A Persona-Based Neural Conversation Model

(Li et al., 2016; ACL)

집현전 2기 중급반 9조
이예진, 전하훈, 구영은

목차

1. Introduction

Natural conversation (open domain)

데이터 기반 시스템의 문제

: training data 중 가장 가능성이 높은 응답을 선택

1. vague or non-committal
2. inconsistent (persona X)

Challenge

1. consistency
2. coherent persona

message	Where do you live now?
response	I live in Los Angeles.
message	In which city do you live now?
response	I live in Madrid.
message	In which country do you live now?
response	England, you?
message	Where were you born?
response	I was born in Canada.
message	Where are you from?
response	England, you?
message	Where did you grow up?
response	I grew up in Texas.
message	How old are you?
response	16 and you?
message	What's your age?
response	18.
message	What is your major?
response	I'm majoring in psychology
message	What did you study in college?
response	English lit.

Table 1: Inconsistent responses generated by a 4-layer SEQ2SEQ model trained on 25 million Twitter conversation snippets.

Persona

정의 및 설명

- character that an artificial agent
- composite of elements of identity (background facts or user profile), language behavior, and interaction style
- 대화 상대에 따라 조정 가능

임베딩으로 표현된 persona를 이용해 사람과 비슷한 대화

2개의 persona-based model

1. Speaker model : 대답을 제공하는 speaker의 persona 임베딩
2. Speaker-Addressee model : speaker + addressee(대화하는 상대)의 persona 임베딩

2. Related work

Data-driven response generation in social media
(Ritter et al. 2011)

- This work follows the line of investigation initiated by Ritter et al(2011).
- generation of conversational dialog as a statistical machine translation (SMT) problem
- dialogue management + response generation
- end-to-end
- purely data-driven
- model learns to converse from human-to-human conversational corpora
- data : Twitter status-response pairs

A neural network approach to context-sensitive generation of conversational responses (Sordoni et al. 2011)

- Ritter et al(2011)의 확장 연구
- Recurrent Neural Network Language Model 사용
- continuous RLM을 사용해 과거의 대화 내용을 기반으로 다음 대답을 생성하는 방법
- context와 message를 알고 있을 때, 다음 response를 생성하는 확률 계산

$$p(r|c,m) = \prod_{t=1}^{T} p(r_t|r_1, \ldots, r_{t-1}, c, m).$$

c : context, 과거에 이루어진 대화들

m : message

r : response, 모델은 message를 입력으로 받으면 response를 출력으로 제공

Building end-to-end dialogue systems using generative hierarchical neural network models (Serban et al. 2015)

- 확장된 대화 기록에 대한 종속성(의존성)을 파악하기 위해 hierarchical neural model 제안
- Hierarchical Recurrent Encoder-Decoder(HRED)

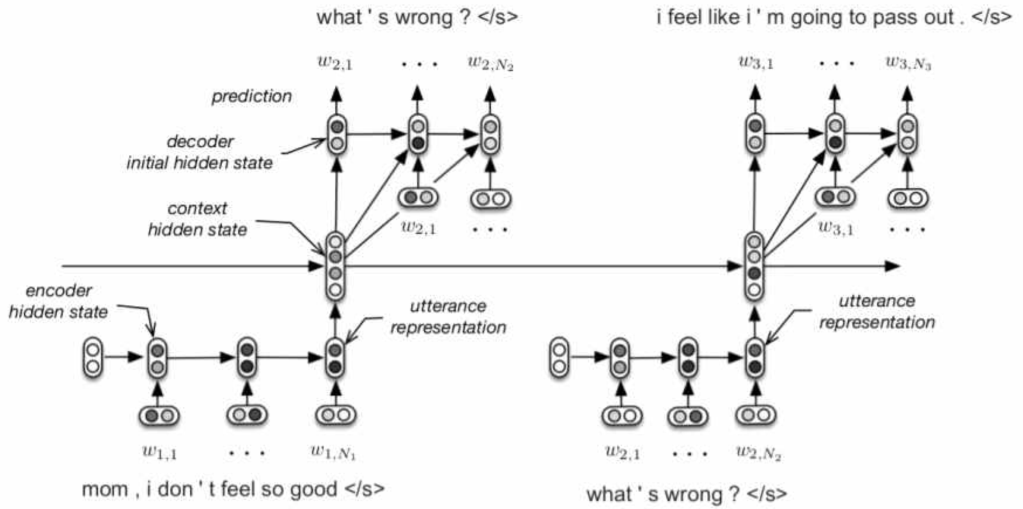


Figure 1: The computational graph of the HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN encodes the tokens appearing within the utterance, and the context RNN encodes the temporal structure of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. Adapted from Sordoni et al. (2015a).

All the World's a Stage : Learning Character Models from Film
(Lin and Walker, 2011)

- how to define both character parameters and
  character models through an automatic corpus based analysis of film screenplays

1. Collect movie scripts from IMSDb;
2. Parse each movie script to extract dialogic utterances, producing an output file containing utterances of exactly one character of each movie (e.g., *pulp-fiction-vincent.txt* has all of the lines of the character Vincent).
3. Select characters we wish to mimic; they must have at least 60 turns of dialogue; this is an arbitrary threshold we set to find leading roles within films;
4. Extract counts (features) reflecting particular linguistic behaviors for each character;
5. Learn models of character types based on these features;
6. Use models to control parameters of the PERSONAGE generator (Mairesse and Walker 2010).

Set:Description
Basic: number of sentences, sentences per turn, number of verbs, number of verbs per sentence
LIWC Word Categories. Anger (hate, kill, pissed), Social processes (talk, us, friend), Friends (pal, buddy, coworker), Causation (because, know, ought), Discrepancy (should, would, could), Assents (yes, OK, mmhmm), Tentative (maybe, perhaps, guess), etc.
Dialogue Act: Accept, Bye, Clarify, Continuer, Emotion, Emphasis, Greet, No-Answer, Reject, Statement, Wh-Question, Yes-Answer, Yes-No-Question, Other
First Dialogue Act: Same as DA but only look at first sentence of each turn.
Pragmatic Markers: Word counts and ratios, plus word category counts: p-taboo, p-seq, p-opinion, p-aggregation, p-softeners, p-emphatics, p-ack, p-pauses, p-concession, p-concede, p-justify, p-contrast, p-conjunction, p-ingroup, p-near-swear, p-relative
Polarity: overall polarity, polarity of sentences, polarity for concessions
Merge Ratio: merging of subject and verb of two propositions
Tag Question Ratio: number of sentences with tag questions out of all sentences
Average Content Word Length: content words are noun, adjective, adverb, and verb; average words' length
Verb Strength: average sentiment values of verbs
Passive Sentence Ratio: number of passive sentences out of all sentences

Table 1: Summary of Feature Sets

추가) 최근 관련 연구

- 발표 논문을 인용한 papers
- **데이터 셋의 구축을 통해 해결해보자**

1) Personalizing Dialogue Agents : I have a dog, do you have pets too? (ACL 2018)
   - Crowd-sourcing으로 페르소나 데이터셋 구축

2) Training Millions of Personalized Dialogue Agents (EMNLP 2018)
   - Crowd-sourcing은 부족, Reddit 데이터를 이용해서 대량의 페르소나 데이터셋 구축

[핑퐁팀 블로그] https://blog.pingpong.us/ml-seminar-season-1/

추가)

Personalizing Dialogue Agents : I have a dog, do you have pets too?

(ACL 2018)

- Amazon Mechanical Turk 이용한 클라우드 소싱
- PERSONA-CHAT이라는 대화 코퍼스 구축
- 데이터 수집

1) Personas : 1155개의 페르소나 클라우드 소싱,
최소 5문장, 1문장 당 최대 15단어

2) Revised Personas : 문장을 Rephrase

3) Persona chat : 두 명의 클라우드 소서들이 랜덤한 페르소나로 서로 대화

"I am a vegetarian. I like swimming. My father used to work for Ford. My favorite band is Maroon5. I got a new job last month, which is about advertising design."

Original Persona	Revised Persona
I love the beach.	To me, there is nothing like a day at the seashore.
My dad has a car dealership	My father sales vehicles for a living.
I just got my nails done	I love to pamper myself on a regular basis.
I am on a diet now	I need to lose weight.
Horses are my favorite animal.	I am into equestrian sports.
I play a lot of fantasy videogames.	RPGs are my favorite genre.
I have a computer science degree.	I also went to school to work with technology.
My mother is a medical doctor	The woman who gave birth to me is a physician.
I am very shy.	I am not a social person.
I like to build model spaceships.	I enjoy working with my hands.

Persona 1	Persona 2
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

추가)
Training Millions of Personalized Dialogue Agents
(EMNLP 2018)

- PERSONA-CHAT 데이터셋의 문제점 보완
- Reddit 코멘트 데이터 이용, 매우 큰 페르소나 데이터셋 구축
  1) Preprocessing : 1.7 billion 댓글로 구성된 Reddit의 덤프 이용, 공백 기준으로 토크나이징
     250k의 고빈도 단어들로 사전 구축
  2) Persona extraction : 유저가 작성한 모든 댓글을 바탕으로 그 유저의 페르소나 작성
  3) Dataset Creation : context, response 구성

- Persona: [“*I like sport*”, “*I work a lot*”]
- Context: “*I love running.*”
- Response: “*Me too! But only on weekends.*”

3. Datasets

Datasets

1) Twitter Sordoni Dataset *(Sordoni et al., 2015)*

- LSTM model(non-persona) 비교용

2) Twitter Persona Dataset

- Speaker Model 학습용
- 6개월 분량의 트위터 (2012.01.01~2012.06.30)

3) Television Series Transcripts

- Speaker-Addressee Model 학습용
- IMSDb(Internet Movie Script Database) *(http://www.imsdb.com)*
- 미국 TV 코메디 『프렌즈(Friends)』, 『빅뱅이론(The Big Bang Theory)』 대본

Datasets

2) Twitter Persona Dataset

- Speaker Model 학습용

- 6개월 분량의 트위터 (2012.01.01~2012.06.30)

ex.

Addressee	index	1	8527	76	4	42	37	58	2361	238	14
	word	**(Sp1)**	**specially**	**as**	**you**	**'re**	**net**	**so**	**gentle**	**yourself**	**!**
Speaker	index	2	10	8996	478	144	175	74	14		
	word	**(Sp2)**	**a**	**billion**	**times**	**better**	**than**	**her**	**!**		

- 4-layer LSTM (1,000 hidden cells for each layer)
- Batch size=128
- Learning rate=1.0
- Parameter initialization: Uniform distribution [-0.1, 0.1]
- Gradient clipping: threshold=5
- Vocabulary size=50,000
- Dropout rate=0.2

Datasets

3) Television Series Transcripts

- Speaker-Addressee Model 학습용
- IMSDb(Internet Movie Script Database) 미국 TV 코메디『프렌즈(Friends)』,『빅뱅이론(The Big Bang Theory)』대본
- 총 13명의 화자
- 69,565개의 턴(turn) - Train(65,565), Dev(2,000), Test(2,000)

ex.

Addressee	index	13	73	578	22	6	3602	18	20	33	16	3032	7
	word	**(Sp13)**	**how**	**clear**	**is**	**the**	**image**	**of**	**me**	**on**	**that**	**screen**	**?**
Speaker	index	11	300	578	2								
	word	**(Sp11)**	**pretty**	**clear**	**.**								

- Pre-training
  - Standard Seq2seq
  - OpenSubtitles(OSDb) dataset 60M-70M 줄의 대사 *(Tiedemann, 2009)*

4. Model

Persona Models

- **Sequence-to-Sequence(Seq2seq)** (LSTM cell)
- Decoding: Greedy, Beam search

1) Speaker Model

각 화자가 발화한 문장들을 바탕으로 화자를 특징짓는 모델

2) Speaker-Addressee Model

각 화자가 상대방에 따라 발화한 문장들을 바탕으로 화자를 특징짓는 모델

→ 화자가 발화한 문장을 바탕으로 화자의 특성을 벡터화

→ 화자의 특성을 고려하여 대화의 응답을 생성(Personalized response generation)

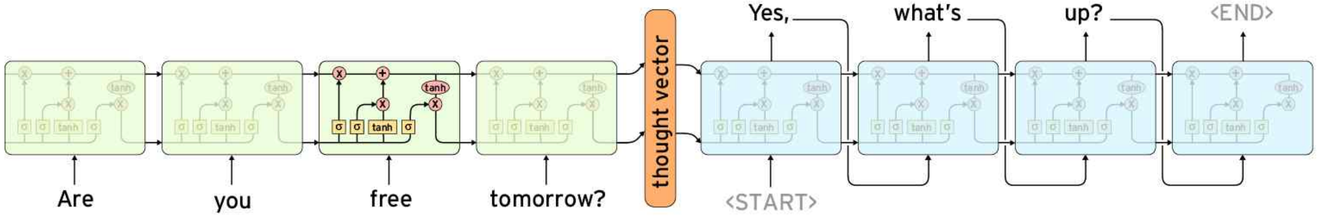
Persona Models

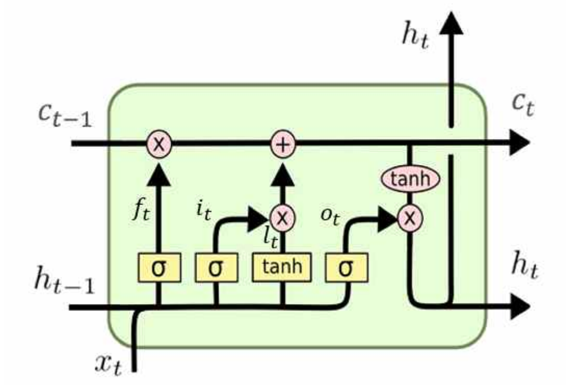
- **Sequence-to-Sequence(Seq2seq)** (LSTM cell)

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \end{bmatrix}$$

$$c_t = f_t * c_{t-1} + i_t * l_t$$

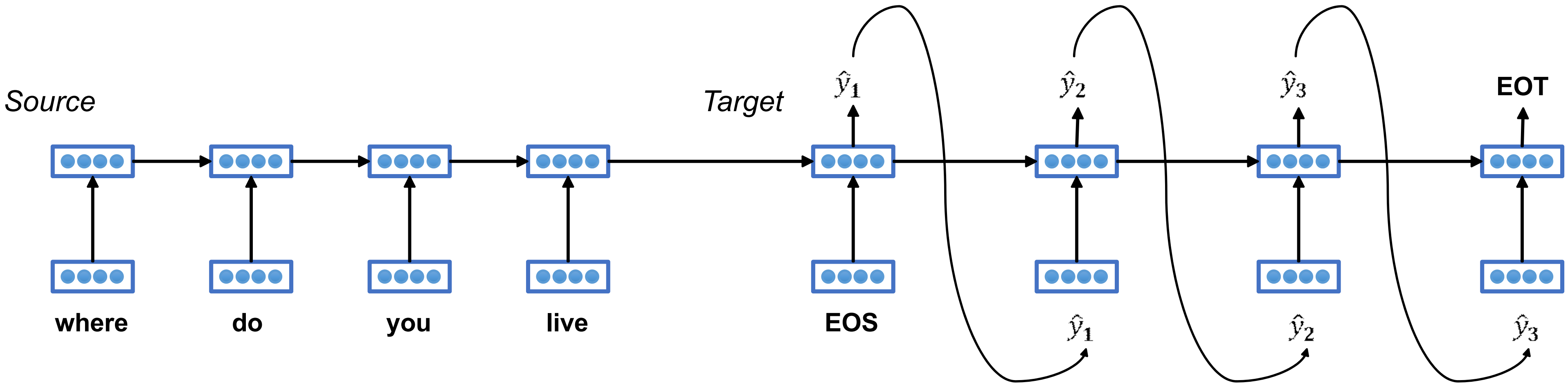
$$h_t^s = o_t * \tanh(c_t)$$





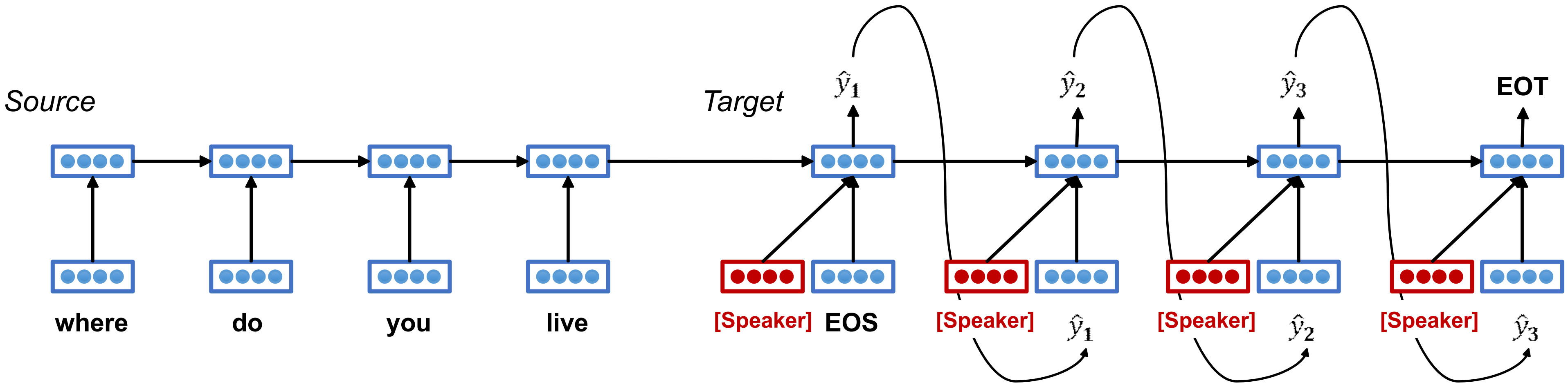
Persona Models

※ EOS : End of Source (Start of Target)
※ EOT : End of Target



Persona Models

※ EOS : End of Source (Start of Target)
※ EOT : End of Target



1) Speaker Model

- 각 화자가 발화한 문장들을 바탕으로 화자를 특징짓는 모델

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ \boldsymbol{v_i} \end{bmatrix}$$

$$c_t = f_t * c_{t-1} + i_t * l_t$$

$$h_t^s = o_t * \tanh(c_t)$$

Speaker embedding $\boldsymbol{v_i}$

$v_i \in \mathbb{R}^{K \times 1}$

1) Speaker Model

- 각 화자가 발화한 문장들을 바탕으로 화자를 특징짓는 모델

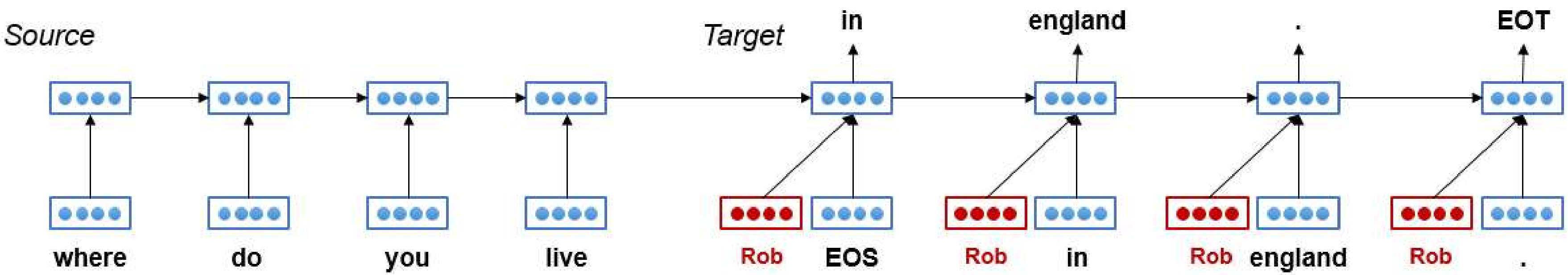
$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ \color{red}{v_i} \end{bmatrix}$$

$$c_t = f_t * c_{t-1} + i_t * l_t$$

$$h_t^s = o_t * \tanh(c_t)$$

Speaker embedding $\boldsymbol{v_i}$

$v_i \in \mathbb{R}^{K \times 1}$



2) Speaker-Addressee Model

- 각 화자가 상대방에 따라 발화한 문장들을 바탕으로 화자를 특징짓는 모델

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ \boldsymbol{V_{i,j}} \end{bmatrix}$$

$W \in \mathbb{R}^{4K \times 3K}$

$V_{i,j} \in \mathbb{R}^{K \times 1}$

$$\boldsymbol{V_{i,j}} = \tanh(W_1 \boldsymbol{v_i} + W_2 \boldsymbol{v_j})$$

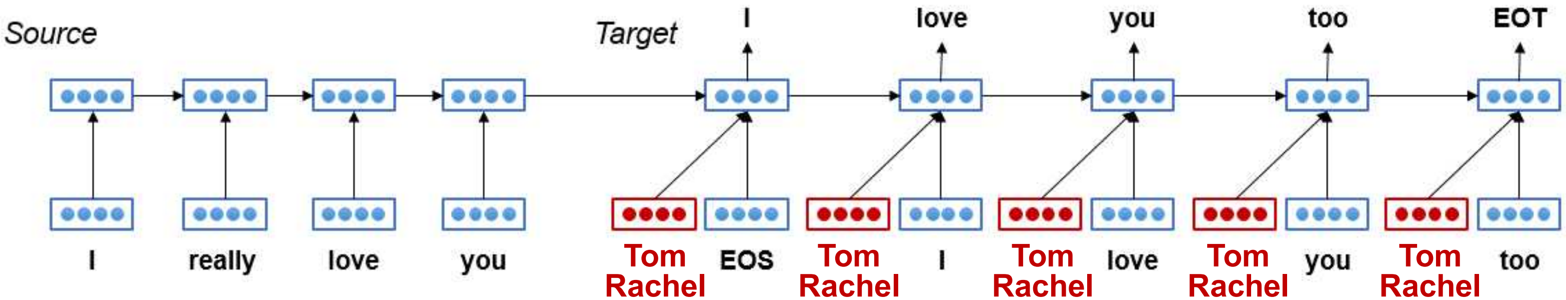
Interactive style of **user** $\boldsymbol{i}$ towards **user** $\boldsymbol{j}$

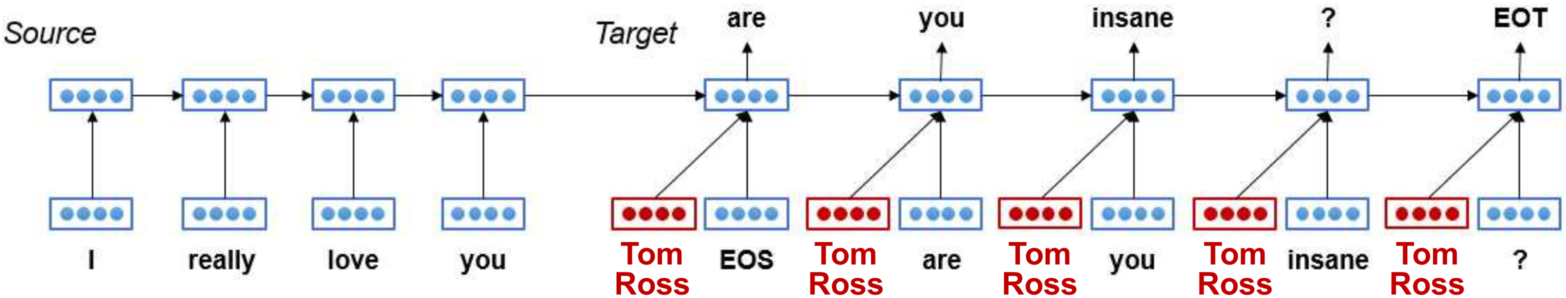
$W_1, W_2 \in \mathbb{R}^{K \times K}$

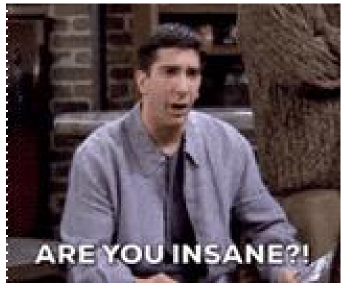
$$c_t = f_t * c_{t-1} + i_t * l_t$$

$$h_t^s = o_t * \tanh(c_t)$$

2) Speaker-Addressee Model







Speaker Embedding

```python
class lstm_target(nn.Module):
```

```python
    if self.speaker:
        self.persona_embedding = nn.Embedding(persona_num, dim)
        self.lstmt = nn.LSTM(dim * 3, dim, num_layers=layer, batch_first=True, bias=False, dropout=params.dropout)
    elif self.addressee:
        self.persona_embedding = nn.Embedding(persona_num, dim)
        self.speaker_linear = nn.Linear(dim, dim)
        self.addressee_linear = nn.Linear(dim, dim)
        self.lstmt = nn.LSTM(dim * 3, dim, num_layers=layer, batch_first=True, bias=False, dropout=params.dropout)
    else:
        self.lstmt = nn.LSTM(dim * 2, dim, num_layers=layer, batch_first=True, bias=False, dropout=params.dropout)
```

```python
def forward(self, context, h, c, embedding, speaker_label, addressee_label):
    embedding = self.dropout(embedding)
    h = self.dropout(h)
    context1 = self.atten_feed(h[-1], context)
    context1 = self.dropout(context1)
    lstm_input = torch.cat((embedding, context1), -1)
    if self.speaker:
        speaker_embed = self.persona_embedding(speaker_label)
        speaker_embed = self.dropout(speaker_embed)
        lstm_input = torch.cat((lstm_input, speaker_embed), -1)
    elif self.addressee:
        speaker_embed = self.persona_embedding(speaker_label)
        speaker_embed = self.dropout(speaker_embed)
        addressee_embed = self.persona_embedding(addressee_label)
        addressee_embed = self.dropout(addressee_embed)
        combined_embed = self.speaker_linear(speaker_embed) + self.addressee_linear(addressee_embed)
        combined_embed = nn.Tanh()(combined_embed)
        lstm_input = torch.cat((lstm_input, combined_embed), -1)
```

1) Speaker Model

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ \boldsymbol{v_i} \end{bmatrix}$$

2) Speaker-Addressee Model

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ \boldsymbol{V_{i,j}} \end{bmatrix}$$

$$\boldsymbol{V_{i,j}} = \tanh(W_1 \boldsymbol{v_i} + W_2 \boldsymbol{v_j})$$

Speaker Embedding

persona_data_twitter_train.txt

```
1 8527 76 4 42 37 58 261 238 14|2 10 8996 478 144 175 74 14
1 21 11 36 26 4 1239 498 874 7|2 21 517 694 498 22 358 2
1 50 10 1400 15 59 19 15648 2|2 22 47 148 257 19 26 7
1 2394 4853 41 10 452 10 1 14|2 24 46 68 28 308 674 2
1 53 12 11 7138 16373 23 216 2|2 4 203 25 26 3076 104 7
```

persona_data_tv_train.txt

```
1 21 22 12 7 22 47 128 31704 1 68 20|3 307 3 307
2 35 3 1 1148 4 101 66 992 10 4476 322|1 170 7 79
13 73 578 22 6 3602 18 20 33 16 3032 7|11 300 578 2
9 24 271 4 432 6 615 87 2918 11 185 2|7 5 85 2
7 5 39 19 787 3 2865 2 32 13 204 2|11 190 1081 7
10 5 546 13 30 143 12 2 5 39 160 2|7 131 4 2
```

```python
def __init__(self, params, voc):
    self.params = params
    self.voc = voc  # EOS: End of source, start of target
    self.EOS = 1    # EOT: End of target
    self.EOT = 2
    self.padding = 0  # Not used, just a reminder
    self.UNK = params.UNK + params.special_word
```

```python
def read_batch(self, file, num, mode='train_or_test'):
    origin = []
    sources = np.zeros((self.params.batch_size, self.params.source_max_length + 1))
    targets = np.zeros((self.params.batch_size, self.params.source_max_length + 1))
    speaker_label = -np.ones(self.params.batch_size)
    addressee_label = -np.ones(self.params.batch_size)
```

```python
            s = line[-2].split()[:self.params.source_max_length]
            t = line[-1].split()[:self.params.target_max_length]
            if s[1:] == []:...
            elif t[1:] == [] and mode != 'decode':...
            source = self.encode(s[1:])
            target = [self.EOS] + self.encode(t[1:]) + [self.EOT]
            l_s = len(source)
            l_t = len(target)
            l_s_set.add(l_s)
            l_t_set.add(l_t)
            origin.append(source)
            sources[i, :l_s] = source
            targets[i, :l_t] = target
            try:
                speaker_label[i] = int(s[0]) - 1
                addressee_label[i] = int(t[0]) - 1
            except:
                print('Persona id cannot be transferred to numbers')
```

Speaker Embedding

persona_data_tv_train.txt

```
1 21 22 12 7 22 47 128 31704 1 68 20|3 307 3 307
2 35 3 1 1148 4 101 66 992 10 4476 322|1 170 7 79
13 73 578 22 6 3602 18 20 33 16 3032 7|11 300 578 2
```

```
sources: (256, 12)
tensor([[  24,   25,   15,  ...,   71,   23,    0],
        [  38,    6,    4,  ..., 4479,  325,    0],
        [  76,  581,   25,  ..., 3035,   10,    0],
        ...,
        [  76,   14,   15,  ...,    0,    0,    0],
        [   7,    4,  900,  ...,    0,    0,    0],
        [  75,    6,  176,  ...,    0,    0,    0]])
----------------------------------
targets: (256, 51)
tensor([[   1,  310,    6,  ...,    0,    0,    0],
        [   1,  173,   10,  ...,    0,    0,    0],
        [   1,  303,  581,  ...,    0,    0,    0],
        ...,
        [   1,   35,   16,  ...,   26,  432,    2],
        [   1,  784,   21,  ...,   21,   89,    2],
        [   1, 5208,  146,  ...,   18,   53,    2]])
----------------------------------
speaker_label: (256,)
tensor([ 0,  1, 12,  8,  6,  9,  3, 10,  2,  8,  3,  0,  6,  2,  1,  6,  8, 10,  8,  2,  3, 10,
         2, 10,  7, 10,  8,  6,  1,  6, 10,  6,  4,  9, 10,  1, 10, 10,  4,  6,  3,  0,  7,  9,
        11,  4, 11,  0,  6,  4,  7,  2, 10,  6, 10,  2,  7,  0,  5, 11,  0,  0,  6,  7,  0,  1,
         2,  2,  4,  2,  6,  3,  2,  4,  7,  6,  5, 12,  6,  7,  3,  6,  8,  8, 10,  0,  7,  0,
         3,  1,  2, 10,  7,  7,  6,  9,  6,  0,  8, 11,  7,  0, 11,  7,  7,  2,  0,  7,  6,  5,
        11,  6,  8,  3,  8,  0,  2,  6,  8,  5,  6,  8,  0,  6,  2,  6,  6,  0, 11,  5,  9,  9,
        11,  6,  4,  2,  4,  7,  0,  0, 11,  1, 11,  5,  2,  2,  0,  3,  4,  1,  7,  0,  7,  6,
         7,  4,  0,  4,  6,  8,  8,  5,  4,  0,  2,  4,  3,  4,  7,  5,  0,  7,  2,  1,  2,  6,
         0,  5,  0,  8,  3,  0,  0,  9,  4,  9,  3,  0,  1,  0,  0,  2,  0,  1,  0,  6, 10,  9,
         2,  9,  1,  5,  3,  7, 11,  3,  7,  0,  3,  7,  6,  0,  2,  1,  3,  4, 10,  3,  6,  6,
         1,  0,  5,  1,  0,  1,  7,  7,  1,  4,  0, 10,  5,  0,  2,  5,  7,  8,  3, 12,  7,  5,
         2,  8,  6,  2,  3,  0,  0,  4,  7,  0,  5,  4,  5,  2])
----------------------------------
addressee_label: (256,)
tensor([ 2,  0, 10,  6, 10,  6,  0,  8,  3,  7,  3,  5,  8,  0,  0,  6,  6, 12,  6,  0,  2,  7,
         5,  7,  6, 11,  6,  7,  2,  6,  9,  8,  2,  7, 12,  3, 11,  6,  2,  7,  4,  2, 10, 12,
         6,  2,  6,  5, 10,  5, 11,  5, 12,  7,  7,  0,  7,  3,  3,  7,  4,  2,  9,  8,  0,  4,
         0,  0,  0,  0,  6,  3,  0,  5,  8,  8,  4,  9,  7,  8,  0,  8,  7,  6, 12,  5,  6,  1,
         4,  4,  1, 12, 11, 10, 10,  6,  8,  2,  6, 11,  8,  2, 10,  6,  8,  0,  2,  7,  7,  2,
        10, 11,  6,  4,  8,  5,  4,  7,  7,  0,  8,  7,  2,  8,  0, 11,  6,  5,  6,  5,  6,  6,
        12,  6,  3,  5,  2,  6,  4,  4,  6,  4,  6,  4,  0,  4,  5,  0,  3,  2, 11,  3,  8,  6,
         6,  5,  1,  5,  9,  6,  6,  4,  1,  2,  1,  5,  4,  1,  6,  0,  3, 10,  5,  4,  1,  9,
         2,  1,  1,  9,  5,  5,  3, 12,  5, 12,  1,  3,  4,  1,  2,  1,  2,  4,  3,  6,  6, 12,
         0,  7,  2,  3,  2,  6,  6,  4,  6,  5,  4, 10,  7,  3,  5,  3,  0,  5,  6,  2,  8,  8,
         3,  2,  0,  4,  2,  0,  6,  6,  4,  5,  3,  6,  4,  3,  0,  4,  8,  6,  4,  6,  8,  5,
         0,  6,  8,  5,  4,  5,  1,  3, 11,  5,  1,  0,  3,  5])
```

```python
def __init__(self, params, voc):
    self.params = params
    self.voc = voc  # EOS: End of source, start of target
    self.EOS = 1    # EOT: End of target
    self.EOT = 2
    self.padding = 0  # Not used, just a reminder
    self.UNK = params.UNK + params.special_word
```

```python
def read_batch(self, file, num, mode='train_or_test'):
    origin = []
    sources = np.zeros((self.params.batch_size, self.params.source_max_length + 1))
    targets = np.zeros((self.params.batch_size, self.params.source_max_length + 1))
    speaker_label = -np.ones(self.params.batch_size)
    addressee_label = -np.ones(self.params.batch_size)
```

```python
            s = line[-2].split()[:self.params.source_max_length]
            t = line[-1].split()[:self.params.target_max_length]
            if s[1:] == []:...
            elif t[1:] == [] and mode != 'decode':...
            source = self.encode(s[1:])
            target = [self.EOS] + self.encode(t[1:]) + [self.EOT]
            l_s = len(source)
            l_t = len(target)
            l_s_set.add(l_s)
            l_t_set.add(l_t)
            origin.append(source)
            sources[i, :l_s] = source
            targets[i, :l_t] = target
            try:
                speaker_label[i] = int(s[0]) - 1
                addressee_label[i] = int(t[0]) - 1
            except:
                print('Persona id cannot be transferred to numbers')
```

Decoding

- Beam size $B = 200$
- Maximum length 20

Reranking

- 포괄적이고 일반적인 답변 위주의 생성을 보완하기 위해

  ex. I don't know
- Beam search 과정에서 가능성 높은 B개를 reranking

$$\log p(\boldsymbol{R}|\boldsymbol{M},\boldsymbol{v}) + \boldsymbol{\lambda}\, \boldsymbol{log}\, \boldsymbol{p}(\boldsymbol{M}|\boldsymbol{R}) + \boldsymbol{\gamma}|\boldsymbol{R}|$$

  - $\log p(R|M,v)$: 질문M과 화자v가 주어졌을 때, 응답R이 나올 확률
  - $\log p(M|R)$: 응답R이 나왔을 때, 질문M이 나올 확률
  - $|R|$: 응답R의 길이 (γ: penalty weight)

5. Experiments

Experiments

1) Evaluation

- *(Sordoni et al., 2015), (Li et al., 2016)* 에서 사용한 BLEU 지표를 사용하여 평가

2) Baseline

- Twitter Sordoni Dataset *(Sordoni et al., 2015)* 을 이용하여 비교

3) Results

- 자신들이 만든 Twitter Persona Dataset을 이용하여 비교
- TV series dataset을 이용하여 비교

4) Qualitative Analysis

- Diverse Responses by Different Speakers
- Human Evaluation

Experiments

2) Baseline

- Twitter Sordoni Dataset *(Sordoni et al., 2015)* 을 이용하여 비교
- SMT baseline *(Ritter et al., 2011)*
- *(Li et al., 2016)* 과 비교했을 때 성능 개선은 더 큰 훈련 말뭉치, dropout사용, 대화의 대화주의적 특성

System	BLEU
MT baseline (Ritter et al., 2011)	3.60%
Standard LSTM MMI (Li et al., 2016)	5.26%
Standard LSTM MMI (our system)	5.82%
Human	*6.08%*

Table 2: BLEU on the Twitter Sordoni dataset (10 references). We contrast our baseline against an SMT baseline (Ritter et al., 2011), and the best result (Li et al., 2016) on the established dataset of (Sordoni et al., 2015). The last result is for a human oracle, but it is not directly comparable as the oracle BLEU is computed in a leave-one-out fashion, having one less reference available. We nevertheless provide this result to give a sense that these BLEU scores of 5-6% are not unreasonable.

Experiments

3) Results

- Twitter Persona dataset을 이용한 비교
- Standard Seq2Seq 모델과 Speaker Model을 비교
- 목적함수 MLE, MMI 모두 일관된 향상에 주목
- MLE 모델에서 더 향상이 큰 이유는 Speaker Model이 더 informative하고 덜 단조로운 출력을 주기 때문

Model	Standard LSTM	Speaker Model
Perplexity	47.2	42.2 (−10.6%)

Table 3: Perplexity for standard SEQ2SEQ and the Speaker model on the Twitter Persona development set.

Model	Objective	BLEU
Standard LSTM	MLE	0.92%
Speaker Model	MLE	1.12% (+21.7%)
Standard LSTM	MMI	1.41%
Speaker Model	MMI	1.66% (+11.7%)

Table 4: BLEU on the Twitter Persona dataset (1 reference), for the standard SEQ2SEQ model and the Speaker model using as objective either maximum likelihood (MLE) or maximum mutual information (MMI).

Experiments

3) Results

- TV series dataset을 이용하여 비교
- Standard Seq2Seq, Speaker Model, Speaker-Addressee Model 모두 비교
- Speaker Model과 Speaker-Addressee Model이 크게 차이가 나지 않는 점은 dataset의 크기 때문
- Speaker Model에서의 Perplexity 차이는 Twitter dataset이 좀 더 noisy하기 때문

Model	Standard LSTM	Speaker Model	Speaker-Addressee Model
Perplexity	27.3	25.4 (−7.0%)	25.0 (−8.4%)

Table 5: Perplexity for standard SEQ2SEQ and persona models on the TV series dataset.

Model	Standard LSTM	Speaker Model	Speaker-Addressee Model
MLE	1.60%	1.82% (+13.7%)	1.83% (+14.3%)
MMI	1.70%	1.90% (+10.6%)	1.88% (+10.9%)

Table 6: BLEU on the TV series dataset (1 reference), for the standard SEQ2SEQ and persona models.

Experiments

4) Qualitative Analysis

- Diverse Responses by Different Speakers
- Twitter dataset에서 무작위의 10명에 화자를 선택
- 다양한 사람들에 대한 구체적인 응답을 생성해낸다.

message	Where is your hometown?	*message*	What company do you work for?	*message*	What did you have for dinner?
baseline	I was born in Canada.	*baseline*	I work for a company.	*baseline*	I had fish and chips.
user1	I'm from England.	*user1*	I don't have a job.	*user1*	I had spag bol.
user2	I'm from Manchester.	*user2*	I'm a manager.	*user2*	Chicken and chips.
user3	I'm from Liverpool.	*user3*	I'm working. #	*user3*	Chicken and rice.
user4	England. You?	*user4*	I work for a company.	*user4*	Fish and chips.
user5	My hometown.	*user5*	I'm working. #	*user5*	I had spag bol.
user6	I'm from Texas.	*user6*	Customer service.	*user6*	I had Mexican food.
user7	I'm from LA.	*user7*	I work in retail.	*user7*	Salad...
user8	England.	*user8*	I work in a cafe.	*user8*	I had chicken and chips.
user9	I was born here.	*user9*	I work at a gas station.	*user9*	I had spag bol.
user10	I was born in the us.	*user10*	I don't know #	*user10*	Pizza.

Table 7: Responses generated by the baseline (LSTM-MMI) and the Speaker Model for ten randomly selected users, without cherry picking. # indicates poor-quality responses produced by the system.

Experiments

4) Qualitative Analysis

- Diverse Responses by Different Speakers
- TV series dataset과 Speaker-Addressee모델 사용
- 구체적인 Addressee에 관한 정보를 생성(ex. 이름)
- Addressee에 관한 대명사를 정확히 식별

message	Do you love me?
baseline	Of course, I love you.
	addressee: Emily, speaker: Sheldon
response	Of course, I love you, Emily.
	addressee: Emily, speaker: Penny
response	Of course, I love you, Emily.
	addressee: Leonard, speaker: Penny
response	Of course, I love you, Leonard.
	addressee: Monica, speaker: Chandler
response	Of course I love you. I want to marry you.
	addressee: Rachel, speaker: Chandler
response	Of course I love you.
	addressee: Ross, speaker: Chandler
response	Of course I love you.
	addressee: Ross, speaker: Rachel
response	Of course I love you. (kisses him)
	addressee: Emily, speaker: Rachel
response	Of course I love you.

Table 8: Responses to *Do you love me?* from the *Speaker-Addressee* model on the TV-series dataset using different addressees and speakers.

Experiments

4) Qualitative Analysis

- Human Evaluation
- Speaker Model에 대해 인적 평가를 위한 실험 설계
- 일관된 응답을 얻을 것으로 기대되는 24쌍의 질문 선택

  ex) What country do you live in?, What city do you live in? …
- the baseline MMI-enhanced Seq2Seq system에서 출력한 응답과 비교
- Persona Model(Speaker Model)이 56.7% 더 일관성 있는 답변을 한다는 결과를 얻음

User1	
message	Where is your hometown?
response	I'm from England.
message	Where are you from?
response	I'm from England.
message	In which city do you live now?
response	I live in London.
message	In which country do you live now?
response	I live in England.
message	What is your major?
response	Business. You?
message	What did you study in college?
response	I did business studies.
message	How old are you?
response	I'm 18.
message	What is your age?
response	I'm 18.
User2	
message	Where is your hometown?
response	I live in Indonesia.
message	Where are you from?
response	England, you?
message	In which city do you live now?
response	I live in Jakarta.
message	In which country do you live now?
response	I live in Indonesia.
message	What is your major?
response	Business, you?
message	What did you study in college?
response	Psychology, you?
message	How old are you?
response	I'm 18.
message	What is your age?
response	I'm 16.

Table 9: Examples of speaker consistency and inconsistency generated by the Speaker Model

6. Conclusions

Conclusions

- Open-domain 대화 생성을 위한 두 가지 Persona모델 제시
- 기분, 감정과 같은 추가적인 특징은 future work.
- BLEU, Perplexity, 인적 평가에서 Speaker의 Consistency는 Baseline보다 뛰어나다.
- Persona모델이 개인적 특성을 포착할 수 있음을 보여줬다.
- Speaker-Addressee 모델에서는 이원적 특성을 포착할 수 있음을 보여줬다.

감사합니다!