Week 11: Recurrent Neural Networks 2

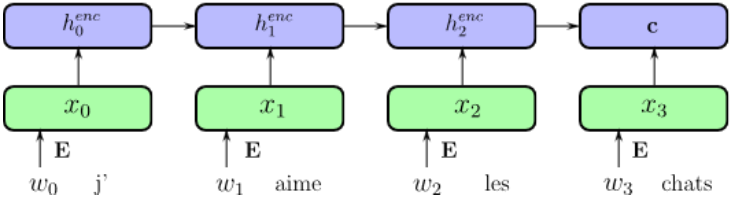
Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

09 May, 2019

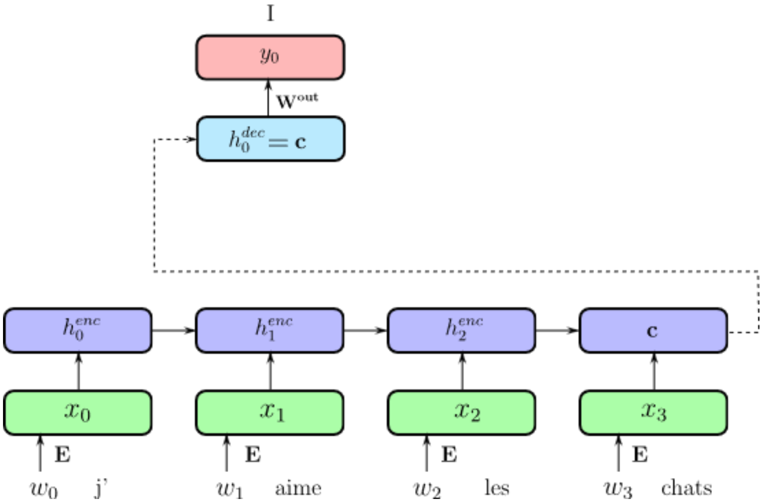
Encoder-decoder model

- Encoder: encode a source sequence into a fixed-length vector



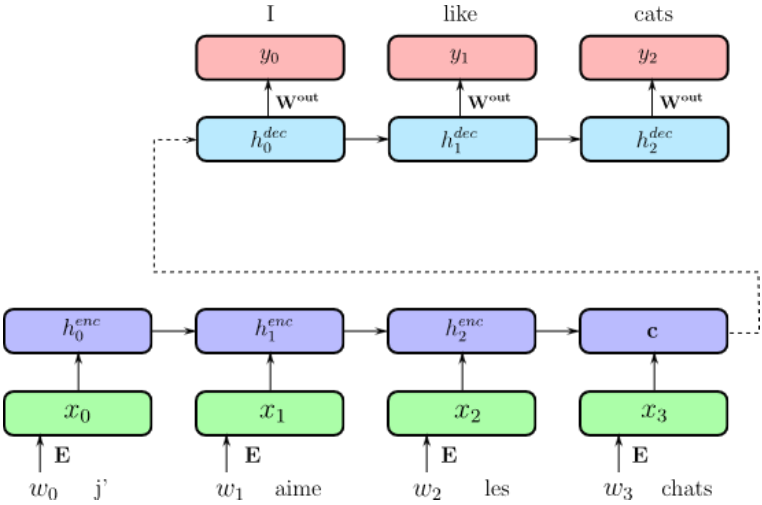
Encoder-decoder (cont')

- Decoder: encoder's last hidden state as initial hidden input



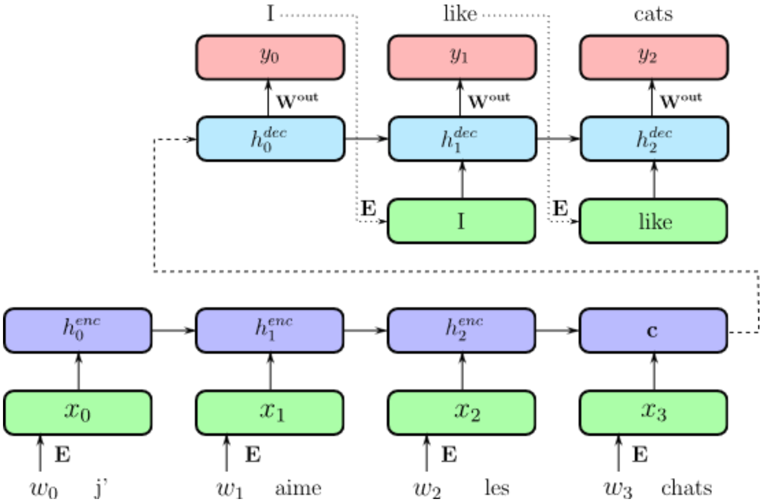
Encoder-decoder (cont')

- Decoder and encoder are often two different LSTMs

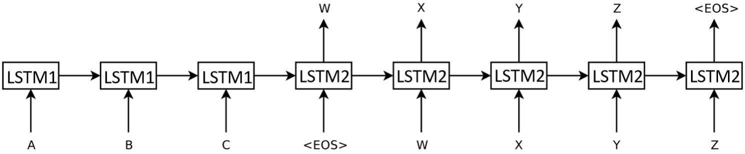


Encoder-decoder (cont')

- Decoder has two inputs at each step!

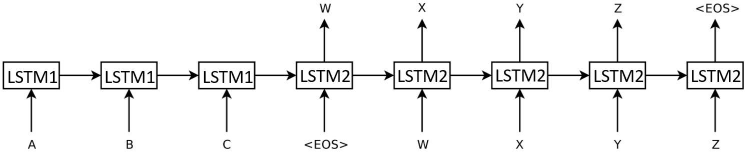


Encoder-decoder for machine translation



- Why output at prev time step as current input in decoder?

Encoder-decoder for machine translation



- Why output at prev time step as current input in decoder?
- Review: conditional language model assigns probability to a sequence of words $\mathbf{y} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_l)$ given condition $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{l} p(\mathbf{w}_t|\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_{t-1})$$

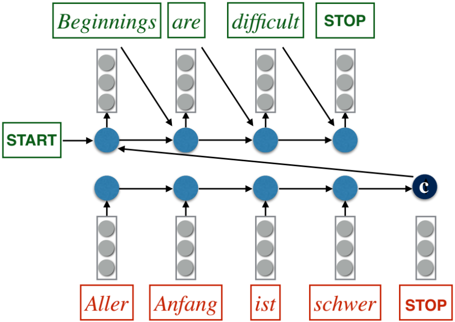
- $\mathbf{x}$: original sentence; $\mathbf{y}$: translated sentence

Figures in prev 4 slides from https://m2dsupsdlclass.github.io/lectures-labs; figures in next 2 slides from Dyer, Oxford NLP course Lecture 7, 2017

Machine translation: training

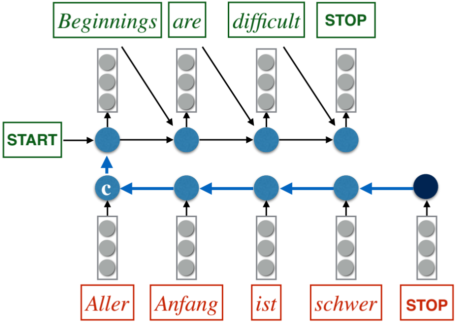
- Train encoder-decoder $p_{\boldsymbol{\theta}}(\cdot)$ by maximizing the log probability of correct translation $\mathbf{y}$ given the source sentence $\mathbf{x}$, i.e., maximizing the following objective function

$$\frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$$



Machine translation: training tricks

- Encoder reads source sequence 'backward': first read last word
- It improves both short and long sentence translations



- Multiple (e.g., 4) layers of LSTMs for both encoder & decoder
- An ensemble of independently trained encoder-decoders

Encoder output is meaningful

- After training, sentences with similar meanings are close to each other in the encoder's feature space

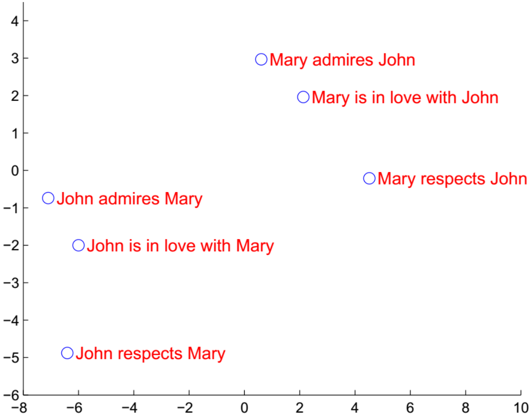


Figure from Sutskever, Vinyals, Le, "Sequence to sequence learning with neural networks", NIPS, 2014

Machine translation: inference

- Once training is finished, given a new source sentence $\mathbf{x}$, the model $p_{\boldsymbol{\theta}}(\cdot)$ can produce the translation

$$\begin{aligned}
\mathbf{y}^* &= \arg\max_{\mathbf{y}} p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \\
&= \arg\max_{\mathbf{y}} \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(\mathbf{w}_t|\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_{t-1})
\end{aligned}$$

Machine translation: inference

- Once training is finished, given a new source sentence $\mathbf{x}$, the model $p_{\boldsymbol{\theta}}(\cdot)$ can produce the translation

$$\begin{aligned} \mathbf{y}^* &= \arg\max_{\mathbf{y}} p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \\ &= \arg\max_{\mathbf{y}} \sum_{t=1}^{|\mathbf{y}|} \log p_{\boldsymbol{\theta}}(\mathbf{w}_t|\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_{t-1}) \end{aligned}$$

- How to find the best translation $\mathbf{y}^*$ efficiently? Greedy search

$$\begin{aligned} \mathbf{w}_1^* &= \arg\max_{\mathbf{w}_1} p_{\boldsymbol{\theta}}(\mathbf{w}_1|\mathbf{x}) \\ \mathbf{w}_2^* &= \arg\max_{\mathbf{w}_2} p_{\boldsymbol{\theta}}(\mathbf{w}_2|\mathbf{x}, \mathbf{w}_1^*) \\ &\vdots \\ \mathbf{w}_t^* &= \arg\max_{\mathbf{w}_t} p_{\boldsymbol{\theta}}(\mathbf{w}_t|\mathbf{x}, \mathbf{w}_1^*, \mathbf{w}_2^*, \ldots, \mathbf{w}_{t-1}^*) \end{aligned}$$

Beam search

- Beam search: keep track of top K hypotheses (translated partial sequences so far) at each time step!

Beam search

- Beam search: keep track of top K hypotheses (translated partial sequences so far) at each time step!
- Example: $K = 2$; first input to decoder is a starting token

$\boldsymbol{x} = \textit{Bier trinke ich}$

beer **drink** **I**

$\langle \mathrm{s} \rangle$
logprob=0

w_0 w_1 w_2 w_3

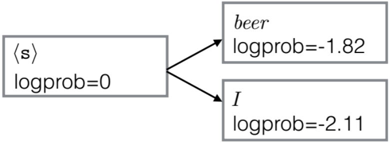
Figures here and in next 7 slide from Dyer, Oxford NLP course Lecture 7, 2017

Beam search (cont')

- 1^{st} time step: keep $K = 2$ most likely words which have higher log probability

$\boldsymbol{x} = \textit{Bier trinke ich}$

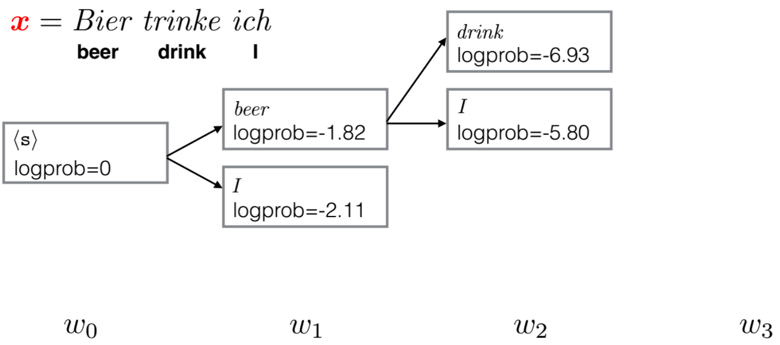
beer **drink** **I**



w_0 w_1 w_2 w_3

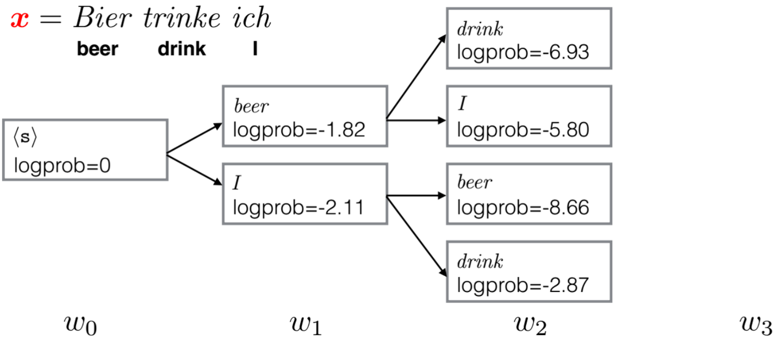
Beam search (cont')

- 2^{nd} time step: for each kept word at 1^{st} time step, proceed to produce $K = 2$ most likely words



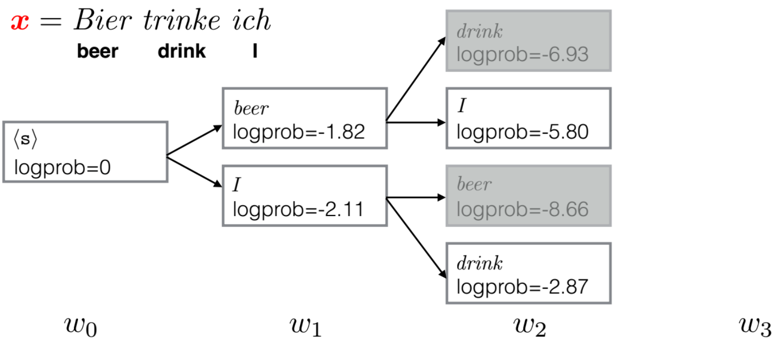
Beam search (cont')

- 2^{nd} time step: for each kept word at 1^{st}, proceed to produce $K = 2$ most likely words



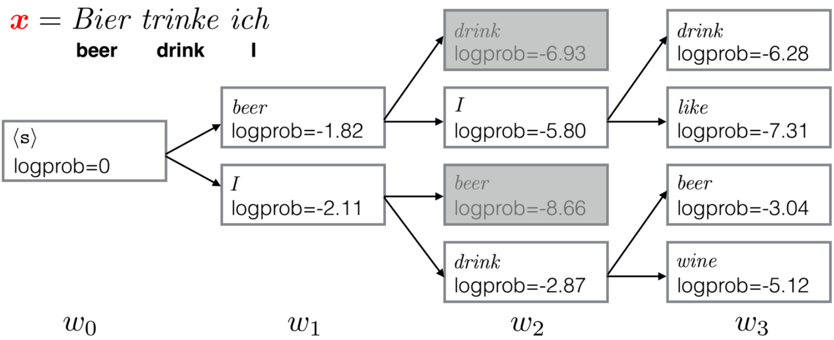
Beam search (cont')

- 2^{nd} time step: only keep $K = 2$ words with higher log prob
- Note: log probability is for each partial sequence of words



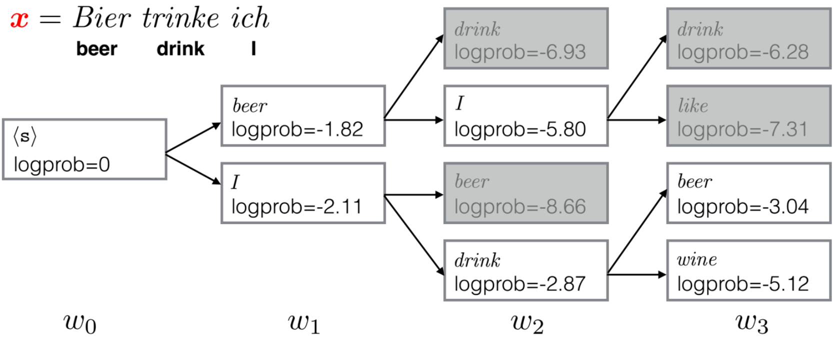
Beam search (cont')

- 3^{rd} time step: for each kept word at previous step, repeat the process as above



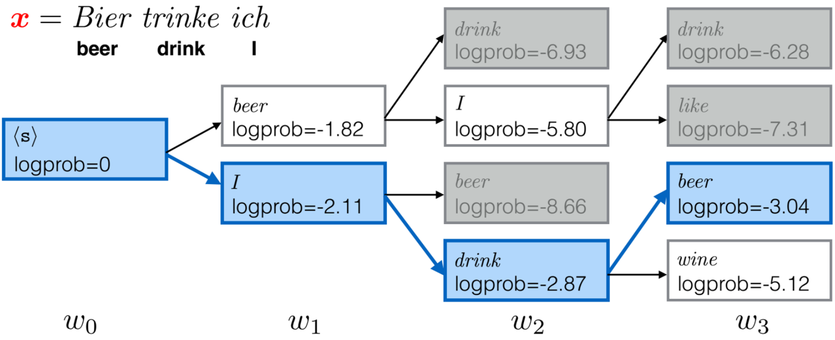
Beam search (cont')

- 3^{rd} time step: again only keep $K = 2$ words with higher log probability



Beam search (cont')

- Once producing 'end of sentence' token, select the best sequence with higher log probability (from K sequences)



Issue of encoder-decoder model

Problem:

- Whole source sentence is represented as a fixed-length vector
- This makes the network difficult to cope with long sentences
- Also, a sentence may have different parts with different concepts. e.g., 'I like apples but I don't like orange'

Issue of encoder-decoder model

Problem:

- Whole source sentence is represented as a fixed-length vector
- This makes the network difficult to cope with long sentences
- Also, a sentence may have different parts with different concepts. e.g., ‘I like apples but I don’t like orange’

Solution:

- Use outputs of encoder at all time steps.
- Build an attention mechanism to determine which outputs of the encoder to attend to during translation.

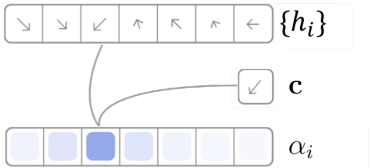
Attention mechanism

- Goal: select most relevant vector(s) given context $\mathbf{c}$
- h_i contains information with a strong focus on the parts surrounding the i^{th} word of the input sequence
- $\mathbf{c}$ may be decoder's hidden output at one time step

$\{h_i\}$ vectors to attend to

$\mathbf{c}$ context

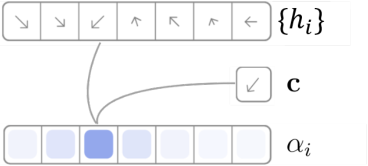
Figures here and in the next 7 slides from https://m2dsupsdlclass.github.io/lectures-labs



$$\boxed{e_i = f(h_i, \mathbf{c})}$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

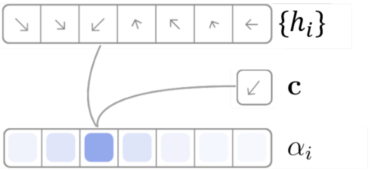
- $f(\cdot)$ may be a consine similarity, a deep network, etc.
- softmax enables to normalize and focus on very few items



$$e_i = f(h_i, \mathbf{c})$$

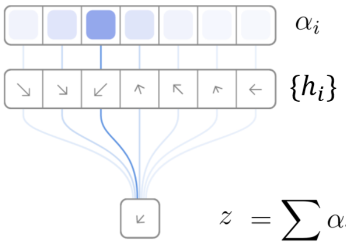
$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

- $f(\cdot)$ may be a consine similarity, a deep network, etc.
- softmax enables to normalize and focus on very few items
- α_i represents degree of 'attention' to region around the i^{th} location in the input sequence, or the importance of the region in predicting the next word during translation.



$$\boxed{e_i = f(h_i, \mathbf{c})}$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

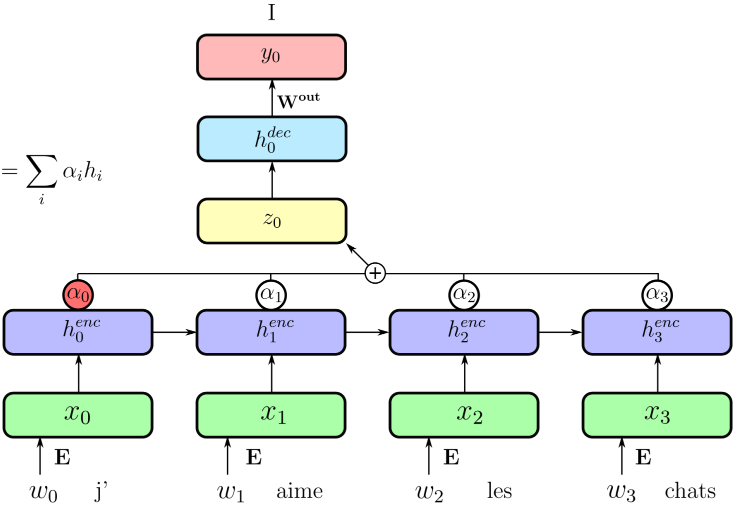


$$z = \sum_i \alpha_i h_i$$

- z: a soft (differentiable) selection on a set of words in the input sequence
- z helps predict next word during translation

- Predict 1^{st} word by decoder: z_0 as input to decoder is the soft selection of source words' representations

$$z_0 = \sum_i \alpha_i h_i$$

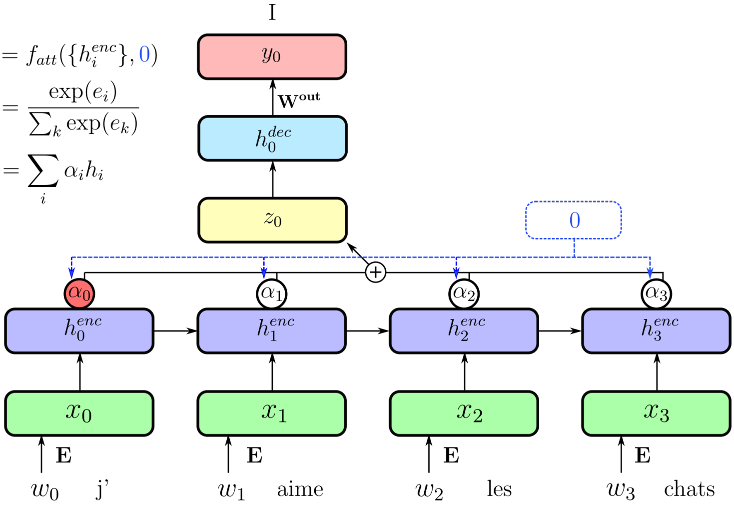


- ‘Attention’ is computed by matching a default context ‘0’ with the hidden state at each time step in encoder

$$e_i = f_{att}(\{h_i^{enc}\}, 0)$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

$$z_0 = \sum_i \alpha_i h_i$$

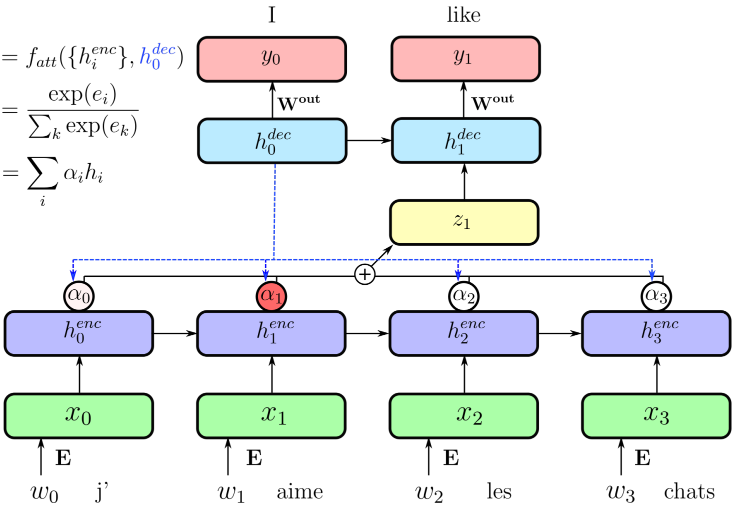


- Predict 2^{nd} word: compute attention to each source word with 'context' being h_0^{dec}

$$e_i = f_{att}(\{h_i^{enc}\}, h_0^{dec})$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

$$z_1 = \sum_i \alpha_i h_i$$

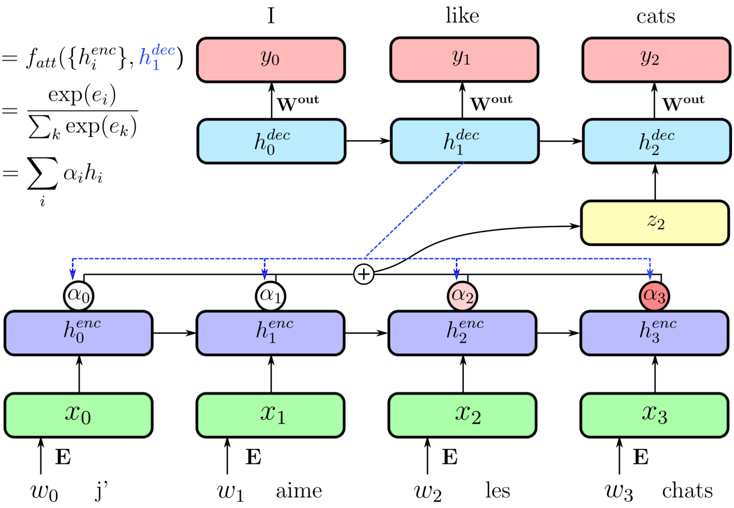


- Predict 3^{rd} word: compute attention to each source word with 'context' being h_1^{dec}

$$e_i = f_{att}(\{h_i^{enc}\}, h_1^{dec})$$

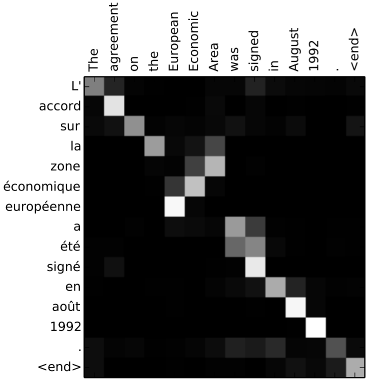
$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

$$z_2 = \sum_i \alpha_i h_i$$



English-French translation result

- x-axis: source sentence (English); y-axis: target sentence
- Each pixel: $\alpha_{i,j}$, weight of j^{th} source word for i^{th} target word



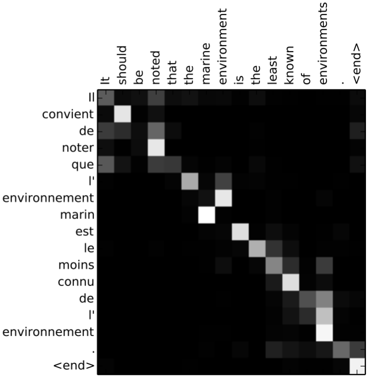
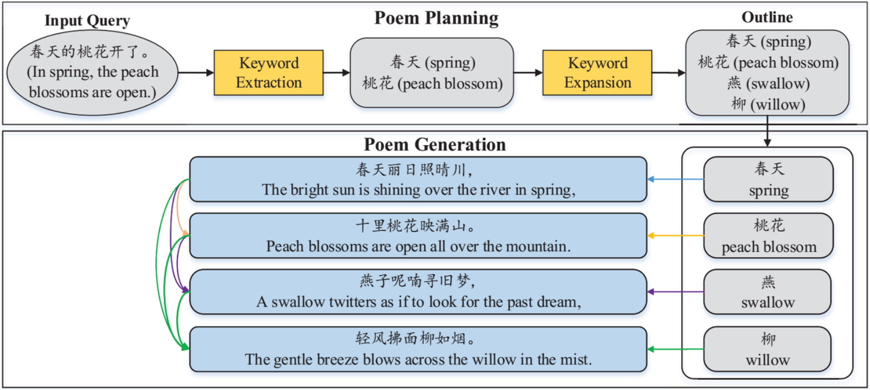


Figure from Bahdanau, Cho, Bengio, "Neural machine translation by jointly learning to align and translate", ICLR, 2015

Poetry generation: another application of attention

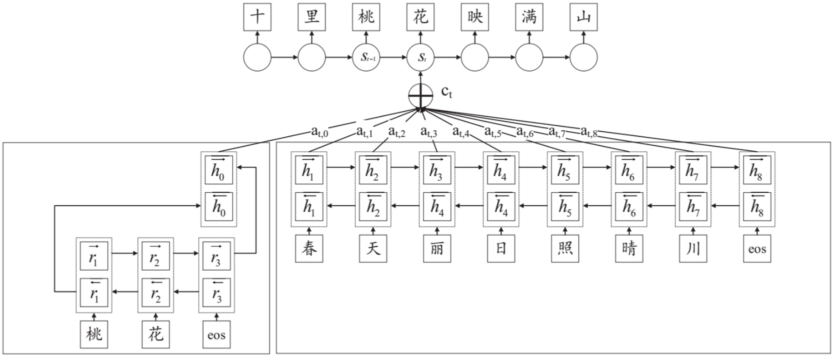
- Goal: generate poem, given a query sentence/words
- Two steps: first key words, then lines for each key words



Figures here and in next 4 slides from Wang et al., "Chinese poetry generation with planning based neural network", arXiv, 2016

Poetry generation

- Bidirectional GRU for encoder and decoder
- For each line: both key word and previous lines are encoded by encoder for attention computation



Poetry generation: training

- Train model by maximizing log-likelihood of training corpus

$$\arg\max \sum_{n=1}^{N} logP(\mathbf{y_n}|\mathbf{x_n}, \mathbf{k_n})$$

Keyword	The Preceding Text	Current Line
床	–	床前明月光
霜	床前明月光	疑是地上霜
明月	床前明月光; 疑是地上霜	举头望明月
故乡	床前明月光; 疑是地上霜; 举头望明月	低头思故乡

- Element of 1^{st} column: $\mathbf{k}_n$; 2^{nd} column: $\mathbf{x}_n$; 3^{rd} column: $\mathbf{y}_n$

Poetry generation: result

- Which poem is generated by model?

秋夕湖上	秋夕湖上
By a Lake at Autumn Sunset	By a Lake at Autumn Sunset
一夜秋凉雨湿衣，	荻花风里桂花浮，
A cold autumn rain wetted my clothes last night,	The wind blows reeds with osmanthus flying,
西窗独坐对夕晖。	恨竹生云翠欲流。
And I sit alone by the window and enjoy the sunset.	And the bamboos under clouds are so green as if to flow down.
湖波荡漾千山色，	谁拂半湖新镜面，
With mountain scenery mirrored on the rippling lake,	The misty rain ripples the smooth surface of lake,
山鸟徘徊万籁微。	飞来烟雨暮天愁。
A silence prevails over all except the hovering birds.	And I feel blue at sunset .

Poetry generation: result

- Which poem is generated by model?
- Enjoy poems with modern title

秋夕湖上 By a Lake at Autumn Sunset 一夜秋凉雨湿衣， A cold autumn rain wetted my clothes last night, 西窗独坐对夕晖。 And I sit alone by the window and enjoy the sunset. 湖波荡漾千山色， With mountain scenery mirrored on the rippling lake, 山鸟徘徊万籁微。 A silence prevails over all except the hovering birds.	秋夕湖上 By a Lake at Autumn Sunset 荻花风里桂花浮， The wind blows reeds with osmanthus flying, 恨竹生云翠欲流。 And the bamboos under clouds are so green as if to flow down. 谁拂半湖新镜面， The misty rain ripples the smooth surface of lake, 飞来烟雨暮天愁。 And I feel blue at sunset .

啤酒 Beer 今宵啤酒两三缸， I drink glasses of beer tonight, 杯底香醇琥珀光。 With the bottom of the glass full of aroma and amber light. 清爽金风凉透骨， Feeling cold as the autumn wind blows, 醉看明月挂西窗。 I get drunk and enjoy the moon in sight by the west window.	冰心 Xin Bing 一片冰心向月明， I open up my pure heart to the moon, 千山春水共潮生。 With the spring river flowing past mountains. 繁星闪烁天涯路， Although my future is illuminated by stars, 往事萦怀梦里行。 The past still lingers in my dream.

Dialogue: another application of RNNs

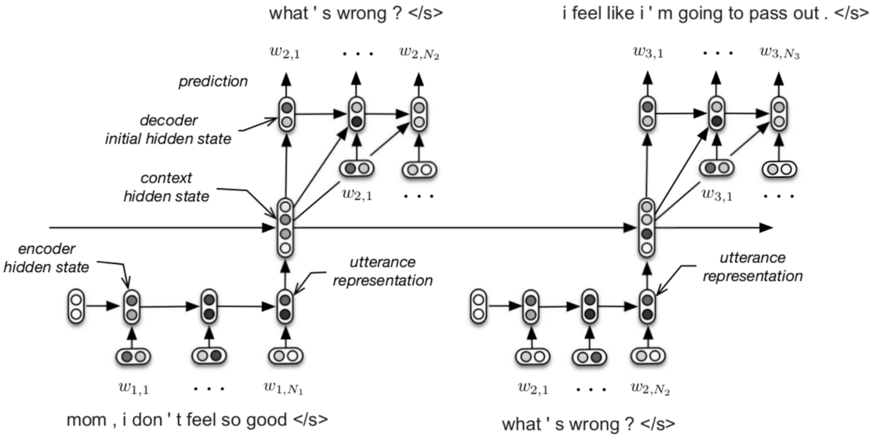
Machine translation vs. Dialogue (chatbot)
Which is more difficult?

Dialogue model: HRED

- Dialogue: a sequence of utterances (sentences)

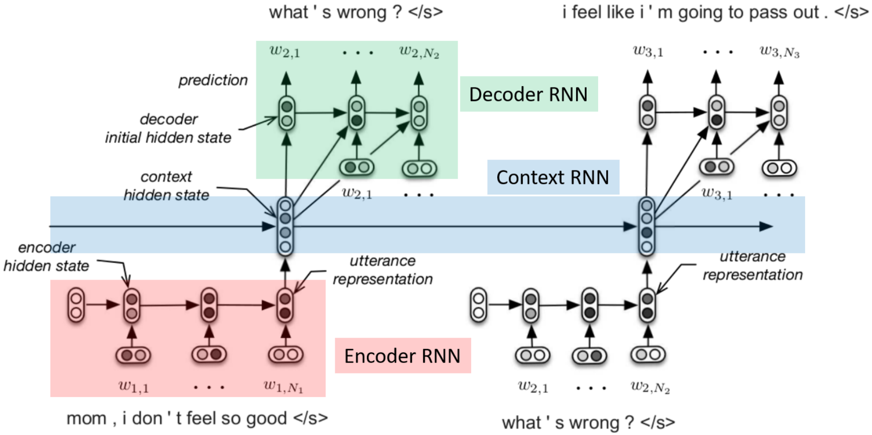
Dialogue model: HRED

- Dialogue: a sequence of utterances (sentences)
- Hierarchical recurrent encoder-decoder: 3 RNNs, 2 levels



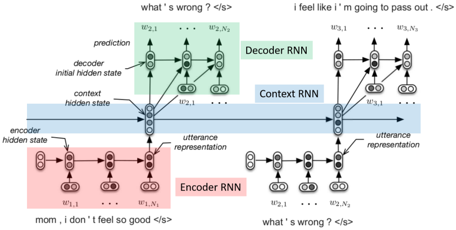
HRED model (cont')

- Context RNN (blue) encode temporal information among multiple sentences (utterances); easier for gradient flow



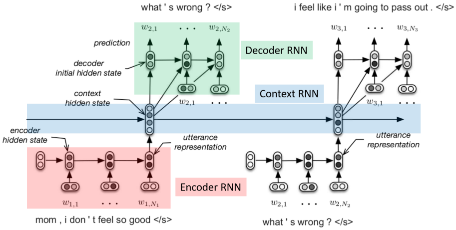
HRED model (cont')

- Context RNN can represents common ground between speakers, e.g., topics or concepts shared between speakers
- Context encoder output as input for each step of decoder



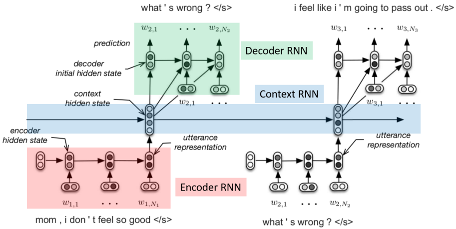
HRED model (cont')

- Context RNN can represents common ground between speakers, e.g., topics or concepts shared between speakers
- Context encoder output as input for each step of decoder
- Other trick 1: train word embedding model with other data
- Other trick 2: pretrain RNN with non-dialogue corpus



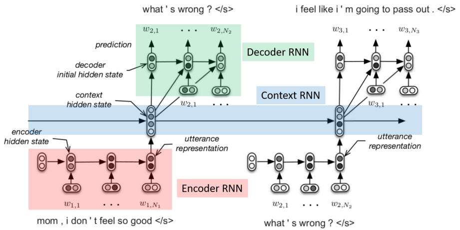
Issue of HRED

- However, most predictions are too generic, like 'I don't know' or 'I am sorry'



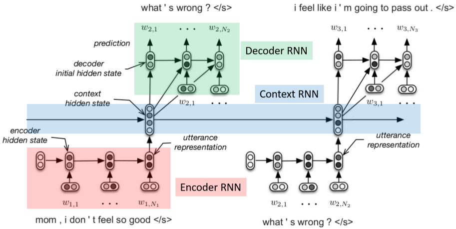
Issue of HRED

- However, most predictions are too generic, like 'I don't know' or 'I am sorry'
- Reason 1: generic utterances appear often in training set



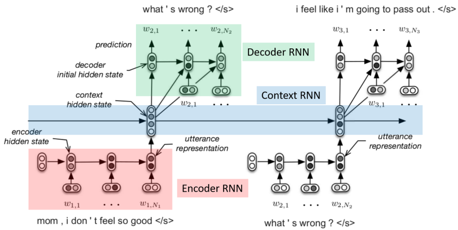
Issue of HRED

- However, most predictions are too generic, like ‘I don’t know’ or ‘I am sorry’
- Reason 1: generic utterances appear often in training set
- Reason 2: many words are punctuation marks or pronouns, making context RNN difficult to learn topics/concepts



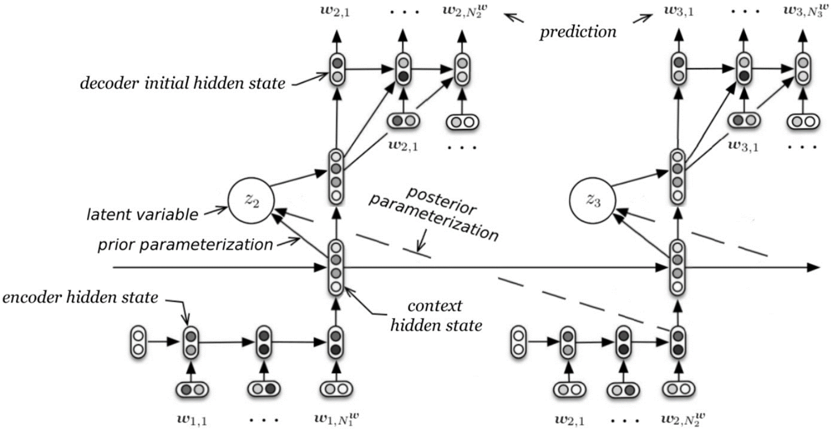
Issue of HRED

- However, most predictions are too generic, like ‘I don’t know’ or ‘I am sorry’
- Reason 1: generic utterances appear often in training set
- Reason 2: many words are punctuation marks or pronouns, making context RNN difficult to learn topics/concepts
- Reason 3: Injections to context RNN is from encoder outputs which largely encode local structure of a sentence, making context RNN difficult to capture structures of whole sentences



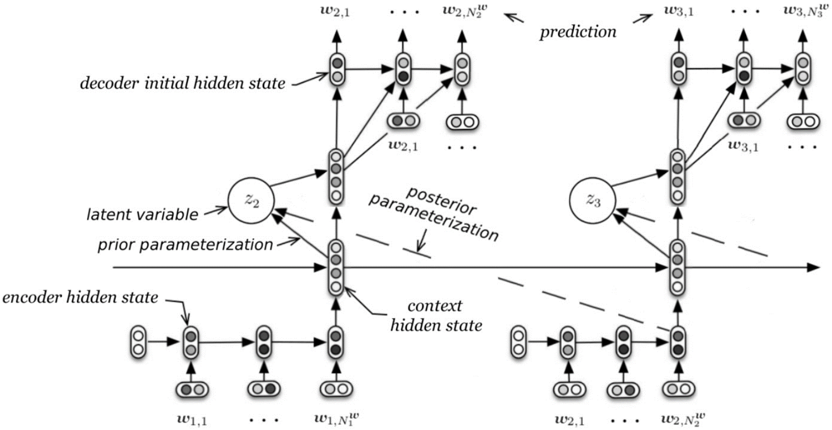
VHRED: Variational HRED

- Introduce a latent variable $\mathbf{z}$ whose distribution is Gaussian



VHRED: Variational HRED

- Introduce a latent variable $\mathbf{z}$ whose distribution is Gaussian
- Concatenate $\mathbf{z}$ and output of context RNN for decoder



VHRED (cont')

- Mean and variance of the Gaussian are functions of all previous utterances.

$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}))$$

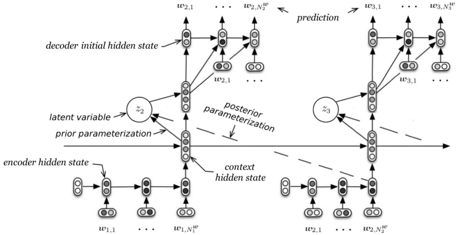


Figure here and table in next slide from Serban et al., "A hierarchical latent variable encoder-decoder model generating dialogues", arXiv, 2016

VHRED (cont')

- Mean and variance of the Gaussian are functions of all previous utterances.
$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}))$$
- The functions are simple feed-forward networks trained end-to-end as part of the whole network model.
- $\mathbf{z}$ can capture/represent high-level information like topic or sentiment of sentence.

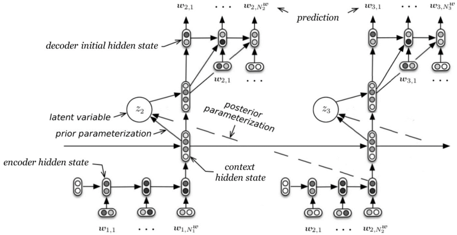


Figure here and table in next slide from Serban et al., "A hierarchical latent variable encoder-decoder model generating dialogues", arXiv, 2016

VHRED result

- VHRED generates longer and more meaningful responses
- Dataset: Twitter Dialogue Corpus; Ubuntu Dialogue Corpus

Context	**Response**
jamaica blood!! <url> → lol. yes [name]!. [name]! → what's poppin in toronto homie	**VHRED:** nothing much. just chillin. what's good with you? **LSTM:** something!!!!!!! **HRED:** something!!!!!!
whhhhhhyyyy is it suddenly cold in tallahassee today?? this is florida! how can i dream of the beach when my fingers are icey cold?? → sittin on deck in sun looking at lake travis in austin! enjoying day b4 birthday	**VHRED:** the sun is shining!!! **LSTM:** i know!!!!!!!!! **HRED:** something!!!!!!!
reaches for [name]! → - approaches him warily - hey there buddy. - swings him into the air - how you doin? → tilts his head and your wariness and reaches for your face [name].	**VHRED:** - looks up at you and smiles - you're so cute. - kisses your cheek - **LSTM:** - smiles - i'm fine thank you. how are you? **HRED:** - grins, looking at you as i look at you -
i think i want an ipad2. but i already have an ibook. want vs. need. → if you have an ibook. then you deserve an ipad!	**VHRED:** that's what i'm thinking. i'm going to buy a new one. **LSTM:** i want one!!!! **HRED:** i don't know what to do with it. i don't know what to do.
ah echt? dan anrtwoord ik volgende keer via twitter als je me smst. → haha doe maar p. goed bezig hoor myrthe d → goed bezig? die snap ik even niet p	**VHRED:** haha, ik heb er geen zin in de something, maar ik heb er geen something meer something **LSTM:** haha something something something **HRED:** something something something something something something something something
i love katie waissel more than like, anything in life. → i wanna c the show again! i need <number> meet her again! n i want a b-day tweet nxt week doubt that will happen x	**VHRED:** me too. i'm going to have to wait for her to come back. i'm so excited. **LSTM:** i'm sure she will. i'm sure she'll be fine. **HRED:** i'm sure she'll be fine, i'm sure she'll be fine.
my math teacher over here talkn bout nickels nd i'm tryna make big money i need to be in calculus → lol	**VHRED:** i need to go to the gym and get my hair done lol **LSTM:** lol **HRED:** how u been

Issue of evaluation

- Which response is better? Response quality score 1-5

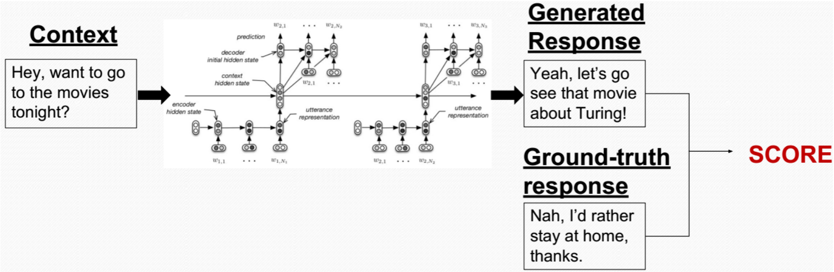


Figure here and in the next 3 slides from Serban et al., "Building end-to-end dialogue systems using generative hierarchical neural network models", AAAI, 2016

Issue of evaluation

- Which response is better? Response quality score 1-5
- Current metrics may give low score to the generated response

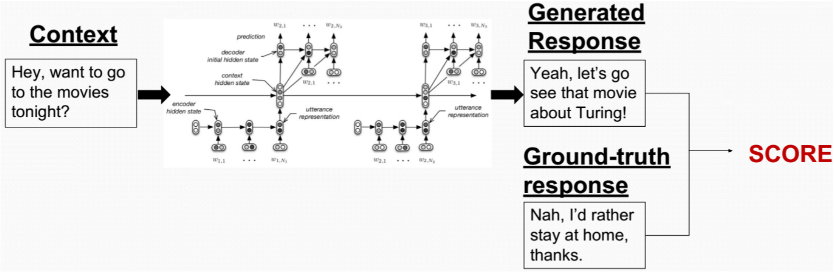


Figure here and in the next 3 slides from Serban et al., "Building end-to-end dialogue systems using generative hierarchical neural network models", AAAI, 2016

Issue of evaluation (cont')

- Humans rank generated responses consistently, i.e., give low score to poor responses and high score to good ones!

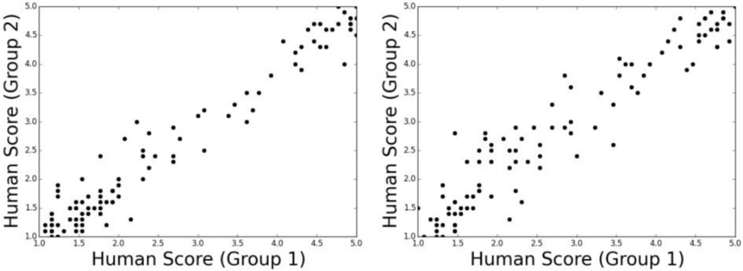
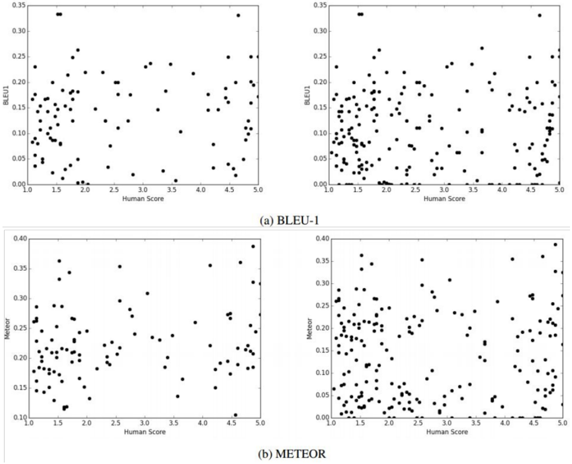


Figure 3: Scatter plots showing the correlation between two randomly chosen groups of human volunteers on the Twitter corpus (left) and Ubuntu Dialogue Corpus (right).

Issue of evaluation (cont')

- Scores from existing metrics (y-axis; e.g., BLEU, METEOR) are not well correlated with human scores (x-axis).



(a) BLEU-1

(b) METEOR

Summary

- Encoder-decoder model is popular for machine translation
- Attention mechanism can well handle longer sentences
- Poem generation by by encoder-decoder with attention
- Chatbot is on the way, difficult to evaluate

Further reading:

- Amodei et al., 'Deep Speech 2: End-to-End Speech Recognition in English and Mandarin', arXiv, 2015