

Chatbot helping with order placements

Dataset

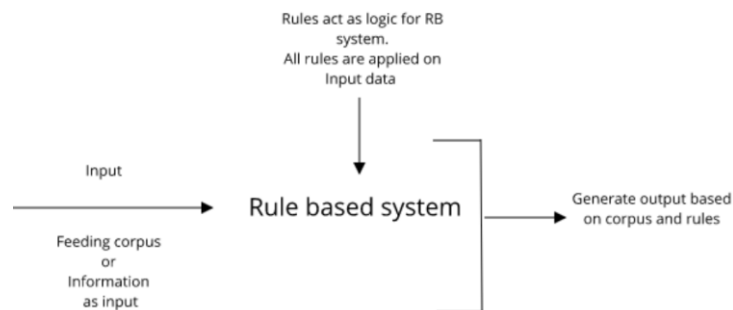
We chose Facebook's bAbI restaurant reservation dataset, which consists of 6 tasks, the first 4 testing the bot's capability to understand questions, issue appropriate API calls (Task 1), update API calls if needed (Task 2), returning the results of the API calls (Task 3) and providing extra information about the chosen restaurant if asked (Task 4). Tasks 5 and 6 are comprised of full dialogs, the difference between them being that in task 6 the bot has to keep track of the previous questions and answers.



Another dataset that we chose to test our bot on is IBM's permuted bAbI tasks. According to their paper, "In original-bAbI dialog tasks at a given time in the conversation, there is only one correct system utterance. Permuted-bAbI dialog tasks allow multiple correct system utterances at a given point in the conversation" [2].

Methods used by Facebook researchers:

- **RULE-BASED SYSTEMS:** “Our tasks T1-T5 are built with a simulator so as to be completely predictable. Thus it is possible to hand-code a rule based system that achieves 100% on them” [1]



- **TF-IDF Match:** For each possible candidate response, a matching score between the input and the response is computed, and rank the responses by score. The score is the TF-IDF weighted cosine similarity between the bag-of-words of the input and bag-of-words of the candidate response

$$TFIDF \text{ score for term } i \text{ in document } j = TF(i, j) * IDF(i)$$

where

IDF = Inverse Document Frequency

TF = Term Frequency

$$TF(i, j) = \frac{\text{Term } i \text{ frequency in document } j}{\text{Total words in document } j}$$

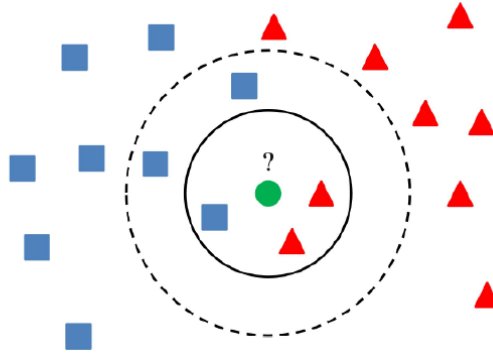
$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term } i} \right)$$

and

t = Term

j = Document

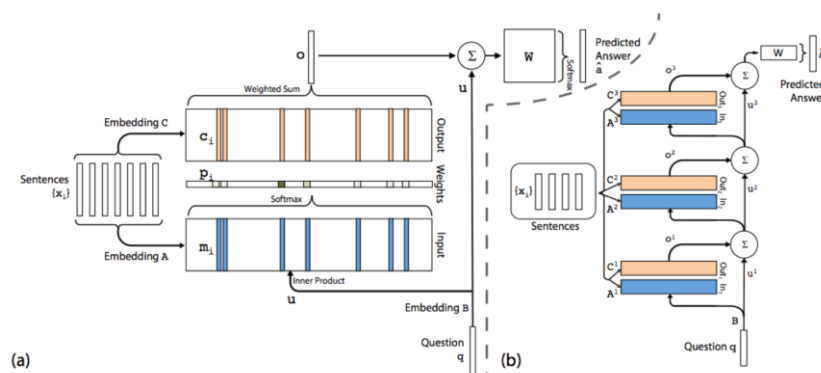
- **Nearest Neighbor:** Using the input, the most similar conversation in the training set is found and the response from that example is outputted.



- **SUPERVISED EMBEDDING MODELS:** “predict the next response given the previous conversation. In this setting a candidate response y is scored against the input x : $f(x; y) = (Ax) \cdot By$, where A and B are $d \times V$ word embedding matrices, i.e. input and response are treated as summed bags-of-embeddings.” [1]

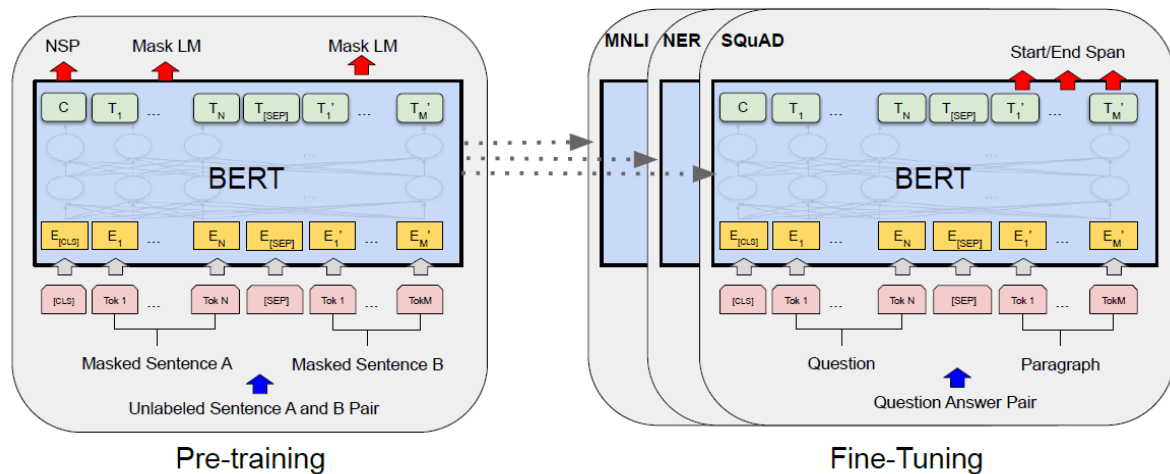
sea	1 0 0 .. 0
salt	0 1 0 .. 0
is	0 0 1 .. 0
a	0 .. 1 .. 0
great	0 0 .. 1 0 0

- **MEMORY NETWORKS:** The researchers used MemN2N architecture.

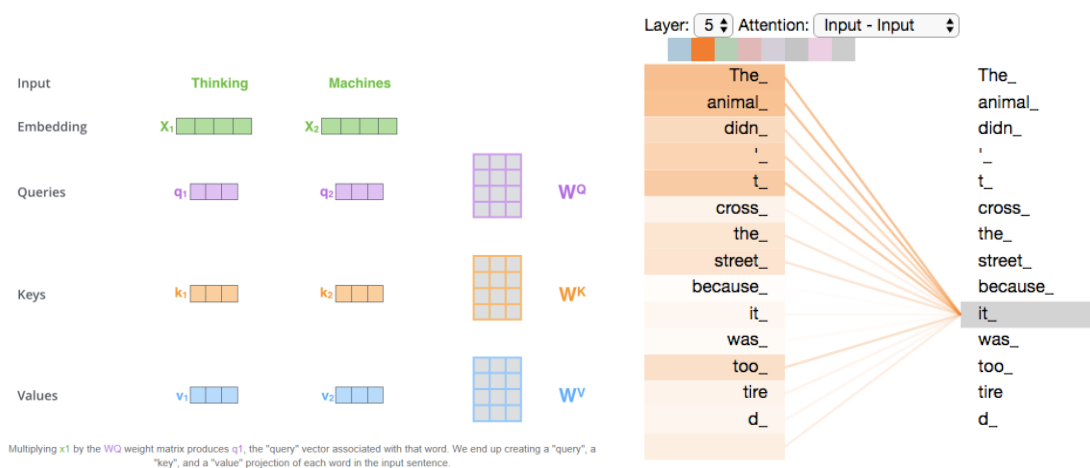


Methodology

In our approach we would like to use BERT. During our research we discovered that BERT is a pretrained network which can be easily fine tuned into other tasks. The way it was pretrained is by giving the model two sentences separated by a special embedding called separator ([SEP]) with some words replaced by a special token ([MASK]) and outputting the most probable words that were masked. This way the network learned special embeddings which are then used for other tasks by just changing the output of the network.



One special thing about BERT is the usage of the attention mechanism. The attention is created by dividing a word embedding into query, key and value vectors which are used to create a mapping of "attention" between each two words in a sentence. This way information that is in the beginning of the sentence is not lost along the way to the end of the sentence like in the case of LSTMs.



First Solution:

Data encoding:

Our data is represented as dialogs, for example:

1 hi *hello what can i help you with today*
 2 can you book a table *i'm on it*
 3 <SILENCE> *any preference on a type of cuisine*
 4 i love italian food *where should it be*
 5 in paris *how many people would be in your party*
 6 we will be two *which price range are looking for*
 7 in a cheap price range please *ok let me look into some options for you*
 8 <SILENCE> *api_call italian paris two cheap*

In order to transform this dialog into input-output format we decided to give as input all sentences up to an answer and the answer as output:

input	output
<i>hi #</i>	<i>hello what can i help you with today</i>
<i>hi # hello what can i help you with today # can you book a table #</i>	<i>i'm on it</i>
<i>hi # hello what can i help you with today # can you book a table # i'm on it # <SILENCE> #</i>	<i>any preference on a type of cuisine</i>
<i>hi # hello what can i help you with today # can you book a table # i'm on it # <SILENCE> # any preference on a type of cuisine # i love italian food #</i>	<i>where should it be</i>
<i>hi # hello what can i help you with today # can you book a table # i'm on it # <SILENCE> # any preference on a type of cuisine # i love italian food # where should it be # in paris #</i>	<i>how many people would be in your party</i>
<i>hi # hello what can i help you with today # can you book a table # i'm on it # <SILENCE> # any preference on a type of cuisine # i love italian food # where should it be # in paris # how many people would be in your party # we will be two #</i>	<i>which price range are looking for</i>

hi # hello what can i help you with today #
can you book a table # i'm on it #
<SILENCE> # any preference on a type of
cuisine # i love italian food # where should
it be # in paris # how many people would
be in your party # we will be two # which
price range are looking for # in a cheap
price range please #

ok let me look into some options for you

hi # hello what can i help you with today #
can you book a table # i'm on it #
<SILENCE> # any preference on a type of
cuisine # i love italian food # where should
it be # in paris # how many people would
be in your party # we will be two # which
price range are looking for # in a cheap
price range please # ok let me look into
some options for you # <SILENCE> #

api_call italian paris two cheap

For the input the # separator marks the end of a sentence. It will be replaced with [SEP] when encoding. The vectors have length 108 because the biggest input has 96 tokens. For example:

hi # hello what can i help you with today # can you book a table # i'm on it # <SILENCE>
any preference on a type of cuisine # i love italian food # where should it be # in paris

Step 1: Start with [CLS] and add tokenized sentences separated by [SEP]

['[CLS]', 'hi', '[SEP]', 'hello', 'what', 'can', 'i', 'help', 'you', 'with', 'today', '[SEP]', 'can', 'you', 'book', 'a', 'table', '[SEP]', 'i', 'm', 'on', 'it', '[SEP]', '<', 'silence', '>', '[SEP]', 'any', 'preference', 'on', 'a', 'type', 'of', 'cuisine', '[SEP]', 'i', 'love', 'italian', 'food', '[SEP]', 'where', 'should', 'it', 'be', '[SEP]', 'in', 'paris', '[SEP]']

Step 2: Transform each token into an id in **ids**, label it with the type of sentence it belongs (user or bot) in **masks**, and add its position in the sentence in **segments**.

Token embeddings (**ids**): [101 7632 102 7592 2054 2064 1045 2393 2017 2007
2651 102 2064 2017 2338 1037 2795 102 1045 1005 1049 2006 2009 102
1026 4223 1028 102 2151 12157 2006 1037 2828 1997 12846 102 1045 2293
3059 2833 102 2073 2323 2009 2022 102 1999 3000 102 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0]

Sentence Embedding (**masks**): [0 1 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 2 2
2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 0
0 0]

0 – not sentence
1 – user
2 - bot

[illegible]

The output is represented by 16 integers. First one L = length of the sentence and the next L represent ids of the words of the question (the remaining are padding). The biggest output has 10 tokens.

For example:

how many people would be in your party

Step 1: tokenize output

['how', 'many', 'people', 'would', 'be', 'in', 'your', 'party']

Step 2: start with the length of the tokenized output and add the ids of each token

[8 2129 2116 2111 2052 2022 1999 2115 2283 0 0 0 0 0 0 0]

Model:

Bert with L = 24, H = 1024, A = 16 connected to a Dense layer with 16 units activated by relu function.

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
<i>input_word_ids</i> (InputLayer)	[(None, 108)]	0	
<i>input_mask</i> (InputLayer)	[(None, 108)]	0	
<i>segment_ids</i> (InputLayer)	[(None, 108)]	0	
<i>keras_layer</i> (KerasLayer)	[(None, 1024),	(None 335141889	<i>input_word_ids</i> [0][0] <i>input_mask</i> [0][0] <i>segment_ids</i> [0][0]
<i>tf_op_layer_strided_slice_2</i> (Te	[(None, 1024)]	0	<i>keras_layer</i> [2][1]
<i>dense_4</i> (Dense)	(None, 16)	16400	<i>tf_op_layer_strided_slice_2</i> [0][0]
Total params: 335,158,289			

Trainable params: 335,158,288

Non-trainable params: 1

The compilation of the model was done with `categorical_crossentropy` loss and `Adam(lr=2e-6)` optimiser.

Experiments:

In the experiments EarlyStopping callback was added (although Google Colab interrupted the execution before that every time because of resource limiter) and also ReduceLROnPlateau.

The experiments used only the training data for the first task using `validation_split=0.2` for the fit function.

Best training:

Epoch 1/100

302/302 [=====] - 1785s 6s/step - loss: 79564.6641 - accuracy: 0.5700 - val_loss: 76907.3125 - val_accuracy: 0.5411 - lr: 2.0000e-06

Epoch 2/100

302/302 [=====] - 1726s 6s/step - loss: 76577.6484 - accuracy: 0.6167 - val_loss: 76687.1484 - val_accuracy: 0.6365 - lr: 2.0000e-06

Epoch 3/100

302/302 [=====] - 1740s 6s/step - loss: 76335.5156 - accuracy: 0.6169 - val_loss: 77009.9141 - val_accuracy: 0.5519 - lr: 2.0000e-06

Epoch 4/100

302/302 [=====] - 1757s 6s/step - loss: 76447.7422 - accuracy: 0.6045 - val_loss: 76723.2969 - val_accuracy: 0.7071 - lr: 2.0000e-06

Epoch 5/100

288/302 [=====>..] - ETA: 1:15 - loss: 76674.1797 - accuracy: 0.6263

Final Solution:

Testing the previous solution we observed that the network relies a lot on the number of sentences in order to predict an output. For example, most of the chats in the datasets would refine the 4 parameters so the output “great let me do the reservation” would be the 9th reply most of the time. But if we would simulate a chat where we would like the first option that the bot gave us, we would say “i like it” and then the bot would say “Sure, let me find an other option for you”, the network being used with users that do not like the first option.

To overcome this problem we changed the training encoding of the chats, trimming the conversations after some replies. For example, until the `api_call` we kept the previous

conversation and gave it to the network, in order for it to know what information the user already gave, but after the `api_call` we deleted the previous conversation, because it would not matter anymore in the context of liking or not the option presented by the bot.

We also trimmed some redundant information such as the first two sentences and replies like “i’m on it” followed by `<SILENCE>`.

Data encoding:

```

1 hi      hello what can i help you with today
2 can you book a table      i'm on it
3 <SILENCE> any preference on a type of cuisine
4 i love italian food      where should it be
5 in paris      how many people would be in your party
6 we will be two      which price range are looking for
7 in a cheap price range please      ok let me look into some options for you
8 <SILENCE> api_call italian paris two cheap
9 actually I would prefer for four      sure is there anything else to update
10 no ok let me look into some options for you
12 <SILENCE>      api_call spanish madrid two cheap
13 resto_madrid_cheap_spanish_1stars R_phone resto_madrid_cheap_spanish_1stars_phone
14 resto_madrid_cheap_spanish_1stars R_cuisine spanish
15 resto_madrid_cheap_spanish_1stars R_address resto_madrid_cheap_spanish_1stars_address
16 resto_madrid_cheap_spanish_1stars R_location madrid
17 resto_madrid_cheap_spanish_1stars R_number two
18 resto_madrid_cheap_spanish_1stars R_price cheap
19 resto_madrid_cheap_spanish_1stars R_rating 1
20 resto_madrid_cheap_spanish_6stars R_phone resto_madrid_cheap_spanish_6stars_phone
21 resto_madrid_cheap_spanish_6stars R_cuisine spanish
22 resto_madrid_cheap_spanish_6stars R_address resto_madrid_cheap_spanish_6stars_address
23 resto_madrid_cheap_spanish_6stars R_location madrid
24 resto_madrid_cheap_spanish_6stars R_number two
25 resto_madrid_cheap_spanish_6stars R_price cheap
26 resto_madrid_cheap_spanish_6stars R_rating 6
27 resto_madrid_cheap_spanish_8stars R_phone resto_madrid_cheap_spanish_8stars_phone
28 resto_madrid_cheap_spanish_8stars R_cuisine spanish
29 resto_madrid_cheap_spanish_8stars R_address resto_madrid_cheap_spanish_8stars_address
30 resto_madrid_cheap_spanish_8stars R_location madrid
31 resto_madrid_cheap_spanish_8stars R_number two
32 resto_madrid_cheap_spanish_8stars R_price cheap
33 resto_madrid_cheap_spanish_8stars R_rating 8
34 <SILENCE>      what do you think of this option: resto_madrid_cheap_spanish_8stars
35 no this does not work for me      sure let me find an other option for you
39 it's perfect great let me do the reservation
40 may i have the phone number of the restaurant here it is
   resto_madrid_cheap_spanish_1stars_phone
41 thanks      is there anything i can help you with
42 no thank you      you're welcome

```

<i>input</i>	<i>output</i>
can you book a table #	any preference on a type of cuisine
can you book a table # any preference on a type of cuisine # i love italian food #	where should it be
can you book a table # any preference on a type of cuisine # i love italian food # where should it be # in paris #	how many people would be in your party
can you book a table # any preference on a type of cuisine # i love italian food # where should it be # in paris # how many people would be in your party # we will be two #	which price range are looking for
can you book a table # any preference on a type of cuisine # i love italian food # where should it be # in paris # how many people would be in your party # we will be two # which price range are looking for # in a cheap price range please #	api_call
actually I would prefer for four #	sure is there anything else to update
no #	api_call
no this does not work for me #	sure let me find an other option for you
it's perfect #	great let me do the reservation
may i have the phone number of the restaurant #	phone_call
may i have the phone number of the restaurant # phone_call # thanks #	is there anything i can help you with
no thank you	you're welcome

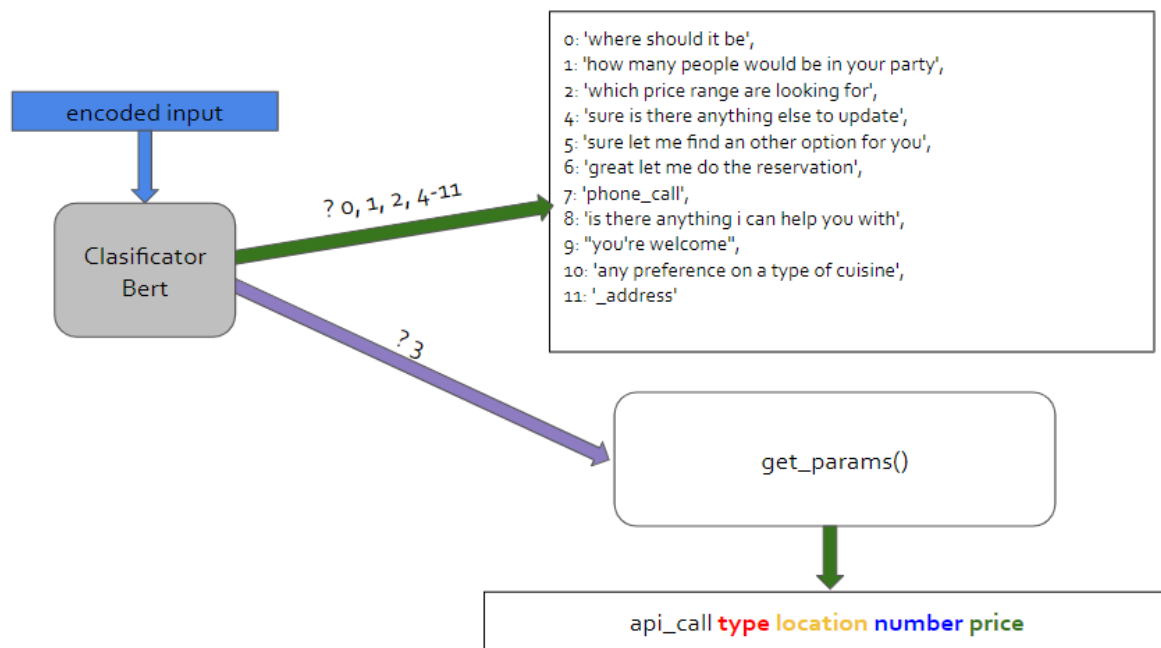
Model:

Bert with L = 12, H = 768, A = 12 connected to Dropout layer and a Dense layer with 12 units activated by softmax function.

Model: "model_7"

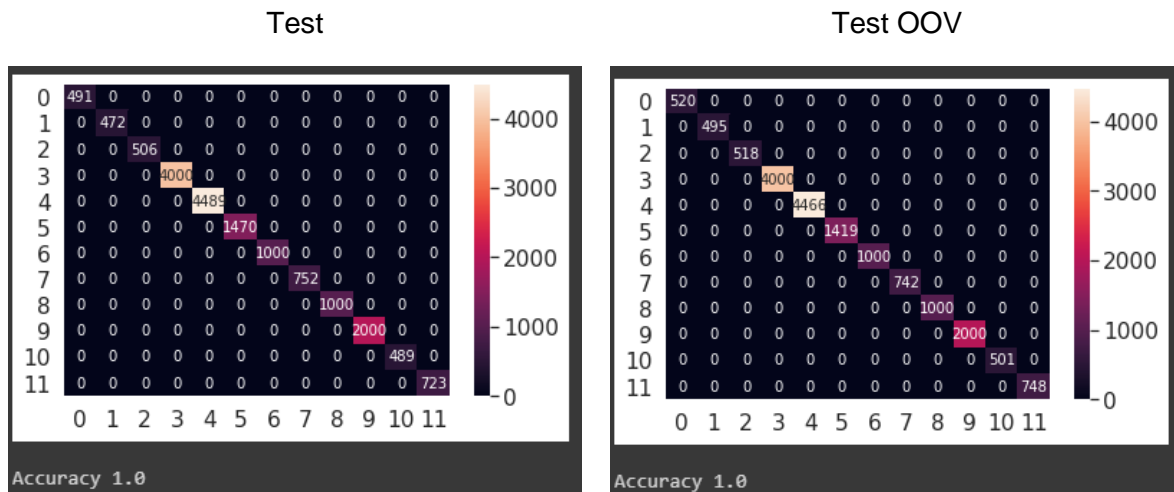
Layer (type)	Output Shape	Param #	Connected to
input_word_ids (InputLayer)	[(None, 100)]	0	
input_mask (InputLayer)	[(None, 100)]	0	
segment_ids (InputLayer)	[(None, 100)]	0	
keras_layer (KerasLayer)	[(None, 768), (None, 109482241		input_word_ids[0][0] input_mask[0][0] segment_ids[0][0]
tf_op_layer_strided_slice_7 (Te	[(None, 768)]	0	keras_layer[7][1]
dropout_7 (Dropout)	(None, 768)	0	tf_op_layer_strided_slice_7[0][0]
dense_7 (Dense)	(None, 12)	9228	dropout_7[0][0]

=====
 Total params: 109,491,469
 Trainable params: 109,491,468
 Non-trainable params: 1
 =====

Architecture of the application:

Results:

This challenge has two tests sets. First test set contains locations, cuisines, number of people and price ranges that can be found in the training data and the second has new ones. However, our architecture works with keyword search so new types of this parameters only have to be added to the `get_params()` function's lists for it to work.



The confusion matrices claim that we have a perfect solution but we came to the conclusion that the training and test sets have similar chats in terms of phrasing and expressions. Also, the dataset is biased, for example “No” is always a answer to the question “Sure is there anything else to update?” and “No thanks” is always an answer to the question “Is there anything i can help you with?”. In our tests we switched the answers and the model acted wrongly.

For example:

```
> Hello
Hello, what can i help you with today?
> may i have a table for four
Any preference on a type of cuisine?
> with spanish cuisine
Where should it be?
> in bombay
Which price range are looking for?
> in a expensive price range please
what do you think of this option:resto_bombay_expensive_spanish_8stars?
> i like it
Great, let me do the reservation.
> may i have the phone number of the restaurant?
Here it is: resto_bombay_expensive_spanish_8stars_phone
> thanks
Is there anything i can help you with?
> no, thanks!
You're welcome.
```

```
> Hello
Hello, what can i help you with today?
> may i have a table for four
Any preference on a type of cuisine?
> with spanish cuisine
Where should it be?
> in bombay
Which price range are looking for?
> in a expensive price range please
what do you think of this option:resto_bombay_expensive_spanish_8stars?
> i like it
Great, let me do the reservation.
> may i have the phone number of the restaurant?
Here it is: resto_bombay_expensive_spanish_8stars_phone
> thanks
Is there anything i can help you with?
> no
what do you think of this option:resto_bombay_expensive_spanish_8stars?
```

References:

- [1] [\[1605.07683\] Learning End-to-End Goal-Oriented Dialog](#) (Facebook's bAbI dialog tasks)
- [2] [\[1808.09996\] Learning End-to-End Goal-Oriented Dialog with Multiple Answers](#) (IBM's permuted bAbI dialog tasks)
- [3] [\[1810.04805\] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#) (BERT)
- [4] [\[1706.03762\] Attention Is All You Need](#) (Transformers and Attention)
- [5] <http://jalammar.github.io/illustrated-transformer/?fbclid=IwAR0a1YXd9a4VCxJuAn5T-wfGwiH3opVvoVdAFk1muxGwk4YDfX2xC-mQioo>