A Neural Conversational Model

Oriol Vinyals, Quoc V. Le

발표: 원종국

Abstract

1. Previous conversational modeling
- Restricted to **specific domains**
- Require **hand-crafted rules**

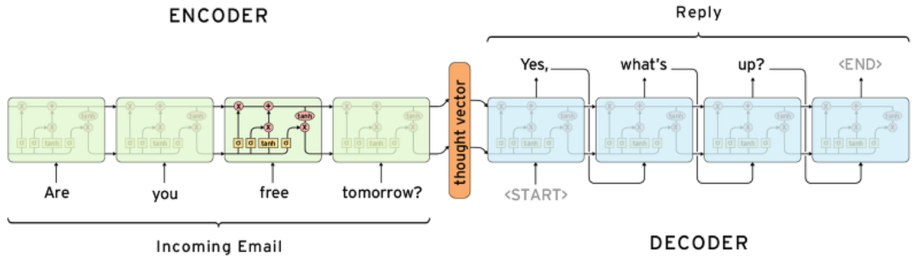
2. ‘Sequence to Sequence’ conversational modeling
- Our model converses by **predicting the next sentence** given the **previous sentence or sentences** in a conversation.
- It can be trained **end-to-end** and thus **requires much fewer hand-crafted rules.**
- **Can generate simple conversations** given a **large conversational training dataset.**

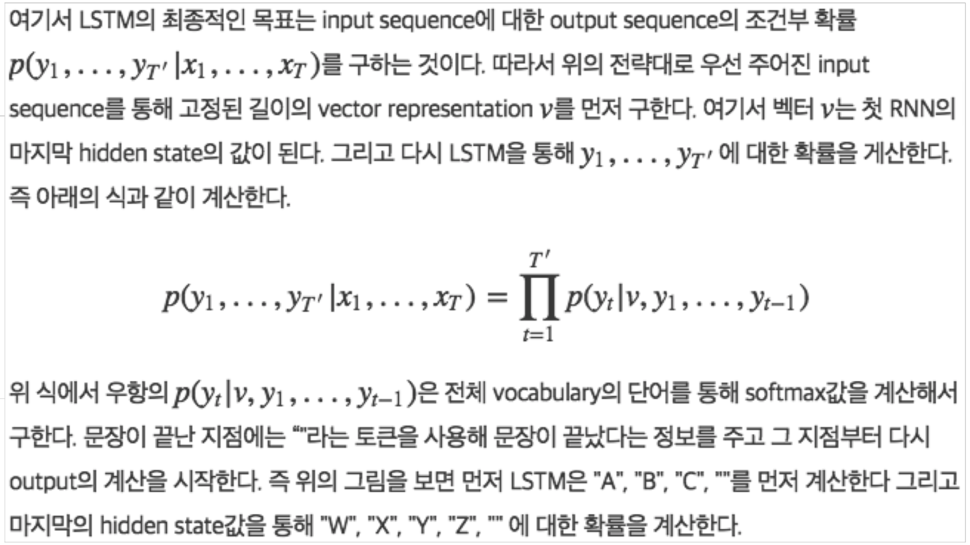
3. Dataset
- Despite optimizing the **wrong objective function**, the model is able to converse well.
- It is able extract knowledge from both a **domain specific dataset**, and from a **large, noisy, and general domain dataset.**
- **Lack of consistency** is a common failure mode our model.

Model

Seq2Seq Review

input sequence를 LSTM을 통해 **고정된 길이의 vector**로 만든 후에 다시 그 vector를 LSTM을 통해 **원하는 target sequence**로 구함
-> input과 target의 길이에 대한 정보가 미리 주어지지 않은 sequence data를 다루는 것이 가능



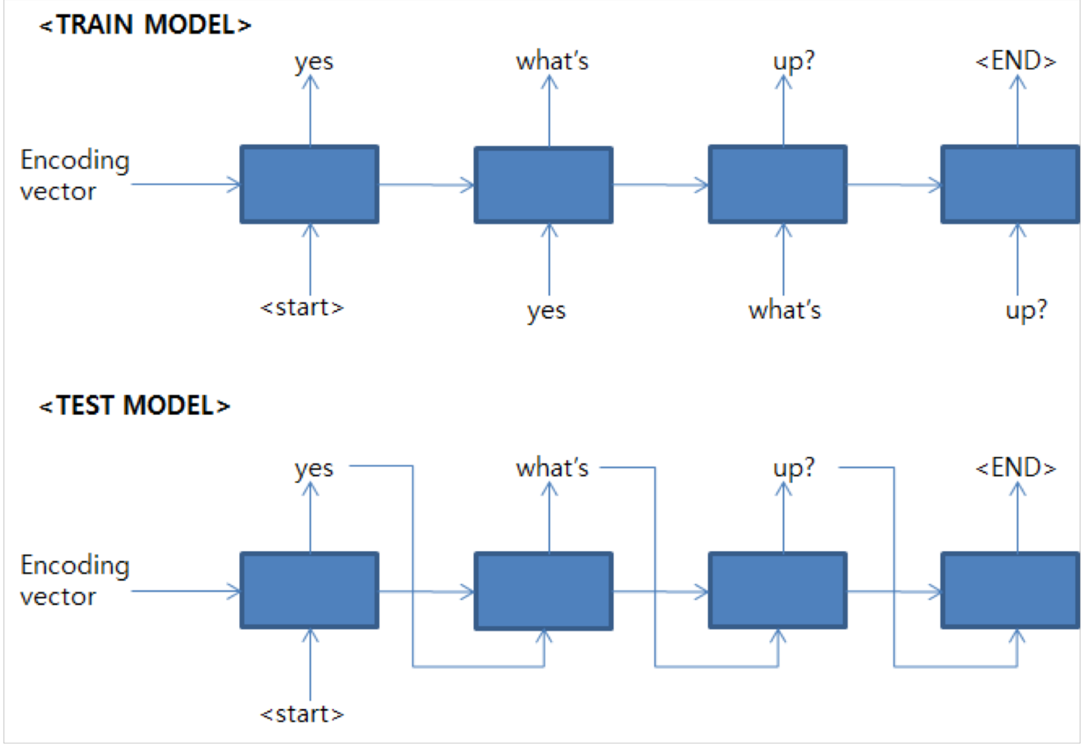


여기서 LSTM의 최종적인 목표는 input sequence에 대한 output sequence의 조건부 확률 $p(y_1,\ldots,y_{T'}|x_1,\ldots,x_T)$를 구하는 것이다. 따라서 위의 전략대로 우선 주어진 input sequence를 통해 고정된 길이의 vector representation v를 먼저 구한다. 여기서 벡터 v는 첫 RNN의 마지막 hidden state의 값이 된다. 그리고 다시 LSTM을 통해 $y_1,\ldots,y_{T'}$ 에 대한 확률을 계산한다. 즉 아래의 식과 같이 계산한다.

$$p(y_1,\ldots,y_{T'}|x_1,\ldots,x_T) = \prod_{t=1}^{T'} p(y_t|v,y_1,\ldots,y_{t-1})$$

위 식에서 우항의 $p(y_t|v,y_1,\ldots,y_{t-1})$은 전체 vocabulary의 단어를 통해 softmax값을 계산해서 구한다. 문장이 끝난 지점에는 “"라는 토큰을 사용해 문장이 끝났다는 정보를 주고 그 지점부터 다시 output의 계산을 시작한다. 즉 위의 그림을 보면 먼저 LSTM은 "A", "B", "C", ""를 먼저 계산한다 그리고 마지막의 hidden state값을 통해 "W", "X", "Y", "Z", "" 에 대한 확률을 계산한다.

출처: http://yujuwon.tistory.com/entry/TENSORFLOW-seq2seq-기반-챗봇-만들기 & https://reniew.github.io/35/



encoder 부분에서는 입력 응답을 받아 하나의 hidden code 값으로 표현을 해주고 decoder 영역에서는 hidden code 값과 start tag를 받아 가장 적합한 결과 단어들을 추출해 준다. 여기서 train과 test의 모델이 각각 다르게 나타나는데, **train의 경우에는 decoder의 output과는 별개로 훈련 셋이 input으로** 들어가는데 반해 test 모델의 경우에는 decoder의 output이 다시 input으로 들어가게 된다. **TRAIN 경우에는 올바른 결과 값을 예상할 수 없기 때문에 이전 output을 feed로 받지 않는 게 맞는 설명이며, TEST 시에는 응답 결과 셋이 없기 때문에 이전 output으로 feed를 받아야 한다.**

출처: http://yujuwon.tistory.com/entry/TENSORFLOW-seq2seq-기반-챗봇-만들기

The strength of this model lies in its simplicity and generality. We can use this model for machine translation, question/answering, and conversations without major changes in the architecture. Applying this technique to conversation modeling is also straightforward: the input sequence can be the concatenation of what has been conversed so far (the context), and the output sequence is the reply.

Unlike easier tasks like translation, however, a model like sequence-to-sequence will not be able to successfully “solve” the problem of modeling dialogue due to several obvious simplifications: **the objective function being optimized does not capture the actual objective achieved through human communication, which is typically longer term and based on exchange of information rather than next step prediction. The lack of a model to ensure consistency and general world knowledge is another obvious limitation of a purely unsupervised model.**

Datasets

1. **Domain specific dataset(**IT helpdesk troubleshooting chat service)
   - customers face computer related issues, and a specialist help them by conversing and walking through a solution.
   - Typical interactions (or threads) are 400 words long, and turn taking is clearly signaled.
   - Our training set contains 30M tokens, and 3M tokens were used as validation.
   - Some amount of clean up was performed, such as removing common names, numbers, and full URLs.

2. **Open-domain conversation dataset**(Movie subtitles)
   - This dataset consists of movie conversations in XML format. It contains sentences uttered by characters in movies.
   - We applied a simple processing step removing XML tags and obvious non-conversational text (e.g., hyperlinks) from the dataset.
   - As turn taking is not clearly indicated, we treated **consecutive sentences assuming they were uttered by different characters.**
   - Unlike the previous dataset, the OpenSubtitles is quite large, and rather noisy because **consecutive sentences may be uttered by the same character.**
   - Our training and validation split has 62M sentences (923M tokens) as training examples, and the validation set has 26M sentences (395M tokens).

We trained our model to predict the next sentence given the previous one, and we did this for every sentence (**noting that this doubles our dataset size, as each sentence is used both for context and as target**).
The split is done in such a way that **each sentence in a pair of sentences either appear together in the training set or test set but not both.**

Experiments

IT helpdesk troubleshooting

- we trained a **single layer LSTM with 1024 memory cells** using **stochastic gradient descent** with **gradient clipping.**
- The vocabulary consists of the **most common 20K words**, which includes special tokens indicating turn taking and actor.

- At convergence, this model achieved a perplexity of 8, whereas an n-gram model achieved 18.

Open subtitles experiment

- we trained a **two-layered LSTM using AdaGrad with gradient clipping**. Each layer of the **LSTM has 4096 memory cells**,
- and we built a vocabulary consisting of the **most frequent 100K words**.
- **To speed up the softmax**, we project the memory cells to **2048 linear units before feeding the information to the classifier**.

- At convergence the perplexity of the recurrent model on the validation set is 17. Our smoothed 5-gram model achieves a perplexity of 28.
- Interestingly, **adding the soft attention mechanism** of (Bahdanau et al., 2014) **did not significantly improve** the perplexity on neither training or validation sets.

We find it encouraging that the **model can remember facts, understand contexts, perform common sense** reasoning without the complexity in traditional pipelines. What surprises us is that the model does so **without any explicit knowledge representation component** except for the **parameters in the word vectors.**

Perhaps most practically significant is the fact that the **model can generalize to new questions.** In other words, **it does not simply look up for an answer by matching the question with the existing database.** In fact, **most of the questions presented above**, except for the first conversation, **do not appear in the training set.**

Human: *what do you think about tesla ?*
Machine: *he 's a good conductor .*
Human: *what do you think about bill gates ?*
Machine: *he 's a good man .*
Human: *What do you think about messi ?*
Machine: *he 's a great player .*
Human: *what do you think about cleopatra ?*
Machine: *oh , she 's very regal .*
Human: *what do you think about england during the reign of elizabeth ?*
Machine: *it was a great place .*

Nonetheless, one drawback of this basic model is that **it only gives simple, short, sometimes unsatisfying answers** to our questions as can be seen above.

Perhaps a more problematic drawback is that the **model does not capture a consistent personality**. Indeed, if we ask not identical but semantically similar questions, the answers can sometimes be inconsistent. This is expected due to the **simplicity of our model and the dataset** in our experiments. The conversation below illustrates such failure:

Human: *what is your job ?*
Machine: *i 'm a lawyer .*
Human: *what do you do ?*
Machine: *i 'm a doctor .*

Discussion

In this paper, we show that a simple language model based on the *seq2seq* framework can be used to train a conversational engine. Our modest results show that it can generate simple and basic conversations, and extract knowledge from a noisy but open domain dataset.

Even though the model has obvious limitations, it is surprising to us that a purely data driven approach without any rules can produce rather proper answers to many types of questions.

However, the model may **require substantial modifications to be able to deliver realistic conversations.** Amongst the many limitations, **the lack of a coherent personality makes it difficult for our system to pass the Turing test**(Turing, 1950).