Multi-Turn Response Selection for Chatbots with Deep Attention Matching Network

Xiangyang Zhou*, Lu Li*, Daxiang Dong, Yi Liu, Ying Chen,
Wayne Xin Zhao, Dianhai Yu and Hua Wu

Baidu – Natural Language Processing
Background

- Naturally and consistently converse with human-beings on open-domain topics.

- Data-driven
  - Retrieval-based method
  - Generation-based method
  - System ensemble
Context-Response Matching

Context $c$

Response $r$

$p(is\_good|c,r)$
Retrieval-based Chatbot

Context $c$

Retrieval System

Dialogue Corpus

Search

Response $r_1$
Response $r_2$
Response $r_3$
Response $r_4$

Rank

Response $r_1$
Response $r_4$
Response $r_2$
Response $r_3$

$p(is\_good|c,r)$

An information retrieval approach to short text conversation. Ji et al., 2014
Adversarial Dialogue Generation

Dialogue Corpus

Real Conversation

Sampled \(<c, r>\)

Generator \(G(r^* | c)\)

Discriminator

\[ p(\text{is_good}|c, r) \]
## Challenges

<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Detecting truly matched segment pairs across context and response.</td>
</tr>
<tr>
<td>• Segment pairs could be matched at different granularities.</td>
</tr>
<tr>
<td>• Segment pairs, across context and response, could be matched because of textual relevance or semantic dependency.</td>
</tr>
</tbody>
</table>
Conversation Context

**Speaker A:** Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

**Speaker B:** Try `dpkg - get-selections`

**Speaker A:** What is that like? A database for packages instead of a flat file structure?

**Speaker B:** `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it

**Response of Speaker A:** No clue what do you need it for, its just reassurance as I don’t know the debian package manager
**Conversation Context**

**Speaker A:** Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

**Speaker B:** Try `dpkg -get-selections`

**Speaker A:** What is that like? A database for packages instead of a flat file structure?

**Speaker B:** `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it

**Response of Speaker A:** No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try `dpkg - get-selections`

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system. I don’t see a path, is the list being held somewhere else?

Speaker B: Try dpkg - get-selections

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the debian package manager
## Motivation

### Previous works

Use GRU/LSTM to encode segments and match context with response only considering textual relevance.

### Motivation

- **Self-Attention**: Using intra-attention of utterance/response to gradually construct multi-grained semantic representations.

- **Cross-Attention**: Using attention across context and response to match with dependency information.
Conversation Context

**Speaker A:** Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

**Speaker B:** Try `dpkg - get-selections`

**Speaker A:** What is that like? A database for packages instead of a flat file structure?

**Speaker B:** `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it

**Response of Speaker A:** No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try dpkg - get-selections

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try dpkg - get-selections

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for its just reassurance as I don’t know the debian package manager
Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try dpkg - get-selections

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try `dpkg` - get-selections

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: Try `dpkg - get-selections`

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: `dpkg` is the Debian package manager - `get-selections` simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don’t know the Debian package manager
Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

Speaker B: `dpkg -get-selections`

Speaker A: What is that like? A database for packages instead of a flat file structure?

Speaker B: `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, it’s just reassurance as I don’t know the debian package manager
Conversation Context

**Speaker A:** Hi I am looking to see what packages are installed on my system, I don’t see a path, is the list being held somewhere else?

**Speaker B:** `dpkg -get-selections`

**Speaker A:** What is that like? A database of packages instead of a flat file structure?

**Speaker B:** `dpkg` is the debian package manager - `get-selections` simply shows you what packages are handed by it.

**Response of Speaker A:** No clue what do you need it for, it’s just reassurance as I don’t know the debian package manager.
Attention Module

\[
V^{att} = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_k}})V
\]

\[
Q^{att} = \text{LayerNorm}(V^{att} + Q)
\]

\[
\text{FFN}(Q^{att}) = \max(0, Q^{att}W_1 + b_1)W_2 + b_2
\]

\[
\text{LayerNorm}(\text{FFN}(Q^{att}) + Q^{att})
\]

Summary

- Capture structures across Q and K-V
- Composite semantic representations of captured structures with input embedding
$u_1$

$u_i$

$u_n$

$r$

Word Embedding
Word Embedding Representation Module

\[ \mathbf{u}_1 \]

\[ \mathbf{u}_i \]

\[ \mathbf{u}_n \]

\[ \mathbf{R} \]

\[ \mathbf{R}^0 = \mathbf{r}, \]
\[ \mathbf{R}^l = \text{AttentiveModule}(\mathbf{R}^{l-1}, \mathbf{R}^{l-1}, \mathbf{R}^{l-1}), \]
\[ \mathbf{R}^L = \text{AttentiveModule}(\mathbf{R}^{L-1}, \mathbf{R}^{L-1}, \mathbf{R}^{L-1}) \]

\[ \mathbf{U}_i^0 = \mathbf{u}_i, \]
\[ \mathbf{U}_i^l = \text{AttentiveModule}(\mathbf{U}_i^{l-1}, \mathbf{U}_i^{l-1}, \mathbf{U}_i^{l-1}), \]
\[ \mathbf{U}_i^L = \text{AttentiveModule}(\mathbf{U}_i^{L-1}, \mathbf{U}_i^{L-1}, \mathbf{U}_i^{L-1}) \]

\[ \text{word-level} \]
\[ \text{phrase-level} \]
\[ \text{sent.-level} \]
Word Embedding Representation Module

\[ u_1 \]

\[ u_i \]

\[ u_n \]

\[ r \]

\[ U_i \]

\[ M^{u_i,r}_{self} \]

\[ M^{u_i,r}_{cross} \]

Multi-grained Representations

Word-word Matching with Cross-Attention
Word Embedding Representation Module

\[ \mathbf{u}_1 \]

\[ \mathbf{u}_i \]

\[ \mathbf{u}_n \]

\[ \mathbf{r} \]

Self-Attention Match

\[ \mathbf{M}^{u_i,r,l}_{\text{self}} = \{ \mathbf{U}^l_i[k]^T \cdot \mathbf{R}^l[t] \}_{n_{u_i} \times n_r} \]

Match with textual information

Word-word Matching with Cross-Attention
The figure illustrates a Word Embedding Representation Module with Multi-grained Representations and Word-word Matching with Cross-Attention. The module consists of several layers, including self-attention and cross-attention components.

**Self-Attention Match**

\[ M_{self}^{u_i, r, l} = \{U_i[k]^T \cdot R[l][t]\}_{n_u_i \times n_r} \]

**Match with textual information**

**Cross-Attention Match**

\[ \widetilde{U}_i^l = AttentiveModule(U_i^l, R^l, R^l) \]
\[ \tilde{R}^l = AttentiveModule(R^l, U_i^l, U_i^l) \]
\[ M_{cross}^{u_i, r, l} = \{\widetilde{U}_i^l[k]^T \cdot \tilde{R}^l[t]\}_{n_u_i \times n_r} \]

**Match with dependency information**
\( \mathbf{u}_1 \)

\( \mathbf{u}_i \)

\( \mathbf{u}_n \)

\( \mathbf{r} \)

Word Embedding

Representation Module

Multi-grained Representations

Word-word Matching with Cross-Attention

Matching Score

\( g(c,r) \)

3D Matching Image Q

\( M^{u_i,r}_{self} \)

\( M^{u_i,r}_{cross} \)
Multi-grained matching, including both **textual** and **dependency** information, of a segment pair across context and response.
## Experiment

- **Ubuntu Corpus**
  - One-one multi-turn conversation
  - Ubuntu troubleshooting

- **Douban Corpus**
  - One-one multi-turn conversation
  - Open domain topics

- **Task**
  - Given multi-turn context and several response candidates
  - Select the best candidate based on matching score

<table>
<thead>
<tr>
<th></th>
<th>Ubuntu Corpus V1</th>
<th>Douban Conversation Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Dev</td>
</tr>
<tr>
<td># context-response-pairs</td>
<td>1M</td>
<td>50k</td>
</tr>
<tr>
<td># candidates per context</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td># positive candidates per context</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Min. # turns per context</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Max. # turns per context</td>
<td>19</td>
<td>98</td>
</tr>
<tr>
<td>Avg. # turns per context</td>
<td>7.71</td>
<td>6.69</td>
</tr>
<tr>
<td>Avg. #words per utterance</td>
<td>10.34</td>
<td>18.56</td>
</tr>
</tbody>
</table>
Experiment

• DAM setup
  • Test stacking 3-7 self-attention layers

\[ L(.) = - \sum_{(c,r,y) \in D} g(c,r)y + (1 - g(c,r))(1 - y) \]

• Comparison
  • **Sequential Matching Network (SMN)** (Wu et al., ACL-2017), **Multi-view Matching** (Zhou et al., EMNLP-2016), **DL2R** (Yan et al., SIGIR-2016), **DualEncoder** (Rowe et al., SigDial-2015)

• Ablation
  • \( DAM_{first} \) : without stacked self-attention
  • \( DAM_{last} \) : only using the last layer of stacked self-attention
  • \( DAM_{self} \) : only using self-attention-match
  • \( DAM_{cross} \) : only using cross-attention-match
<table>
<thead>
<tr>
<th>Model</th>
<th>Ubuntu Corpus</th>
<th>Douban Conversation Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_2@1$</td>
<td>$R_{10}@1$</td>
</tr>
<tr>
<td>DualEncoder_{lstm}</td>
<td>0.901</td>
<td>0.638</td>
</tr>
<tr>
<td>DualEncoder_{bilstm}</td>
<td>0.895</td>
<td>0.630</td>
</tr>
<tr>
<td>MV-LSTM</td>
<td>0.906</td>
<td>0.653</td>
</tr>
<tr>
<td>Match-LSTM</td>
<td>0.904</td>
<td>0.653</td>
</tr>
<tr>
<td>Multiview</td>
<td>0.908</td>
<td>0.662</td>
</tr>
<tr>
<td>DL2R</td>
<td>0.899</td>
<td>0.626</td>
</tr>
<tr>
<td>SMN_{dynamic}</td>
<td>0.926</td>
<td>0.726</td>
</tr>
<tr>
<td>DAM</td>
<td>0.938</td>
<td>0.767</td>
</tr>
<tr>
<td>DAM_{first}</td>
<td>0.927</td>
<td>0.736</td>
</tr>
<tr>
<td>DAM_{last}</td>
<td>0.932</td>
<td>0.752</td>
</tr>
<tr>
<td>DAM_{self}</td>
<td>0.931</td>
<td>0.741</td>
</tr>
<tr>
<td>DAM_{cross}</td>
<td>0.932</td>
<td>0.749</td>
</tr>
</tbody>
</table>
Self-Attention Match Visualization

Stack-0 ➔ Stack-2 ➔ Stack-4
Cross-Attention Match Visualization

Self-Attention Match

Cross-Attention Match
Summary

• We propose a novel deep attention matching network for multi-turn response selection that entirely based on attention.

• We use stacked self-attention to construct multi-grained semantic representations.

• We use cross-attention to match context with its candidate response considering both textual and dependency information.
Thanks

• Code & Data: https://github.com/baidu/ Dialogue/DAM

• Our Authors