E0-270 Project on Neural Dialog Generation

Siddharth Jha, Vinayak, Divij Mishra

Why is dialogue generation tough?

- The aim is to generate more and more human-like responses to a given utterance.

- This involves-
  - Fact matching
  - Ability to correlate information
  - Long term memory
  - Presence of knowledge base

Papers read

1. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion
   (Jul 2015) (Sordoni, Bengio, et. al.)
2. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Networks
   (Apr 2016) (Serban, Sordoni, et. al.)
3. Adversarial Learning for Neural Dialogue Generation
   (Sep 2017) (Li, Monroe, et. al.)
4. End-to-End Adversarial Learning for Generative Conversational Agents
   (Nov 2017) (Ludwig)
5. Knowledge Diffusion for Neural Dialogue Generation
   (Jul 2018) (Liu, Chen, et. al.)

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Ideas:

1. Generative probabilistic model for query prediction and auto-completion
2. Use of HRED (Hierarchical Recurrent Encoder-Decoder) architecture to encode context
3. General model, applicable outside of query-prediction setting - in particular, applicable to dialogue generation.

Proposed advantages:

1. Goes beyond query co-occurrence to avoid *data sparsity* and predict *long-tail queries*
2. Increased context-awareness
3. Synthetic suggestions

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Key idea:

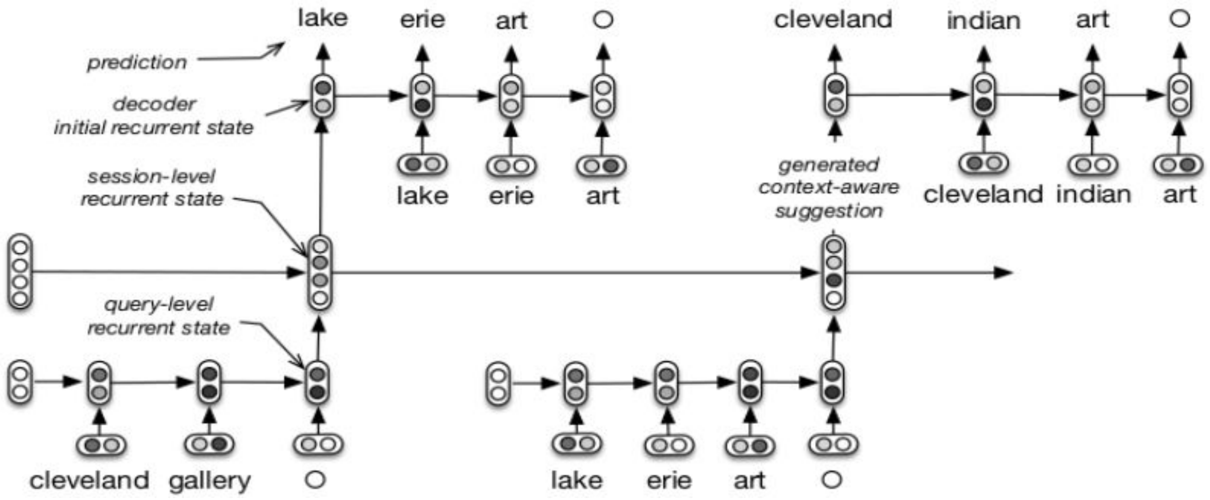
1. Obtain vector embedding of words.
2. Obtain vector embedding of a query by sequentially averaging across the words. This is done by the *query-level RNN.*
3. Additional *session-level RNN*, which iterates over all the query vectors to output a single vector embedding of the query session till the present.
4. Use an RNN as a decoder to translate this session vector into a response vector.

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Architecture:

HRED:



A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Architecture:

Query-level encoder:

1. The RNN sequentially reads a query, updating its hidden state after each word.
2. It uses the GRU function to generate the next hidden state.
3. Outputs the query vector.

$$h_{m,n} = GRU_{enc}(h_{m,n-1}, w_{m,n}), \quad n = 1, \ldots, N_m,$$

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Side note on GRU:

Gated Recurrent Unit: It's a set of operations that generates a hidden state using present word embedding and previous hidden state.

$$
\begin{aligned}
r_n &= \sigma(I_r w_n + H_r h_{n-1}), \\
u_n &= \sigma(I_u w_n + H_u h_{n-1}), \\
\bar{h}_n &= \tanh(I w_n + H(r_n \cdot h_{n-1})), \\
h_n &= (1 - u_n) \cdot h_{n-1} + u_n \cdot \bar{h},
\end{aligned}
$$

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Architecture:

Session-level encoder:

1. Acts like the query-level encoder.
2. Uses an RNN (again, with GRU) to sequentially read the query vectors and output the session vector.

$$s_m = GRU_{ses}(s_{m-1}, q_m), \quad m = 1, \ldots, M,$$

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Architecture:

Query decoder:

1. Uses the session state to generate the next query.
2. Uses a non-linear function to project the session-state into the decoder-space.

$$d_{m,0} = \tanh(D_0 s_{m-1} + b_0)$$

3. Using this state as the initial hidden state of the new query, it uses an RNN (with GRU) to obtain the next hidden state *given the next word*.

$$d_{m,n} = GRU_{dec}(d_{m,n-1}, w_{m,n}), \quad n = 1, \ldots, N_m$$

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Architecture:

Query-decoder continued:

1. The probability for a word to be the next word is given by the softmax function:

$$P(w_{m,n} = v \mid w_{m,1:n-1}, Q_{1:m-1}) = $$

$$= \frac{\exp o_v^\top \omega(d_{m,n-1}, w_{m,n-1})}{\sum_k \exp o_k^\top \omega(d_{m,n-1}, w_{m,n-1})}$$

$o_v \in \mathbb{R}^{d_e}$ is the output embedding of word v

$$\omega(d_{m,n-1}, w_{m,n-1}) = H_o\, d_{m,n-1} + E_o\, w_{m,n-1} + b_o$$

2. The word with maximum value for the function is chosen.

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Learning:

1. Learns the parameters for the 3 GRUs and the parameters involved in finding the probability of a word
2. Done by maximising the log likelihood of a session of queries
3. The objective function for maximisation -

$$\begin{aligned}\mathcal{L}(S) &= \sum_{m=1}^{M} \log P(Q_m | Q_{1:m-1}) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N_m} \log P(w_{m,n} | w_{m,1:n-1}, Q_{1:m-1}).\end{aligned}$$

A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion

(July 2015) (Sordoni, Bengio et. al.)

Evaluation:

a) Sample size->18,000 sessions (taken from AOL)

b) Test data-> 9,500 sessions

c) Training data->7,500 sessions (Wrong values?)

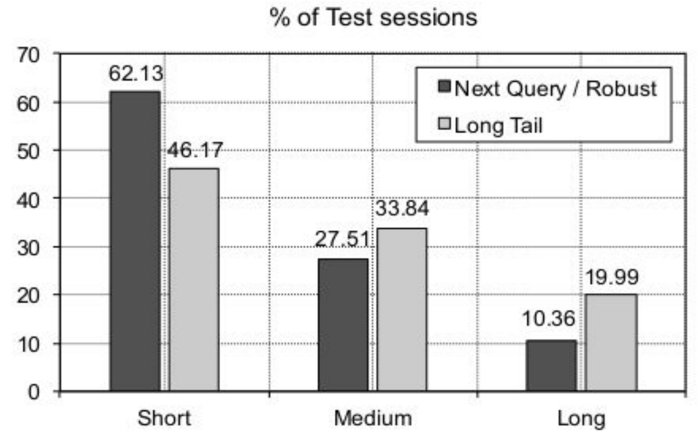


Figure 4: Proportion (%) of short (2 queries), medium (3 or 4 queries) and long (at least 5 queries) sessions in our test scenarios.

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

(Apr 2016) (Serban, Sordoni, et. al.)

Ideas:

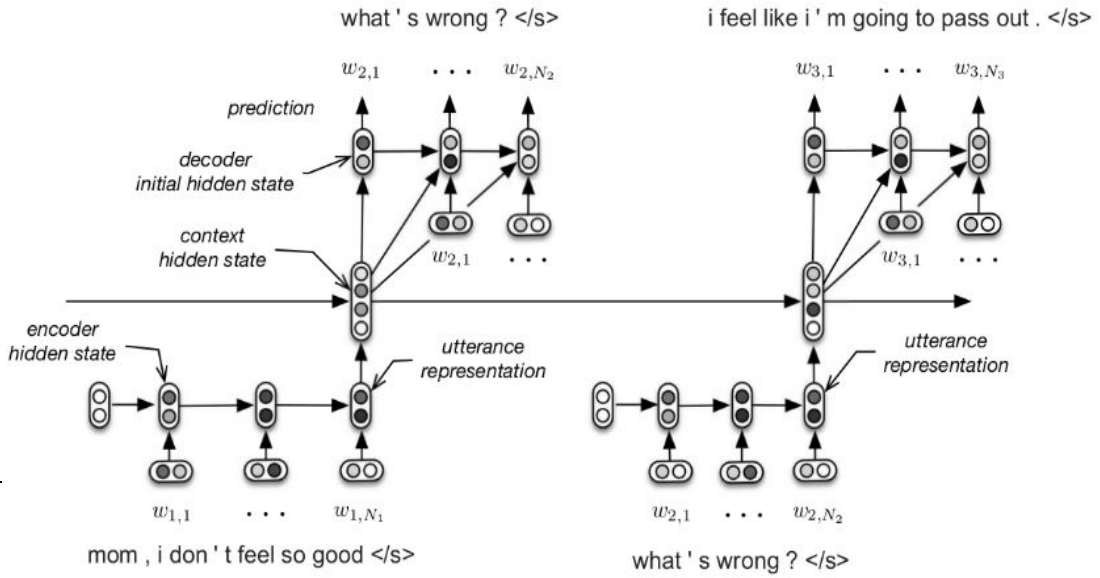
1. Generative probabilistic model for dialogue generation
2. Uses the previously described HRED architecture
3. Captures more information from a dialogue with a bidirectional encoder
4. Provides an end-to-end model

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

(Apr 2016) (Serban, Sordoni, et. al.)

Architecture:

1. Basic architecture similar
2. Encode an utterance twice,
   in both directions, take average


what ' s wrong ? </s>
i feel like i ' m going to pass out . </s>
$w_{2,1}$ . . . w_{2,N_2}
$w_{3,1}$ . . . w_{3,N_3}
prediction
decoder initial hidden state
context hidden state
$w_{2,1}$ . . .
$w_{3,1}$ . . .
encoder hidden state
utterance representation
utterance representation
$w_{1,1}$ . . . w_{1,N_1}
$w_{2,1}$ . . . w_{2,N_2}
mom , i don ' t feel so good </s>
what ' s wrong ? </s>

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

(Apr 2016) (Serban, Sordoni, et. al.)

Pre-training:

1. To find a good set of initial hyper-parameters, the HRED was pre-trained on the Q-A SubTle dataset, constructed from movie subtitles.
2. Word embeddings initialized from Word2Vec, trained on the Google News dataset.
3. To find the word embedding parameters, the word embedding is decomposed into a smaller embedding matrix. Learning is performed on this matrix.

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

(Apr 2016) (Serban, Sordoni, et. al.)

Evaluation metrics:

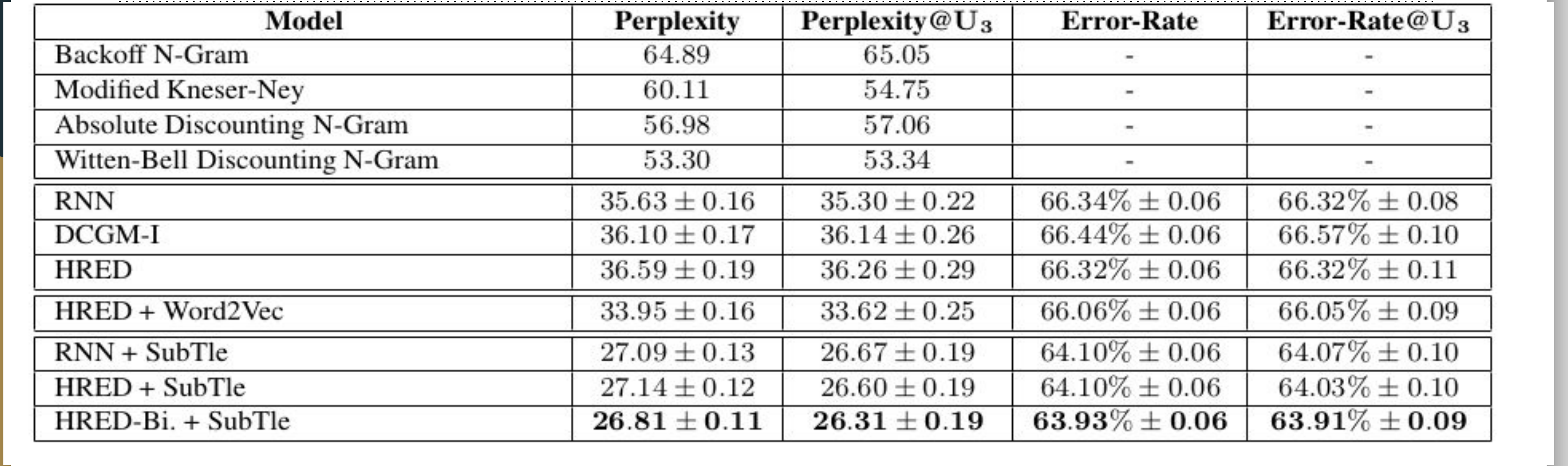
1. Perplexity: Measures the ability of the model to form connection between multiple sets of words

$$\exp\left(-\frac{1}{N_W}\sum_{n=1}^{N}\log P_\theta(U_1^n, U_2^n, U_3^n)\right)$$

2. Word classification error: Measures the ability of the model to generate frequently used words, defined as the number of words in the dataset this model has predicted incorrectly divided by the total number of words in the dataset.

Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

(Apr 2016) (Serban, Sordoni, et. al.)



Model	Perplexity	Perplexity@U_3	Error-Rate	Error-Rate@U_3
Backoff N-Gram	64.89	65.05	-	-
Modified Kneser-Ney	60.11	54.75	-	-
Absolute Discounting N-Gram	56.98	57.06	-	-
Witten-Bell Discounting N-Gram	53.30	53.34	-	-
RNN	35.63 ± 0.16	35.30 ± 0.22	$66.34\% \pm 0.06$	$66.32\% \pm 0.08$
DCGM-I	36.10 ± 0.17	36.14 ± 0.26	$66.44\% \pm 0.06$	$66.57\% \pm 0.10$
HRED	36.59 ± 0.19	36.26 ± 0.29	$66.32\% \pm 0.06$	$66.32\% \pm 0.11$
HRED + Word2Vec	33.95 ± 0.16	33.62 ± 0.25	$66.06\% \pm 0.06$	$66.05\% \pm 0.09$
RNN + SubTle	27.09 ± 0.13	26.67 ± 0.19	$64.10\% \pm 0.06$	$64.07\% \pm 0.10$
HRED + SubTle	27.14 ± 0.12	26.60 ± 0.19	$64.10\% \pm 0.06$	$64.03\% \pm 0.10$
HRED-Bi. + SubTle	$\mathbf{26.81 \pm 0.11}$	$\mathbf{26.31 \pm 0.19}$	$\mathbf{63.93\% \pm 0.06}$	$\mathbf{63.91\% \pm 0.09}$

Adversarial Learning for Neural Dialogue Generation

(Sep 2017) (Li, Monroe, et. al.)

Ideas:

1. Draw intuition from the Turing test, by using a generative model, and a discriminative model which tries to distinguish between machine-generated and human-generated dialogues.
2. Uses the discriminator output as reward for the generative model.

Proposed advantages:

1. Unlike manually defined reward systems, this reward system will cover all crucial aspects of dialogue generation, like coherence and informativeness.

Adversarial Learning for Neural Dialogue Generation

(Sep 2017) (Li, Monroe, et. al.)

Architecture:

Adversarial REINFORCE:

1. Generative model G:
   Similar to the decoder from the last paper, uses RNNs to sequentially generate an utterance given the dialogue history.
2. Discriminative model D:
   Takes a sequence of utterances as input and returns a label, indicating whether the input was generated by a human or machine. Uses an HRED to encode the input and a 2-class softmax function to calculate the probability of each class.

Adversarial Learning for Neural Dialogue Generation

(Sep 2017) (Li, Monroe, et. al.)

Architecture:

Adversarial REINFORCE continued:

1. Utterances classified as human-generated by D are used as rewards for G, according to the **REINFORCE algorithm (Williams,** $J(\theta) = \mathbb{E}_{y \sim p(y|x)}(Q_+(\{x,y\})|\theta)$
2. The gradient is approximated by:
$$\begin{aligned} \nabla J(\theta) &\approx [Q_+(\{x,y\}) - b(\{x,y\})] \\ &\quad \nabla \log \pi(y|x) \\ &= [Q_+(\{x,y\}) - b(\{x,y\})] \\ &\quad \nabla \sum_t \log p(y_t|x, y_{1:t-1}) \end{aligned}$$
3. D is simultaneously updated with a human-generated dialogue as a positive example, and the actual machine-generated dialogue as a negative example.

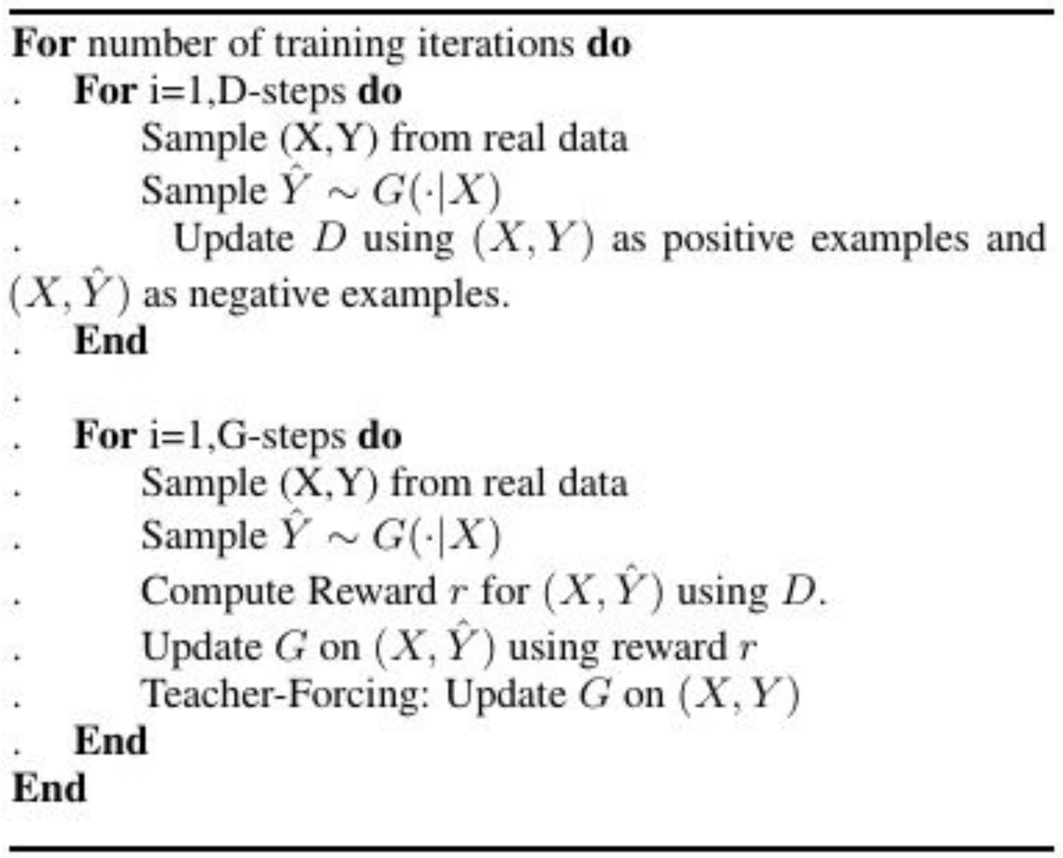
Adversarial Learning for Neural Dialogue Generation

(Sep 2017) (Li, Monroe, et. al.)

Architecture:

Reward for Every Generation Step (REGS):

1. A way to give more weight to certain tokens of an utterance - “I don’t know.”
   a. Monte-Carlo search
   b. Training discriminator to assign rewards to partially decoded sequences.
2. Found that MC performed better, although it required more training time.
3. Teacher forcing:
   1. The training system is fragile
   2. Propose to feed the discriminator with human-generated dialogues and force it to return 1, to stabilise the learning process.



```
**For** number of training iterations **do**
.    **For** i=1,D-steps **do**
.        Sample (X,Y) from real data
.        Sample $\hat{Y} \sim G(\cdot|X)$
.        Update $D$ using $(X, Y)$ as positive examples and $(X, \hat{Y})$ as negative examples.
.    **End**
.
.    **For** i=1,G-steps **do**
.        Sample (X,Y) from real data
.        Sample $\hat{Y} \sim G(\cdot|X)$
.        Compute Reward $r$ for $(X, \hat{Y})$ using $D$.
.        Update $G$ on $(X, \hat{Y})$ using reward $r$
.        Teacher-Forcing: Update $G$ on $(X, Y)$
.    **End**
**End**
```

End-to-End Adversarial Learning for Generative Conversational Agents

(Nov 2017) (Ludwig)

Key idea:

1. Encodes the last few utterances of the conversation into a ‘thought’ vector.
2. Concatenates the thought vector and the input embedding vector, and uses that while generating each token of the output sentence.
3. This ensures context preservation.

End-to-End Adversarial Learning for Generative Conversational Agents

(Nov 2017) (Ludwig)

Architecture:

1. Converts the one-hot representations of the words of the input and output utterances to vectors of the embedding layer.

2. Uses 2 separate LSTMs to process these vectors, since one is related to context and the other to the incomplete answer.

$$e_c = \Gamma_c (E_c; \mathcal{W}_c) \qquad E_c = W_e X$$
$$e_a = \Gamma_a (E_a; \mathcal{W}_a) \qquad E_a = W_e Y$$

End-to-End Adversarial Learning for Generative Conversational Agents

(Nov 2017) (Ludwig)

Architecture:

1. The vectors are then concatenated and processed by a ReLU activation and a softmax function to output the probabilities for the next token.

$$
\begin{aligned}
e &= [e_c \quad e_a] \\
y_h &= \sigma(W_1\, e + b_1) \\
\mathbf{p} &= \varphi(W_2\, y_h + b_2)
\end{aligned}
$$

2. As earlier, the token with the maximum probability is chosen.

A key advantage of this model is that is prevents *short answers with high prior probability*.

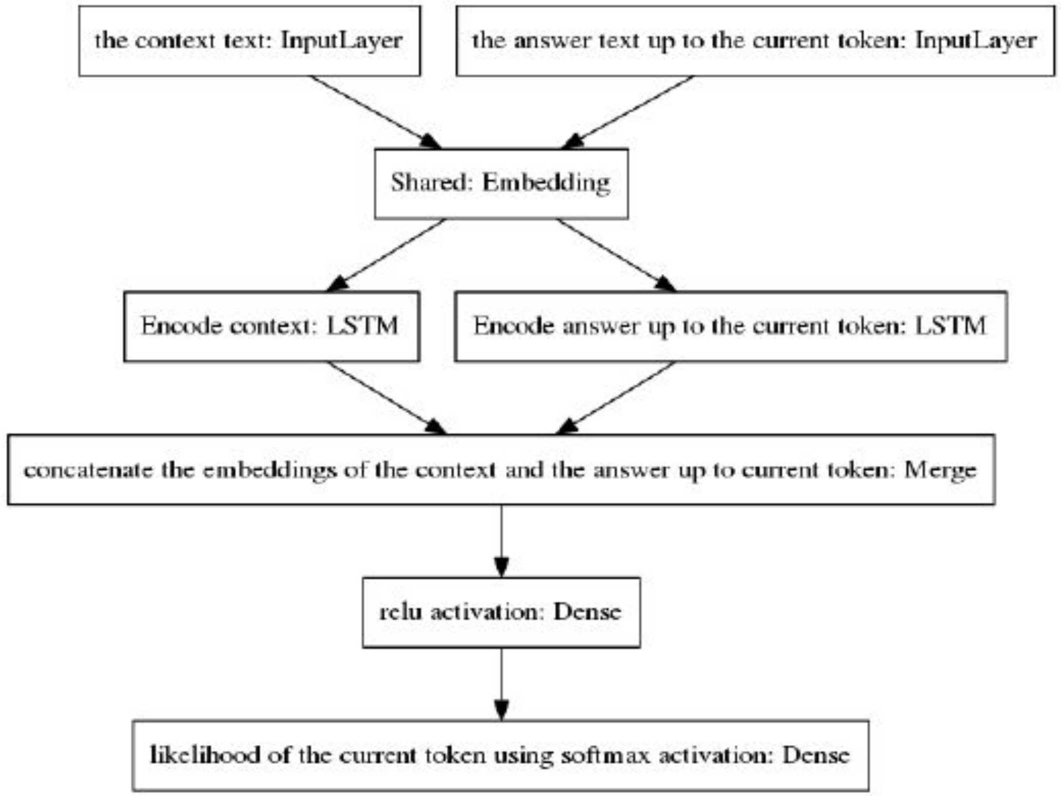
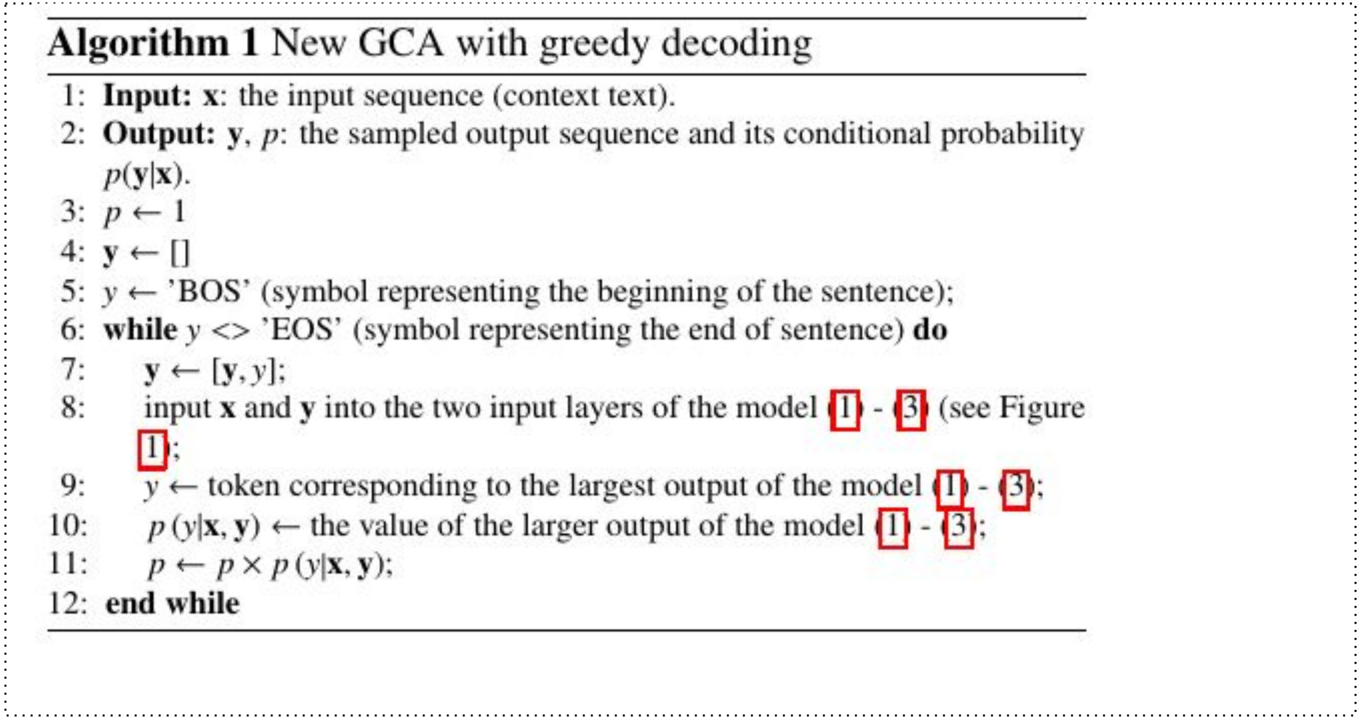

the context text: InputLayer
the answer text up to the current token: InputLayer
Shared: Embedding
Encode context: LSTM
Encode answer up to the current token: LSTM
concatenate the embeddings of the context and the answer up to current token: Merge
relu activation: Dense
likelihood of the current token using softmax activation: Dense

Figure 1: The model of the new GCA.



Algorithm 1 New GCA with greedy decoding

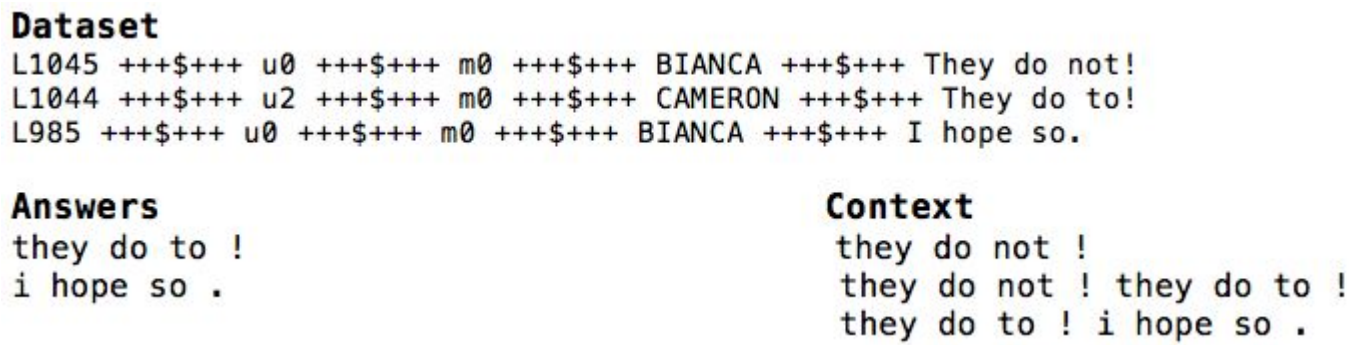
```
1: Input: $\mathbf{x}$: the input sequence (context text).
2: Output: $\mathbf{y}$, $p$: the sampled output sequence and its conditional probability $p(\mathbf{y}|\mathbf{x})$.
3: $p \leftarrow 1$
4: $\mathbf{y} \leftarrow []$
5: $y \leftarrow$ 'BOS' (symbol representing the beginning of the sentence);
6: while $y <>$ 'EOS' (symbol representing the end of sentence) do
7:     $\mathbf{y} \leftarrow [\mathbf{y}, y]$;
8:     input $\mathbf{x}$ and $\mathbf{y}$ into the two input layers of the model (1) - (3) (see Figure 1);
9:     $y \leftarrow$ token corresponding to the largest output of the model (1) - (3);
10:    $p(y|\mathbf{x}, \mathbf{y}) \leftarrow$ the value of the larger output of the model (1) - (3);
11:    $p \leftarrow p \times p(y|\mathbf{x}, \mathbf{y})$;
12: end while
```

Key contributions-

1. The code was adopted from its python2 version and appropriately implemented in its python 3 version.
2. A new data set was chosen to perform testing, training and experiments.
3. The data set processed to be in QA format.
4. Training was done by bootstrapping on authors weights and by training the network from scratch.
5. The architecture was modified by using GRU instead of LSTM.
6. The effect of vocabulary on data was explored.

Key Results

1. Better results when bootstrapped weights were used
2. Use of data independent vocabulary (as proposed by author) improves performance.
3. Better performance on non factoid sentence.
4. Similar performance of GRU and LSTM architectures.
5. The sentences containing words from the traininig data generated more meaningful responses.



Dataset

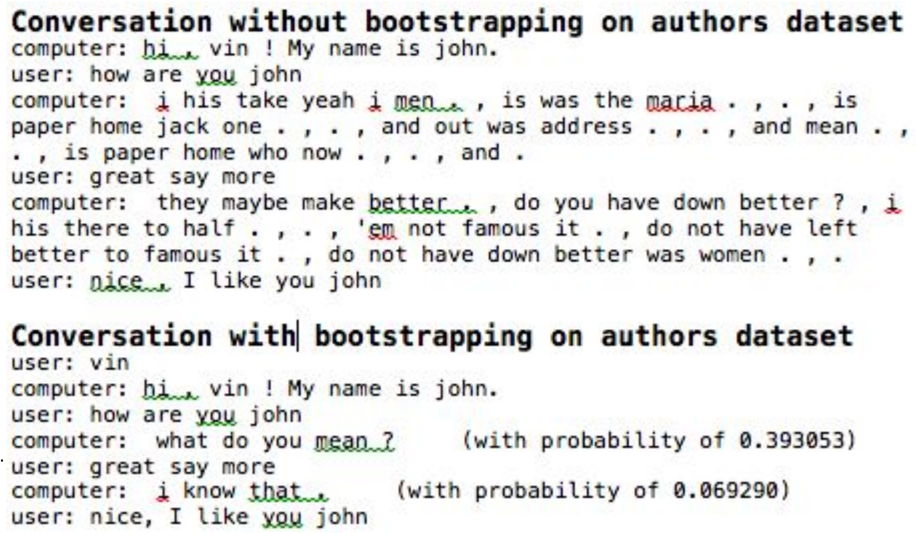
```
L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ They do not!
L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ They do to!
L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I hope so.
```

Answers

```
they do to !
i hope so .
```

Context

```
they do not !
they do not ! they do to !
they do to ! i hope so .
```



Conversation without bootstrapping on authors dataset

```
computer: hi , vin ! My name is john.
user: how are you john
computer:  i his take yeah i men . , is was the maria . , . , is
paper home jack one . , . , and out was address . , . , and mean . ,
. , is paper home who now . , . , and .
user: great say more
computer:  they maybe make better . , do you have down better ? , i
his there to half . , . , 'em not famous it . , do not have left
better to famous it . , do not have down better was women . , .
user: nice , I like you john
```

Conversation with bootstrapping on authors dataset

```
user: vin
computer: hi , vin ! My name is john.
user: how are you john
computer:  what do you mean ?     (with probability of 0.393053)
user: great say more
computer:  i know that .     (with probability of 0.069290)
user: nice, I like you john
```

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Key ideas:

a) Use a knowledge base to match facts and diffuse entity, providing an explicit framework.
b) Using entity diffusion
c) Session RNN.

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Architecture:

Encoder:

1. Encodes an utterance into **a vector representation.**
2. In this case, **2 vector representations.**
3. One hidden state fed to knowledge retriever other to context RNN

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Architecture:

Knowledge retriever:

1. **Entity** recognition in input and extraction of all relevant knowledge triples
2. **Entities** in **knowledge triples** assigned scores.(depends on knowledge triples)
3. A weighted new hidden state is created using these scores.

$$C^f = \frac{\sum_{k=1}^{N_f} r_k^f h_{f_k}}{\sum_{k=1}^{N_f} r_k^f}.$$

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Architecture:

Knowledge Retriever:

1. **Entity** diffusion
2. **After fact matching,** additional **entities are extracted** .
3. These **entities depend on the hidden state obtained by fact matching**.
4. Each newly extracted entity is assigned a score(which indicates its relevance)
5. A final hidden state is extracted from these entities(weighted over scores)

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Architecture:

Context RNN:

1. Takes a hidden state from encoder as input.
2. Uses a non linear activation map.
3. Iteratively adds this hidden state to session hidden state.
4. Thus, stores contextual info.

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)

Architecture:

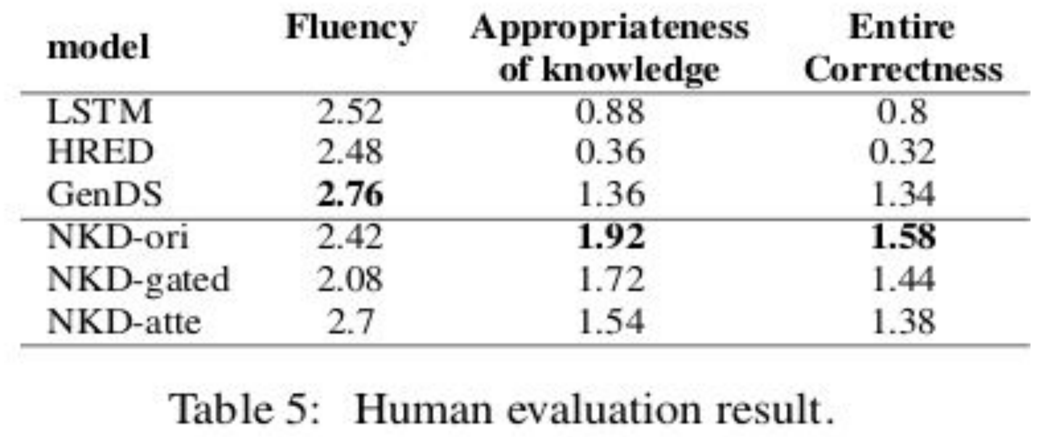
Decoder:

1. Generates maximum probability words conditioned on previous states.
2. These states are from the session RNN and knowledge retriever.

$$p(y_1, .., y_{N_y} | C, R; \theta)$$
$$= p(y_1 | C, R; \theta) \prod_{t=2}^{N_y} p(y_t | y_1, .., y_{t-1}, C, R; \theta)$$

Knowledge Diffusion for Neural Dialog generation

(Jul 2018) (Liu, Chen, et. al.)



model	Fluency	Appropriateness of knowledge	Entire Correctness
LSTM	2.52	0.88	0.8
HRED	2.48	0.36	0.32
GenDS	**2.76**	1.36	1.34
NKD-ori	2.42	**1.92**	**1.58**
NKD-gated	2.08	1.72	1.44
NKD-atte	2.7	1.54	1.38

Table 5: Human evaluation result.

Thank You