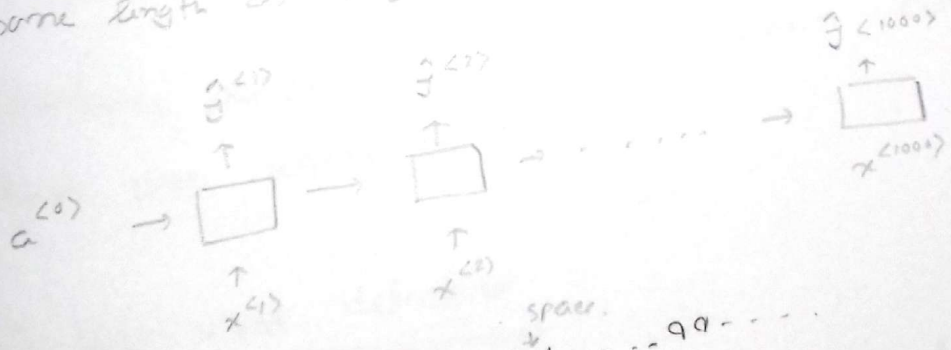


CTC cost for speech recognition :- (Alex graves JA paper).

(Connectionist temporal classification)

[CTC for predicting correctly in speech recognition where output is not same length as input]

** The quick brown fox.



space.

blank

ttt - - - h ..... ee .... ␣ - - - - qq - - - -

*rule :- collapse repeated character not separated by blank.

L-2 Trigger word detection. [5.00 pm 20.02.17]

Trigger word detection algorithm :-

(Almost optimized algorithm is not found yet)

(slide এ দেখে নিতে হবে)

(Programming exercise এ implement করা)

Conclusion.

** **Deep learning is a super power.**

Programming Assignment :-

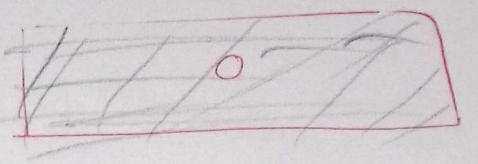
1. Translating human readable dates into machine readable dates.

X → Processed version of human readable dates (m, T_x)

Y → Processed version of machine readable dates (m, T_y).

→ X_{oh} : one-hot version of X

→ Y_{oh} : one hot version of Y



** Attention mechanism এর use করে।
(LSTM, Bi-LSTM ইত্যাদি ব্যবহার করে Simple)

→ An attention model allows a network to focus on most relevent parts of the input when producing specific part of output.

→ A network ~~can~~ using an attention mechanism can translate from input of length T_x to output length T_y where T_x and T_y can be different.

[Attention model are useful to learn complex mappings from one sequence to another].

2. Trigger word detection.

→ Structure a speech recognition project.

→ Synthesize and process audio recordings to create train/dev datasets.

→ Train a trigger word detection model and make predictions.