Evaluating Deep Learning Architectures for task specific conversational Systems

Ravindra Abhay Kudchadkar 2018TT10941 tt1180941@iitd.ac.in
Abhinav Gupta 2018TT10868 tt1180868@iitd.ac.in
Aditya Mohan Mishra 2018ME10582 me1180582@iitd.ac.in

1 Introduction

The vastness in Information Retrieval (IR) in today's era has produced numerous methods to overcome the human-machine barrier. Teaching machines to converse with humans is one of the biggest challenges in modern day AI. Many different types of algorithms have been designed in the past decade and we wish to implement and conduct a few ablation experiments on them in this paper.

Most of the algorithms are either single-turn based, where only the last sentence influences the choice of the next response (examples like Bi-LSTM), bag-of-words type where the entire conversation is considered as one big query to be retrieved from a set of responses (like the BM-25), and multi-turn based neural algorithms where via different approaches, the computer is able to process and train models for testing (examples - Deep Attention Matching, Multi-hop Selector Network, Interaction-over-Interaction, Sequential Matching Network).

Our task at hand is to first implement these state-of-the-art models, and then train/test them against a corpus (**Ubuntu Dialog Corpus**). Once that is done, the experiment was to compare findings for different parametric values like maximum number of utterances, interaction blocks, Hop-k values and more. Finally, we conducted statistical significance tests like the Wilcoxon Single-Rank Test and the Friedman test to identify whether our models are similar or different in terms of recall, ranking and average precision.

2 Useful Mechanisms

The following mechanisms of Attention Transformer and Recurrent Units will be used in the upcoming models to be discussed.

2.1 Attention Mechanism

The Attentive Module or the Transformer takes three input sentences: $Q = [e_i]_{i=1}^{n_q}, K = [e_i]_{i=1}^{n_k}, V = [e_i]_{i=1}^{n_v}$ are the Query, Key and Value sentence respectively where e_i represent the dimension embedding and n_q, n_k, n_v are the sentence lengths. Firstly a scalar dot product is used to model the attention of the query words to the words in the key and then a softmax is applied on the matrix obtained. We then the results on the value sentence by a simple batch matrix multiplication:

$$\mathbf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{softmax}\left(\frac{\mathbf{Q}.\mathbf{K^T}}{\sqrt{\mathbf{d}}}\right)\mathbf{V}$$

(Ashish Vaswani et al., 2017)
where $Q[i]$ is the i_{th} embedding in the query sentence Q and d is the length of the word embedding. Each row of V_{att}, denoted as $V_{att}[i]$, consists of the combined semantic representations of words in the value sentence that might possibly have dependencies to the word at the ith position in query sentence. We then sum up The attention part (V_{att}) and the initial query (Q). This is done to increase the complexity of their representation and now the embedding has the joint inference of the initial embedding and the attentive embedding. Many time we encounter the problem of vanishing and exploding gradients. So we also apply a normalisation layer. The result obtained above is then passed through a single layer perceptron (SLP) with a ReLU activation to further process the combined representations:

$$\mathbf{SLP} = \mathit{ReLU}(\mathbf{W_1 x} + \mathbf{b_1})\mathbf{W_2} + \mathbf{b_2}$$

where, $W_1; b_1; W_2; b_2$ are the trainable weights. The result SLP(x) is a 2D-tensor that has the same shape as x (which is a 2D- tensor in the shape of Q), SLP(x) is then added to x, and the result of the combination is then normalized as the final out-

puts. Thus we extract dependency information using this transformer block.

3 Architectures and Implementation

3.1 Okapi BM-25

BM-25 is an old retrieval function based on bag-of-words type algorithm. It uses a linear term-matching based scoring to determine the ranking of the documents. The terms are identified weights in the model with respect to the formula:

$$Score_Q(D) = \Sigma_{i=1}^{qlen} \frac{f_q(D).(k_1+1)w(q)}{f_q(D) + k_1.(1-b+b.\frac{|D|}{D_{avg}})}$$

,where q is a query term, $f_q(D)$ is the frequency of that term in document D, —D— is the document length, D_{avg} is the average length of 10 documents, k1 and b are hyper-parameters and $w(q)$ is the IDF of the term q, given by:

$$w(q) = ln(\frac{N - n_q + 0.5}{n_q + 0.5} + 1)$$

,where N is the number of documents and n_q is the number of documents containing the query term (or the document frequency). Here the assumption according to training data set was that only the first document is considered relevant.

In the training data-set, we had 1 positive and 9 negative responses for each utterance block. We assumed the utterance block as 1 query and then used the 10 responses as our collection of documents. We applied BM-25 on these blocks and then stored the scores of each document.

According to this, the values were $N = 10$ and we assumed k_1 and b as 1.5 and 0.251 respectively.

3.2 Dual-Encoder using bi-LSTM

We fuse all the utterances into a single context. Then we form the word embedding of the context and the response using Glove Vectors. These word embeddings have a size of 50. We pass the context through a bi-LSTM and obtain the final hidden state. We then pass the response through another bi-LSTM and obtain the final hidden state. Then we simply use a matrix M and carry out a batch multiplication of the final hidden state of the context, response and M. Sigmoid activation is used to produce the final score. M is learnable.

3.3 Sequential Matching Network

Initially we have the sentence representations of every utterance in the context and the corresponding response. After some preprocessing we convert them to word embedding by using pre-trained glove vectors. These word embeddings have a size of 50. Thus incorporating 50 features. Then these embedded representations are passed through a GRU and the final cell state is taken as the output for every utterance and response. The number of hidden states of the GRU is taken to be the maximum utterance length. We then make two different matrices M1 and M2. M1 is just a word word similarity matrix, formed by multiplying the initial word embedding of every utterance with the word embedding of the response. The matrix M2 is formed by multiplying the cell state obtained from the GRU for every utterance and response. After this we aggregate the features using a CNN consisting of a single convolution layer with a kernel size of (3,3) and a single pooling layer with a stride of (3,3). The output of the pooling layer is then flattened and passed through a linear layer to obtain a representation of with a dimension of the initial maximum sentence length. This is then passed through a GRU, whose final hidden state is flattened and a single score is obtained (using a sigmoid activation) for a particular context (utterances) response pair.

3.4 Deep Attention Matching

Initially we have the sentence representations of every utterance in the context and the corresponding response. Each of these are sliced or expanded up to a fixed number of utterances and a fixed utterance length. After some preprocessing we convert them to word embedding by using pre-trained glove vectors. The word embeddings are Glove vectors with a dimension of 50. We then exploit the property of **self attention** to make the utterances attend to themselves and also make the response attend to itself. We iteratively pass each of the outputs (for the utterances as well as responses) through the attentive module and form stacked multi-grained representations. If we have iterated N times there will be N levels of granularity. Basically we are making them attend to themselves multiple times. So we stack the self attended representations N times as well as the cross attended. We have chosen N to be 6.

We then make each of these utterances and re-

sponse at each level of granularity to **cross attend** to each other, and form another type of multi grained representations of them. These representations will basically focus on the semantic compositions that are inter dependent in the utterances and the response.

Now we have two types of representations. One self attended and one cross attended for the utterances and the responses. We then take a scalar dot product of the representations of every utterance with the response. In case of self attended multi grain representations this means that the dot product will increase between those segments of the utterance and response which have high textual relevance at a particular level of granularity. For cross attention representations since the inter-dependent parts of the context and the response would be close to each other, so the dot product between these would mean that the model provides dependency aware matching information.

Finally the matrices formed from the dot products are stacked behind each other thus forming a 3D volume. This is basically an aggregation step which stores the matching degree between each level of granularity at every part of the sentence. This model then uses a sequential Convolutional Neural Network consisting of 2 3D convolutions (kernel size = (3,3,3)) and 2 max pooling layers (kernel size = (3,3,3)) to filter out the necessary features from the 3D volume. Simple flattening of the output is done and by using a single linear layer(single layer perceptron) the corresponding score for the context response pair is obtained.

Writing the model Symbolically :

Initially we have the word embeddings at zeroth granularity for every utterance and response:

$$\mathbf{U_i^0} = [\mathbf{e^0_{u_i,0}}, ..., \mathbf{e^0_{u_i,n_{u_i-1}}}], \mathbf{R^0} = [\mathbf{e^0_{r,0}}, ..., \mathbf{e^0_{r,n_{r-1}}}]$$

After self attention at n^{th} granularity:

$$\mathbf{U_i^{n+1}} = \mathbf{Transformer}(\mathbf{U_i^n}, \mathbf{U_i^n}, \mathbf{U_i^n})$$

$$\mathbf{R^{n+1}} = \mathbf{Transformer}(\mathbf{R^n}, \mathbf{R^n}, \mathbf{R^n})$$

After Cross Attention at n^{th} granularity:

$$\mathbf{\tilde{U}_i^{n+1}} = \mathbf{Transformer}(\mathbf{U_i^n}, \mathbf{R^n}, \mathbf{R^n})$$

$$\mathbf{\tilde{R}^{n+1}} = \mathbf{Transformer}(\mathbf{R^n}, \mathbf{U_i^n}, \mathbf{U_i^n})$$

Matrices from dot products :

$$\mathbf{M_{self}^{u_i,r,n}} = \mathbf{U_i^n} \cdot \mathbf{R^n}$$

$$\mathbf{M_{cross}^{u_i,r,n}} = \mathbf{\tilde{U}_i^n} \cdot \mathbf{\tilde{R}^n}$$

After aggregation

$$\mathbf{A} = \mathbf{M_{self}^{u_i,r,n}} \oplus \mathbf{M_{cross}^{u_i,r,n}}$$

where $\oplus$ is the concatenation operation. Here each pixel has 2N+1 channels. This A is then passed through a CNN as described above and the score for a particular C and R is formulated as:

$$\mathbf{score}(\mathbf{c}, \mathbf{r}) = \sigma(\mathbf{w}.\mathbf{f}(\mathbf{c}, \mathbf{r}) + \mathbf{b})$$

where σ is sigmoid and w,b are learn able weights and f is the prediction made for each pair.

3.5 Interaction Over Interaction

Initially we have the sentence representations of every utterance in the context and the corresponding response. After some preprocessing we convert them to word embedding by using pre-trained glove vectors of size 50.

Interaction block : Comprises of three strategies, they are, self attention, interaction and compression. Again the self attention module catches the self dependent segments between every utterance in the context and the response. The interaction strategy leverages the property of cross attention to model the types of interactions between the context and the response pairs. Finally the compression module then takes the input of the above 2 strategies and compresses them so that the output is in the shape of the original word embeddings. This is done by multiplying the elements of the matrix obtained from self attention to that obtained from cross attention through an element wise multiplication. The output is then utilized in the next interaction block to model more complex interactions.

The last step is the aggregation step of all the above matching features. This is done through a simple dot product multiplication of the utterance and response matrices of the self attention, cross attention and the compression step, thus obtaining three similarity matrices for each interaction block. All these matrices are then stacked behind each other, concatenated basically to form a 2D volume. Just like DAM we then utilize a CNN consisting of 2 2D convolutions (kernel size = (3,3)) and 2 max pooling layers (kernel size = (3,3)) to obtain the relevant features from the volume, then by using a single layer perceptron we convert the features into a single D-dimensional

vector to be fed into a Gated recurrent Unit (GRU), to capture the temporal relationship between the segments in the context and the response. Finally after flattening of the last hidden state of the GRU we sum the predictions of each interaction block, not only the last block as during training this can prevent the problem of exploding/vanishing gradients, also each block will capture unique features and utilizing all of these would reduce loss during training. The scores are obtained by passing the predictions through a sigmoid activation.

Writing the model Symbolically :

Initially we have the word embeddings for every utterance and response:

$$\mathbf{U_i^0} = [\mathbf{e^0_{u_i,0}}, ..., \mathbf{e^0_{u_i,n_{u_i}-1}}], \mathbf{R^0} = [\mathbf{e^0_{r,0}}, ..., \mathbf{e^0_{r,n_{r-1}}}]$$

After self attention at n^{th} interaction block:

$$\mathbf{\hat{U}_i^{n+1}} = \mathbf{Transformer}(\mathbf{U_i^n}, \mathbf{U_i^n}, \mathbf{U_i^n})$$

$$\mathbf{\hat{R}^{n+1}} = \mathbf{Transformer}(\mathbf{R^n}, \mathbf{R^n}, \mathbf{R^n})$$

After Cross Attention at n^{th} interaction block:

$$\mathbf{\bar{U}_i^{n+1}} = \mathbf{Transformer}(\mathbf{U_i^n}, \mathbf{R^n}, \mathbf{R^n})$$

$$\mathbf{\bar{R}^{n+1}} = \mathbf{Transformer}(\mathbf{R^n}, \mathbf{U_i^n}, \mathbf{U_i^n})$$

We also make the the embedding at n^{th} layer interact with the cross attention representation through simple element wise multiplication:

$$\mathbf{\tilde{U}_i^{n+1}} = \mathbf{\bar{U}_i^{n+1}} \odot \mathbf{U_i^n}$$

$$\mathbf{\tilde{R}^{n+1}} = \mathbf{\bar{R}^{n+1}} \odot \mathbf{R^n}$$

Matrices formed:

$$\mathbf{M_{1,i}^{n+1}} = \frac{\mathbf{U_i^n} \cdot \mathbf{R^n}}{\sqrt{\mathbf{d}}}$$

$$\mathbf{M_{2,i}^{n+1}} = \frac{\mathbf{\hat{U}_i^n} \cdot \mathbf{\hat{R}^n}}{\sqrt{\mathbf{d}}}$$

$$\mathbf{M_{3,i}^{n+1}} = \frac{\mathbf{\bar{U}_i^n} \cdot \mathbf{\bar{R}^n}}{\sqrt{\mathbf{d}}}$$

Aggregation step:

$$\mathbf{A_i^n} = \mathbf{M_{1,i}^n} \oplus \mathbf{M_{2,i}^n} \oplus \mathbf{M_{3,i}^n}$$

After passing through the CNN as described above, flattening and then passing through a GRU we get the last hidden state as:

$$\mathbf{h_i^n} = \mathbf{GRU}(\mathbf{v^i}, \mathbf{h_i^{n-1}})$$

where v is the vector fed into the GRU at the nth block. Final score is formulated as:

$$\mathbf{score^n}(\mathbf{c}, \mathbf{r}) = \sigma(\mathbf{w}.\mathbf{h_i^n} + \mathbf{b})$$

. The final score is the sum of scores of all blocks.

3.6 Multi Hop Selector Network

At first we have the sentence representations of every utterance in the context and the corresponding response. After some preprocessing we convert them to word embedding by using pre-trained glove vectors of size 50. We then use self attention to make the utterances attend to themselves thus capturing dependency information and then pass it through a unit called as the **context selector**. This unit is used to select only the relevant context. The unit comprises of 2 sub units which are the **word selector** and the **utterance selector**. Using these 2 units we can select promising words and utterances at different levels of granularity.

Word Selector: This leverages cross attention to model the interaction between every utterance of the context and the last utterance in the context and obtain something called as the alignment matrix. We then pool over every row and then pool over every column and obtain the matching features and concatenate them together. The matrix thus obtained tells us which segments from every utterance have matching characteristics with the last utterance. The relevance scores are then obtained from the above by using a softmax activation.

Utterance Selector: The word selector is not able to tell us whether the query and the key are compatible on an higher semantic scale and thus we utilize the utterance selector. Here mean pooling is done on the word dimension on the representation obtained from the word selector to obtain the utterance level embedding. Here we use a similarity measure like cosine similarity to measure the relevance between every utterance and the last utterance.

We then make a representation using both the word and utterance level representations to fuse the scores obtained from both. We basically linearly combine both the representations using an equally weighted sum.

The above is a description of a "Hop 1" selector. Where 1 stands for how many utterances are we comparing the utterances to. In this case its only the last utterance, so 1. But in some cases the last utterance might be too small like "Take Care", so we can also use the last k utterances, yielding k different selectors. The scores obtained from all the selectors are combined to select relevant utterances for response matching. We can use a threshold say γ and for all those utterances whose scores

are below gamma will be eliminated or filtered out using masking. We took k=3 and γ to be 0.3.

Now we have the filtered context and the self attended response representations. We first do a normal dot product and obtain a matrix and then obtain another matrix using cosine similarity. Both of them are then stacked together to form the similarity matrix. This matrix is a part of the origin matching step.

For self matching we pass the newly formed filtered utterances through the attentive module for self attention. Thus giving more and more characterisation. Here too like origin matching a matrix is first formed using dot product and the second is formed using cosine similarity. Then both are stacked together along another dimension to obtain self matching matrix. Similarly we now also use the cross attention between between the utterances and response and form another matrix using the same procedure.

In the last step we aggregate all the three matrices obtained above by concatenating them along a dimension. Again a simple CNN comprising of a 2D convolution (kernel size=(3,3)) and max-pooling (stride=(3,3)) is used to find the relevant features. We then flatten the output to a proper dimension and then pass it through a GRU to find temporal dependencies. The last hidden unit of the GRU encodes the final information and is flattened and passed through a linear layer. Thus giving us the predictions. Scores are obtained by applying a sigmoid function to the prediction.

Writing the model symbolically: Initially we have the word embeddings for every utterance and response:

$$\mathbf{U_i} = [\mathbf{e_{u_i,0}}, ..., \mathbf{e_{u_i,n_{u_i}-1}}], \mathbf{R} = [\mathbf{e_{r,0}}, ..., \mathbf{e_{r,n_{r-1}}}]$$

Let the max number of utterances be L:

$$\hat{\mathbf{U}}_\mathbf{i} = \mathbf{Transformer}(\mathbf{U_i}, \mathbf{U_i}, \mathbf{U_i})$$

Cross attention to find matrix of feature maps

$$\mathbf{A} = \mathbf{Transformer}(\hat{\mathbf{U}}_\mathbf{i}, \hat{\mathbf{U}_\mathbf{L}}, \hat{\mathbf{U}_\mathbf{L}})$$

Mean pooling over dimension 2 and 3:

$$\mathbf{P_i} = [\mathbf{MaxPool}(\mathbf{A_{dim=2}}), \mathbf{MaxPool}(\mathbf{A_{dim=3}}))]$$

The word selector score for a particular utterance is given by a linear layer and σ is softmax:

$$\mathbf{s_{1,i}} = \sigma(\mathbf{w}.\mathbf{P_i} + \mathbf{b}))$$

Mean Pooling over word dimension

$$\tilde{\mathbf{U}} = \mathbf{MeanPool}(\hat{\mathbf{U}}_\mathbf{i})$$

The utterance selector score for U_i is :

$$\mathbf{s_{2,i}} = \frac{\tilde{\mathbf{U}}_\mathbf{i} \cdot \tilde{\mathbf{U_L}}}{||\tilde{\mathbf{U}}_\mathbf{i}||||\tilde{\mathbf{U_L}}||}$$

The final score by the context selector is

$$\mathbf{s_i^1} = \alpha \mathbf{s_{1,i}} + (\mathbf{1} - \alpha)\mathbf{s_{2,i}}$$

From k different selectors:

$$\mathbf{S} = [\mathbf{s_i^1}, \mathbf{s_i^2}, ...\mathbf{s_i^k}]$$

Filtering the context:

$$\bar{\mathbf{U}}_\mathbf{i} = (\mathbf{w} \cdot \mathbf{S}) \odot (\sigma(\mathbf{w} \cdot \mathbf{S}) \geq \gamma) \odot \tilde{\mathbf{U}}_\mathbf{i}$$

Matrices from origin matching, self matching and cross matching through dot product and cosine similarity. Let $\hat{U_i}$ be the filtered utterances:

$$\mathbf{M_{1,i}} = [\hat{\mathbf{U}}_\mathbf{i} \cdot \mathbf{R_i}, \frac{\hat{\mathbf{U}}_\mathbf{i} \cdot \mathbf{R_i}}{||\hat{\mathbf{U}}_\mathbf{i}||||\mathbf{R_i}||}]$$

$$\hat{\mathbf{U}}_\mathbf{i}^\mathbf{self} = \mathbf{Transformer}(\hat{\mathbf{U}}_\mathbf{i}, \hat{\mathbf{U}}_\mathbf{i}, \hat{\mathbf{U}}_\mathbf{i})$$

$$\mathbf{R}^\mathbf{self} = \mathbf{Transformer}(\mathbf{R}, \mathbf{R}, \mathbf{R})$$

$$\mathbf{M_{2,i}} = [\hat{\mathbf{U}}_\mathbf{i}^\mathbf{self} \cdot \mathbf{R}_\mathbf{i}^\mathbf{self}, \frac{\hat{\mathbf{U}}_\mathbf{i}^\mathbf{self} \cdot \mathbf{R}_\mathbf{i}^\mathbf{self}}{||\hat{\mathbf{U}}_\mathbf{i}^\mathbf{self}||||\mathbf{R}_\mathbf{i}^\mathbf{self}||}]$$

$$\hat{\mathbf{U}}_\mathbf{i}^\mathbf{cross} = \mathbf{Transformer}(\hat{\mathbf{U}}_\mathbf{i}, \hat{\mathbf{R}}, \hat{\mathbf{R}})$$

$$\mathbf{R}^\mathbf{self} = \mathbf{Transformer}(\mathbf{R}, \hat{\mathbf{U}}_\mathbf{i}, \hat{\mathbf{U}}_\mathbf{i})$$

$$\mathbf{M_{3,i}} = [\hat{\mathbf{U}}_\mathbf{i}^\mathbf{cross} \cdot \mathbf{R}_\mathbf{i}^\mathbf{cross}, \frac{\hat{\mathbf{U}}_\mathbf{i}^\mathbf{cross} \cdot \mathbf{R}_\mathbf{i}^\mathbf{cross}}{||\hat{\mathbf{U}}_\mathbf{i}^\mathbf{cross}||||\mathbf{R}_\mathbf{i}^\mathbf{cross}||}]$$

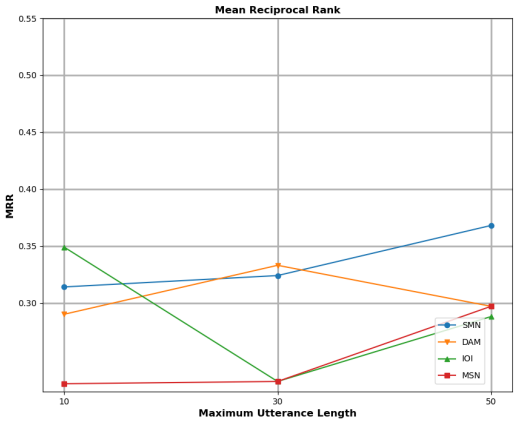
Aggregation step:

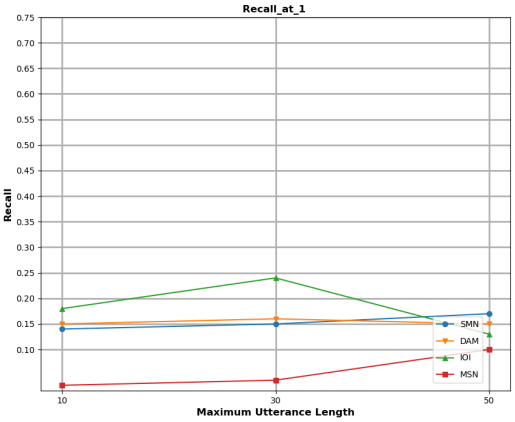
$$\mathbf{M_i} = [\mathbf{M_{1,i}}, \mathbf{M_{2,i}}, \mathbf{M_{3,i}}]$$

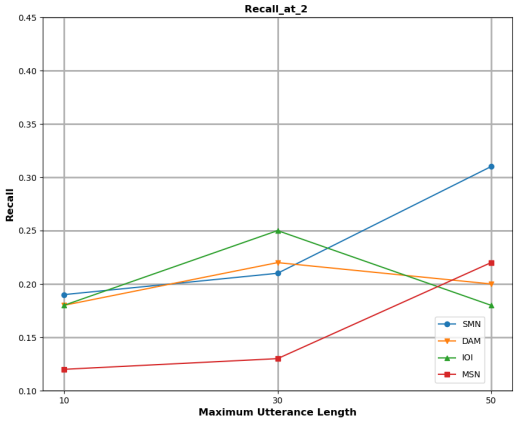
After passing through the CNN as described above, flattening and then passing through a GRU we get the last hidden state as: Final score is formulated as:

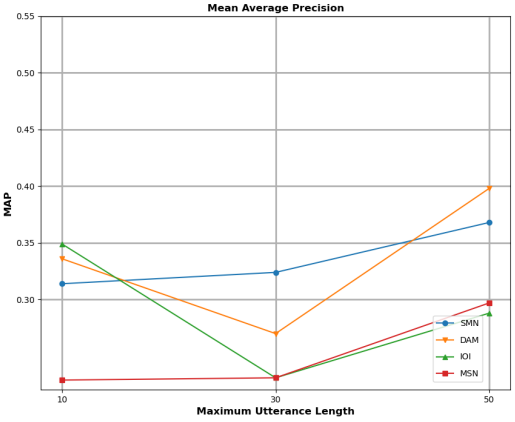
$$\mathbf{score}(\mathbf{c}, \mathbf{r}) = \sigma(\mathbf{w}.\mathbf{GRU}(\mathbf{v}, \mathbf{h^{n-1}}) + \mathbf{b})$$

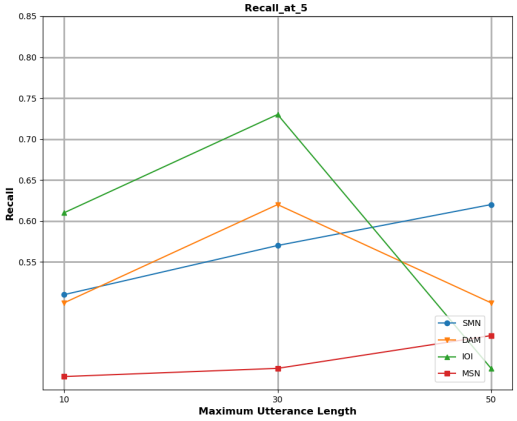
. where v is the vector fed into the GRU.


Mean Reciprocal Rank
MRR
Maximum Utterance Length
SMN
DAM
IOI
MSN


Recall_at_1
Recall
Maximum Utterance Length
SMN
DAM
IOI
MSN


Recall_at_2
Recall
Maximum Utterance Length
SMN
DAM
IOI
MSN


Mean Average Precision
MAP
Maximum Utterance Length
SMN
DAM
IOI
MSN


Recall_at_5
Recall
Maximum Utterance Length
SMN
DAM
IOI
MSN

4 Experiment

What we could and couldn't accomplish:

We trained the bi-LSTM (A single turn model) for a maximum context length of 150. Then we trained the Multi-turn Models (SMN, DAM, IOI, MSN) each, on three different utterance lengths of 10, 30, 50 to check whether there is any improvement in the metrics during evaluation and to check sentence length dependency. The optimizer used was Adam's optimizer with a decay of 0.0001, loss function used was Binary Cross Entropy loss with logits (since our labels are only 0 and 1), a learning rate of 0.001 was used, and every model was trained for 25 epochs, on 1 CPU for a batch size of 20. The size of the original data was 1 million training examples and 500 thousand test examples. But this was taking too long to train on just 1 CPU. So we had to reduce the size of the dataset by 10, reducing the training examples to 100 thousand and testing to 50 thousand. The approximate runtime for SMN was around 6 hours while models like IOI even took 46 hours on just a 100k dataset. The metrics that we used to evaluate were Recall@1, Recall@2, Recall@5, Mean Reciprocal Rank, Mean Average Precision. We used the recall measures because there is just one positive example and 9 negative examples per context in the testing data. MSN is currently the state of the art Multi-turn selection model but we couldnt achieve the same results with our replica of MSN (or any other model on that matter), due to the very stringent time constraint (we could run the model for only 1 set of hyper-parameter values and had no time to change them and see if there is any improvement). As can be seen from the tables we have obtained very poor values for each of these models, with BM-25 outperforming all of them (which is quite unnatural and quite the opposite of what we wanted our outcome to be, considering BM25 which is a non-neural model and was our benchmark model). This is probably because we reduced the size of the dataset and that resulted in underfitting (since deep learning based models are highly dependent on large amounts of data).

What we had planned to do but couldn't execute due to lack of time: We were only able to determine to some extent the dependency of sentence length on the retrieval result. We also wanted to check the dependency of the number of utterances used (in our case it was kept at a constant 10), but we could vary from 2-10 in steps of 3 to check the effect on the retrieved results. We also wanted to carry out individual study for IOI and MSN, where in we could observe the effect of change in the number of interaction blocks of IOI and change in the number of Hopk selectors in MSN on the results.

	R@1	R@2	R@5	MRR	MAP
BM_{25}	0.643	0.732	0.921	0.750	0.750
BiLSTM	0.14	0.22	0.52	0.320	0.320

Table 1: Single Turn Models for maximum context length of 150

	R@1	R@2	R@5	MRR	MAP
SMN	0.14	0.19	0.51	0.314	0.314
DAM	0.15	0.18	0.50	0.290	0.290
IOI	0.18	0.18	0.61	0.349	0.349
MSN	0.03	0.12	0.41	0.229	0.229

Table 2: Maximum Utterances = 10

	R@1	R@2	R@5	MRR	MAP
SMN	0.15	0.21	0.57	0.324	0.324
DAM	0.16	0.22	0.62	0.333	0.333
IOI	0.24	0.25	0.73	0.231	0.231
MSN	0.04	0.13	0.42	0.231	0.231

Table 3: Maximum Utterances = 30

5 Statistical Significance Tests

We performed two statistical significance tests:

1. Wilcoxon Singed-Rank Test:

   This test is used to test whether the distributions coming from two paired samples are the same or not. Here we assume that the samples are independent and identically distributed (i.i.d). It is also assumed that the samples can be paired and be ranked. If the p-value obtained from the test is below the set significance level it means that the samples probably come from a different distribution and are statistically different. We compare the scores of each of the multi-turn model with the other and see how statistically different they are from one other. The p-values obtained were:

	R@1	R@2	R@5	MRR	MAP
SMN	0.17	0.31	0.62	0.368	0.368
DAM	0.15	0.20	0.50	0.299	0.299
IOI	0.13	0.18	0.42	0.288	0.288
MSN	0.10	0.22	0.46	0.297	0.297

Table 4: Maximum Utterances = 50

DAM v/s SMN : 0.302

IOI v/s SMN : 0.562

MSN v/s SMN : 0.00001

DAM v/s IOI : 0.875

MSN v/s DAM : 0.001

IOI v/s MSN : 0.019

We can see that only MSN when compared with all the other models has a statistically significant deviation.

For BM-25 and bi-LSTM the p-value came out to be 0.04.

2. Friedman Test:

   This test has the same characteristics as above but this is used to test whether more than two samples come from different distributions. Here $p = 0.0019$ is less than the significance level of $\alpha = 0.05$. We can imply that the retrieval results from different multi-turn models were significantly different.

6 Conclusion

We initially dived in with full enthusiasm, thinking we could achieve better results for each of these models when compared to their initial implementations. Although we couldn't achieve any state of the art results, we are quite satisfied by out effort, and we also learnt so much by reading so many research papers, finding different ways to implement certain algorithms (like attention) and by constantly waiting anxiously as to when the model would finish training and many more. Even though in the previous check points we mentioned that we would try to build a hybrid model incorporating features from these models, but then we were finding it hard to explain as to why this particular part would result in better performance. We had also thought of using the PMI term-term co-occurrence matrix as one feature in this hybrid model (which would provide to some degree of knowledge distillation), but computing this matrix became very time expensive and so we had to drop this idea. So instead we thought of rigorously evaluating the models that were already made, but time wasn't enough for that as well, even though we were working at our maximum efficiency, with the burden of quizzes, assignments from other courses as well. All in all, this is what we could do in a one month time. We would like to thank Prof: Srikanta Bedathur for giving us this opportunity as well as HPC-IITD for allowing us to use their super-computing resources.

References

Yuan, Chunyuan and Zhou, Wei and Li, Mingming and Lv, Shangwen and Zhu, Fuqing and Han, Jizhong and Hu, Songlin. 2019. *Multi-hop Selector Network for Multi-turn Response Selection in Retrieval-based Chatbots*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) pages 111-120.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu and Hua Wu. 2018. *Multi-Turn Response Selection for Chatbots with Deep Attention Matching Network*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. *One Time of Interaction May Not Be Enough: Go Deep with an Interaction-over-Interaction Network for Response Selection in Dialogues*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics .

Rui Yan, Yiping Song, and Hua Wu. 2016. *Learning to respond with deep neural networks for retrieval based human-computer conversation system.* Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 55–64.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser. 2017. *Attention Is All You Need*. In Advances in neural information processing systems, pages 5998–6008.

Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019. 2019. *Multirepresentation fusion network for multi-turn response selection in retrieval-based chatbots.*. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pages 267–275. ACM.

Debanjan Chaudhuri, Agustinus Kristiadi, Jens Lehmann, and Asja Fischer. 2018. *Improving response selection in multi-turn dialogue systems by incorporating domain knowledge.* In Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 497–507.

Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, Ming Zhou. 2017. *Sequential Matching Network: A New Archtechture for Multi-turn Response Selection in Retrieval-based Chatbots* ACL 2017.