ATTENTIVE HISTORY SELECTION FOR CONVERSATIONAL QUESTION ANSWERING

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, Mohit Iyyer

Presented by - Vedanshi Kataria (20774266)
CONTENTs

➤ Introduction to Conversation Agents
➤ Motivation
➤ Bert Encoder
➤ Proposed Methods
➤ Experiments and Evaluation
➤ Ablation Analysis
➤ Future Work
CONVERSATIONAL AGENTS

➤ Can be of multiple types:

➤ Open Domain: General conversation, Natural Dialogues. Example:

➤ Closed Domain: Task(s) specific conversation, Conversational Search

➤ Early Conversational Agents involved Intent Detection, Slot Filling, Information Retrieval Model, NLU module

➤ Siri and Google Assistant can be looked at as an example of a combination of both these types.
MOTIVATION

➤ Information Retrieval in the form of general conversational Question Answering (ConvQA) requires the system to remember old conversation as well.

➤ Existing systems only use the current question to find an answer from the context provided.

➤ No existing work that focuses on learning to select or re-weight conversational history turns.

➤ There may be three different types of conversation turns:

  ➤ **Drill Down**: the current question is a request for more information about a topic being discussed

  ➤ **Topic Shift**: the current question is not immediately relevant to something previously discussed

  ➤ **Topic Return**: the current question is asking about a topic again after it had previously been shifted away from
BERT ENCODER

- Encodes question $q_k$, paragraph $p$ (context), and conversational histories $H_k$ into contextualised representations.

- Input: $(q_k, p, H_k)$. This input is used to generate $(k - 1)$ variations of the instance where each variation contains the same question and passage, with only one turn of conversation history.

- If the context paragraph is too long, a sliding window is used to split it. Suppose the paragraph is split into $n$ pieces, the training instance $(q_k, p, H_k)$ will generate $n(k - 1)$ input sequences.

- Generates contextualised token-level representations based on the embeddings for tokens, segments, positions, and a special positional history answer embedding (PosHAE)
PROPOSED METHOD 1 – POSITIONAL HISTORY ANSWER EMBEDDINGS

➤ **Intuition** behind adding Positional Embeddings: Utility of a historical utterance could be related to its position.

➤ Previous works have been simply appending “n” previous answers to the question.

➤ **Observed Benefits:** Enables the ConvQA model to capture the spatial patterns of history answers in context.

*Diagram of Encoder with PosHAE*
PROPOSED METHOD 2 – HISTORY ATTENTION MECHANISM

➤ Inputs: Generated token-level and sequence-level representations for all variations

➤ A single layer feed forward network is used to learn the attention weights.

➤ Attention Vector $D \in \mathbb{R}^h$ is learnt to compute attention weight for each sentence presentation $s_k^i$ using $w_i = \frac{e^{D^i s_k}}{\sum_{i'=1}^l e^{D^i s_k^i}}$

➤ *Fine-grained history attention*: Instead of using sequence level representation $S_K$ as input for attention network, use token level representation

![Diagram of the proposed method 2 - History Attention Mechanism](image)
**PROPOSED METHOD 3 – MULTI TASK LEARNING (1)**

- **Answer Span Prediction**: For each token, predict the probability of being BEGIN token as well as END token i.e. learn *begin vector* $B$ and *end vector* $E$.

- The probability for token being *begin token* and *end token* is respectively, where $B$ and $E$ are the learnt vectors and $t_k(m)$ is the token representation for the $m^{th}$ token in the $k^{th}$ sequence.

- Cross Entropy loss is computed for both, $B$ and $E$ as:
  \[
  L_B = - \sum_M \mathbb{1}\{m = m_B\} \log p^B_m \quad , \quad L_E = - \sum_M \mathbb{1}\{m = m_E\} \log p^E_m
  \]

- The final loss is $L_{ans} = \frac{1}{2}(L_B + L_E)$.

- Invalid predictions are discarded at testing time. Examples:
  - predicted span overlaps with the question part of the sequence
  - end token comes before the begin token
PROPOSED METHOD 3 – MULTI TASK LEARNING (2)

➤ Dialog Act Prediction: Two sets of parameters $A \in \mathbb{R}^{|V_a| \times h}$ and $C \in \mathbb{R}^{|V_a| \times h}$ are learnt predict the dialog act of affirmation and confirmation respectively. $|V_a|$ and $|V_c|$ denote number of classes.

➤ Affirmation Classes: Yes, No, Cannot Say

➤ Confirmation Classes: Drill Down, Topic Shift, Topic Return

➤ This is an independent predictor that does not consider conversation history.

➤ We calculate cross entropy loss for both Affirmation and Confirmation as $L_A$ and $L_C$. 
TRAINING

➤ Hyper parameters $\lambda$ and $\mu$ are used combine the losses of both the tasks: $L = \mu L_{\text{ann}} + \lambda L_A + \lambda L_C$

➤ Advantages:

➤ Two tasks provide more supervising signals to fine-tune the encoder.

➤ Representation learning benefits from regularisation effect by optimising for multiple tasks.
COMBINED MODEL REPRESENTATION

End to End System Representation
ATTENTION VISUALIZATION

- Brighter spots mean higher attention weights.
- Token ID refers to the token position in an input sequence. A sequence contains 384 tokens.
- Relative history position refers to the difference of the current turn # with a history turn #. The selected examples are all in the 7th turn.
- Dialog Acts (Confirmation):
  - **Drill Down**: the current question is a request for more information about a topic being discussed
  - **Topic Shift**: the current question is not immediately relevant to something previously discussed
  - **Topic Return**: the current question is asking about a topic again after it had previously been shifted away from
EXPERIMENTATION & EVALUATION

➤ Data: QuAC (Question Answering in Context) dataset
  ➤ Designed for modelling and understanding information-seeking conversations
  ➤ Contains interactive dialogs between an information-seeker and an information provider
  ➤ Information-seeker tries to learn about a hidden Wikipedia passage by asking a sequence of freeform questions
  ➤ Dialog data contains dialog act information
  ➤ Questions are more open-ended, unanswerable, or only meaningful within the dialog context

<table>
<thead>
<tr>
<th>Items</th>
<th>Train</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td># Dialogs</td>
<td>11,567</td>
<td>1,000</td>
</tr>
<tr>
<td># Questions</td>
<td>83,568</td>
<td>7,354</td>
</tr>
<tr>
<td># Average Tokens Per Passage</td>
<td>396.8</td>
<td>440.0</td>
</tr>
<tr>
<td># Average Tokens Per Question</td>
<td>6.5</td>
<td>6.5</td>
</tr>
<tr>
<td># Average Tokens Per Answer</td>
<td>15.1</td>
<td>12.3</td>
</tr>
<tr>
<td># Average Questions Per Dialog</td>
<td>7.2</td>
<td>7.4</td>
</tr>
<tr>
<td># Min/Avg/Med/Max History Turns Per Question</td>
<td>0/3.4/3/11</td>
<td>0/3.5/3/11</td>
</tr>
</tbody>
</table>
EXPERIMENTATION & EVALUATION

➤ Key take-aways:

➤ Bert + PosHAE has better training efficiency and performance than FlowQA

➤ HAM performs better than BERT + PosHAE

➤ Applying BERT-Large to HAM substantially improves answer-span prediction. A more powerful encoder can boost the performance.

<table>
<thead>
<tr>
<th>Models</th>
<th>F1</th>
<th>HEQ-Q</th>
<th>HEQ-D</th>
<th>Yes/No</th>
<th>Follow up</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF++</td>
<td>51.8 / 50.2</td>
<td>45.3 / 43.3</td>
<td>2.0 / 2.2</td>
<td>86.4 / 85.4</td>
<td>59.7 / 59.0</td>
</tr>
<tr>
<td>BiDAF++ w/ 2-C</td>
<td>60.6 / 60.1</td>
<td>55.7 / 54.8</td>
<td>5.3 / 4.0</td>
<td>86.6 / 85.7</td>
<td>61.6 / 61.3</td>
</tr>
<tr>
<td>BERT + HAE</td>
<td>63.9 / 62.4</td>
<td>59.7 / 57.8</td>
<td>5.9 / 5.1</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FlowQA</td>
<td>64.6 / 64.1</td>
<td>– / 59.6</td>
<td>– / 5.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>BERT + PosHAE</td>
<td>64.7 / –</td>
<td>60.7 / –</td>
<td>6.0 / –</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HAM</td>
<td>65.7‡ / 64.4</td>
<td>62.1 / 60.2</td>
<td>7.3 / 6.1</td>
<td>88.3 / 88.4</td>
<td>62.3 / 61.7</td>
</tr>
<tr>
<td>HAM (BERT-Large)</td>
<td>66.7‡ / 65.4</td>
<td>63.3 / 61.8</td>
<td>9.5 / 6.7</td>
<td>88.2 / 88.2</td>
<td>62.4 / 61.0</td>
</tr>
</tbody>
</table>
ABLATION ANALYSIS

➤ Performance Drop:

➤ By replacing fine-grained history attention with sequence-level history attention

➤ By disabling the history attention module, performance drops dramatically for 4.6% and 3.8%

➤ Disabling history attention also hurts the performance for dialog act prediction

➤ Removing the answer span prediction task, a relatively large performance drop for dialog act prediction is observed

➤ Performance Increase:

➤ Removal of the dialog act prediction task results in a slight and insignificant increase in the performance for answer span prediction.

➤ The encoder benefits from a regularisation effect because it is optimised for two different tasks and thus alleviates overfitting.
REFERENCES


THANK YOU