Building a Dialog Agent

Assignment 3 - NLP Course - Naver Labs - CentraleSupelec - Spring 2019

Paul Asquin
OSY/MSc in AI
paul.asquin@student.ecp.fr

Benoit Laures
OSY/MSc in AI
benoit.laures@student.ecp.fr

Ayush K. RAI
MSc in AI
ayush.rai2512@student-cs.fr

1 RETRIEVAL BASED DIALOG AGENT

In this work, we implemented a retrieval based dialog agent. Retrieval based models have a set of pre-defined responses in their stack, which they can rank and output the best response given an utterance or a context. Formally the input to a retrieval-based model is a context c (the conversation up to this point) and a set of potential responses r. The retrieval based dialog agent outputs a score for all the response in the set r and then chooses the one with the highest score.

Note : Before moving ahead we would like to mention that in this report utterance and context would be used to refer to the same idea. Similary distractor and response would point to the same notion and finally meaning of correct answer and groundtruth would be the same.

1.1 Data Preprocessing

The dataset used for this task is PersonaChat[3], which was also used for The Conversational Intelligence Challenge 2 at NeurIPS 2018. The format of the training and validation is given as follows :

<Person 1 Persona>
<Persona 2 Persona>
Utterance1 <TAB> CorrectAnswer <TAB> Distractor1 | Distractor2 | ... | DistractorN

The idea is to convert this problem into classification task. For every dialog we reogranize the training dataset in the way illustrated in Table 1. It is evident from Table 1 that we consider every utterance in the same dialog as an independent training example. We believe given the gigantic size of the dataset such assumption will be a minor issue. Similarly the trained model would be highly biased towards negative training examples but huge training dataset size of compensate for such problems.

It is important to note that we don't make any assumptions regarding the number of distractors for every utterance. In the table we have used N_1 (number of responses for Utterance1), N_2 (number of responses for Utterance2), N_3 (number of utterances for a given dialogue), N_4 (number of responses for $Utterance N_3$) just for illustration purpose. We have also taken care that same groundtruth response is not included as a distractor with two contradicting labels. We have avoided such situation. Also we have included the persona information of both person1 and person2 as a part of the context. Since this transformed dataset is too big to fit in the memory, we write the dataframes as CSV when it reaches 10 000 lines (3 to 4 dialogues). Hence training on a large number of dialogues skyrocket the number of dataframes.

Dialog	Utterance	Response	Label
$Dialog1$	$Utterance1$	$Response1$ (Groundtruth)	1
$Dialog1$	$Utterance1$	$Response2$	0
...	...	...	...
$Dialog1$	$Utterance1$	$ResponseN_1$	0
$Dialog1$	$Utterance2$	$Response1$ (Groundtruth)	1
$Dialog1$	$Utterance2$	$Response2$	0
...	...	...	...
$Dialog1$	$Utterance2$	$ResponseN_2$	0
...	...	...	...
$Dialog1$	$UtteranceN_3$	$Response1$ (Groundtruth)	1
...	...	...	...
$Dialog1$	$UtteranceN_3$	$ResponseN_4$	0

Table 1: Dataset Reorganization for a Dialog

As a standard NLP data preprocessing step we tokenize the persona (both for person1 and person2), every utterance, every response and every correct answer in the training dataset. These tokenized words are given unique ids and using Pretrained Glove[2] embeddings a word vector is associated with these unique word ids.

1.2 Model Architecture

In order to address this problem, we utilized the Network Architecture of DualEncoder LSTM proposed in [1], depicted in Fig 1. The main component of the DualEncoder LSTM model is the Encoder, which is described below.

1.2.1 ***Encoder:*** The Encoder part has three major components

- **Embedding layer**: The embedding layer is initialized with pre-trained GloVe[2] vectors for those words available, otherwise randomly initialized (from standard normal distribution), fine-tuned during training. The dimension of the embedding layer is vocabulary size × embedding dimension.
- **LSTM layer**: The next layer is unidirectional, single-layer LSTM with input-to-hidden weights initialized from a uniform distribution and hidden-to-hidden weights with orthogonal initialization as mentioned in [1]. At each time step, one word vector of the input utterance is fed into the LSTM and the hidden state is updated. For this classification task, we are only interested in the last hidden state of each input sequence, which can be interpreted as a numerical summary of the input utterance.
- **Dropout layer**: A dropout layer was added to the model such that it is applies directly to the last hidden state of each

input sequence. The dropout probability hyperparameter is set between 0.1 and 0.5.

1.2.2 ***DualEncoder LSTM:.*** To obtain the dual encoder model, one instance of the encoder model is applied to the context utterance and subsequently to the response utterance for each training example. The two outputs (the last hidden state of the context input sequence, denoted as c, and the last hidden state of the response input sequence, denoted as r) are then used to calculate the probability that this is a valid pair: a weight matrix M is initialized as another learnable parameter (in addition to the embedding weights and the LSTM weights) and used to map between c and r. This can be interpreted as a predictive approach: given some context c, by multiplying c with M we predict what a possible response r' could look like. We then measure the similarity of r' to the actual response r using the dot product and convert it to a probability using the sigmoid function.

$$p(label = 1|c, r, M) = \sigma(c^T Mr) = \sigma(r, r')$$

A high similarity will result in a high dot product and a sigmoid value that goes towards 1. The model is trained by minimizing the binary cross entropy loss. To select the best answer we then take the one with the highest probability.

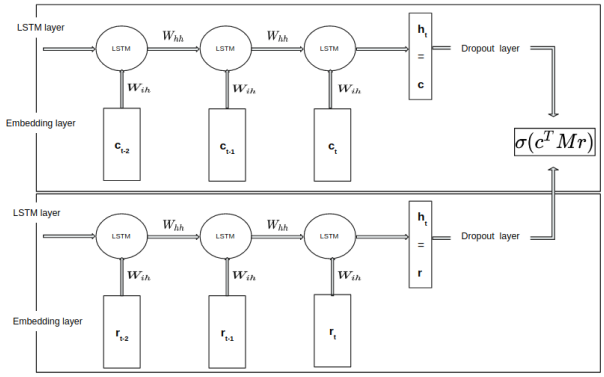

Figure 1: Dual Encoder LSTM

2 RESULTS AND EXPERIMENTS

We made several experiments with the training procedure by varying multiple hyper-parameters like initial learning rate, dropout probability, weight decay etc. Here we present some of our experimental results.

As we can see, the model heavily overfits the training data. We attempted to limit it by varying the dropout value (especially with high values) or the L2 weight decay but it showed limited effectiveness. An interesting solution would be to train on much more data but the size of the dataset was too big for our computers to handle and the training phase becomes way too long. Finally, we also tried to increase the embedding size as pretrained Glove embeddings were also available in dimensions 50, 100, 200; however increasing this size would also explode the training time.

2.0.1 ***Experiment 1***

.

Num DIALOGUES TRAIN = 50
Num DIALOGUES VAL = 50
NUM EPOCHS = 25
EMBEDDING DIM = 50
HIDDEN LAYER SIZE = 50
INITIAL LEARNING RATE = 1e-4
L2 PENALTY= 1e-4
Dropout Probability = 0.5
Retrieval Accuracy on Training Dataset = 34.6%
Retrieval Accuracy on Validation Dataset= 4.62%

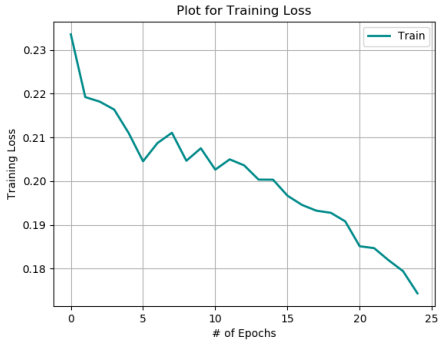

Figure 2: Experiment 1

2.0.2 ***Experiment 2***

.

Num DIALOGUES TRAIN = 100
Num DIALOGUES VAL = 100
NUM EPOCHS = 10
EMBEDDING DIM = 50
HIDDEN LAYER SIZE = 50
INITIAL LEARNING RATE = 1e-4
L2 PENALTY= 1e-4
Dropout Probability = 0.5
Retrieval Accuracy on Training Dataset = 97.4%
Retrieval Accuracy on Validation Dataset = 6.3%

2.0.3 ***Experiment 3***

.

Num DIALOGUES TRAIN = 150
Num DIALOGUES VAL = 150
NUM EPOCHS = 5
EMBEDDING DIM = 50
HIDDEN LAYER SIZE = 50
INITIAL LEARNING RATE = 1e-4
L2 PENALTY= 1e-2
Dropout Probability = 0.5
Retrieval Accuracy on Training Dataset = 3.6%
Retrieval Accuracy on Validation Dataset = 5.2%

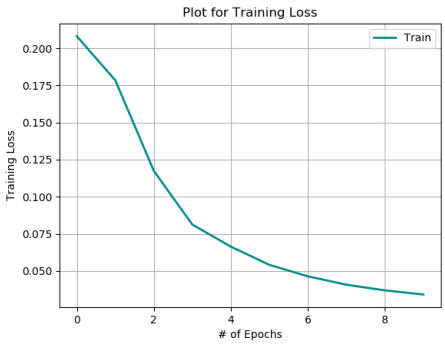

Figure 3: Experiment 2

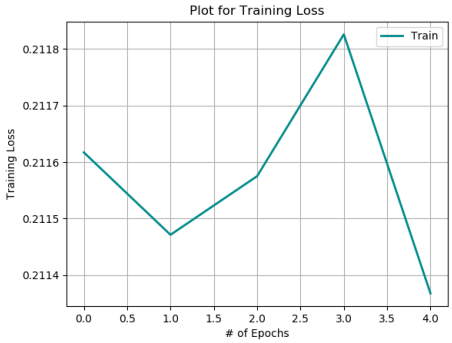

Figure 4: Experiment 3

3 CONCLUSION

In this task, we worked on Retrieval Based Dialog Agent system using Dual Encoder LSTM based model and performed empirical analysis of the results by varying various hyperparameters. In future we would like to use Seq2Seq model for this task.

REFERENCES

[1] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* (2015).

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[3] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243* (2018).