Applications of Deep Learning in Spoken Dialogue Systems

Steve Young

Engineering Department
Trumpington Street
Cambridge, UK
sjy@eng.cam.ac.uk

Apple Europe Limited
90 Hills Road
Cambridge, UK
steve_young@apple.com
Dialog System Architecture

User → Recognition Hypotheses → ASR → Semantic Decoder → Belief Tracker → Understanding → Dialogue Level → Dialog Policy → Dialog Manager → Database/Application → System Actions

System Response

User

TTS → Message Generator → Response Planner → Generation → System Response

Belief State
Understanding: ASR $\rightarrow$ Beliefs

Repeated for Each Slot $s$

ASR Hyp#1 $[p_1]$

ASR Hyp#1 $[p_2]$

Last System Act

Per Turn Semantic Decoding

Per Utterance Belief Tracking

$P_s(v)$

Belief State = Concatenation of Slot Probability Vectors

Generation: actions -> words

Need to convert abstract system actions to natural language e.g.

\[
\text{inform(name=“The Peking”, food=“chinese”)} \quad \rightarrow \quad \text{“The Peking serves chinese food”}
\]

Solution: delexicalise the training data, and train a conditional LSTM

\[
\text{inform(name=\langle name \rangle, food=\langle food \rangle)} \quad \rightarrow \quad \text{“\langle name \rangle serves \langle food \rangle food”}
\]

Train to predict output sequences word by word conditioned on system action

\[
\text{“\langle s \rangle \langle name \rangle serves”} \quad \rightarrow \quad \text{“\langle food \rangle”}
\]

Semantic Control

“\langle name \rangle serves \langle food \rangle food” inform(\langle name \rangle, \langle food \rangle)
Generation: actions -> words

Need to convert abstract system actions to natural language e.g.

\[
\text{inform(name="The Peking", food="chinese")} \quad \rightarrow \quad \text{"The Peking serves chinese food"}
\]

Solution: delexicalise the training data, and train a conditional LSTM

\[
\text{inform(name=<name>, food=<food>)} \quad \rightarrow \quad \text{"<name> serves <food> food"}
\]

At runtime, condition with system action and prime with start symbol …

… then re-lexicalise.

Dialog Manager

1. Belief state $\mathbf{b}$ encodes the state of the dialog, including all relevant history.

2. Belief state is updated every turn of the dialog.

3. The policy $\pi$ determines the best action to make at each turn via a mapping from the belief state $\mathbf{b}$ to actions $\mathbf{a}$.

4. Every dialog ends with a reward: +ve for success, -ve for failure. Plus a weak -ve reward for every turn to encourage brevity.

5. Reinforcement Learning is used to find the best policy.
Reinforcement Learning

Policy: \( \pi(b, a) : \mathbb{R}^n \times A \rightarrow [0,1] \)

Reward: \( R = \sum_{\tau=1}^{T} r(b_{\tau}, a_{\tau}) \)

Value: \( Q_\pi(b_t, a_t) = \mathbb{E}_\pi \left[ \sum_{\tau=t+1}^{T} r(b_{\tau}, a_{\tau}) \right] \)

Problem:

Find optimal policy \( \pi^* = \arg\max_{\pi} \left\{ E[R | \pi] \right\} \)

or

Solve \( Q^*_\pi(b_t, a_t) = r_{t+1} + \max_{a} \left\{ Q^*_\pi(b_{t+1}, a) \right\} \)
Implementations Algorithms

**Policy Gradient**
- Advantage Actor Critic (A2C)
- Trust Region Actor Critic (TRACER)
- Natural Actor Critic (eNACER)

**Value Iteration**
- Deep Q Network (DQN)
- [ Gaussian Process (GP) ]

See also David Silver’s 2016 ICML Tutorial.
The Labelling Problem

So can we train End-To-End using just \( \langle u_t, m_t, r_t \rangle \)?
Sequence to Sequence models

**LSTM Encoder**

**LSTM Decoder**

Supervised Learning (no reward):
maximise logP of correct response \( m \) given input \( u \) for every input/output pair in training set.

\[
L(\theta) = \sum_{<u_i, m_i>} \log P(m_i | u_i)
\]

Good for chatbots, but no explicit knowledge base and no planning

The action $a_t$ is now a discrete latent variable

$$p(m_t | u_t) = \sum_a g(m_t | a) \pi(a | u_t)$$

Unfortunately, there is no tractable way to compute this inference and Monte Carlo methods are too slow.
Neural Variational Inference (NVI)

\[ u_t \xrightarrow{\text{SLU}} b_t \xrightarrow{\pi_{\theta_1}(a_t \mid u_t)} a_t \xrightarrow{\text{Policy}} a_t^{(n)} \xrightarrow{g_{\theta_2}(m_t \mid a_t)} m_t \xrightarrow{\text{NLG}} \]

\[ q_{\phi}(a_t \mid u_t, m_t) \]

Maximise the variational lower bound

\[ L(m, u, \theta_1, \theta_2, \phi) = \mathbb{E}_q[\log g(m_t \mid a_t)] - \lambda D_{KL}(q(a_t) \mid \pi(a_t \mid .)) \]

NVI Optimisation

1) Randomly sample a minibatch of training data
   \[ \mathcal{D} = \langle u_1, m_1 \rangle \ldots \langle u_N, m_N \rangle \]

2) For each \( \langle u_i, m_i \rangle \), generate N samples from inference net
   \[ a_i^{(1)} \ldots a_i^{(N)} \sim q_\phi(a \mid u_i, m_i) \]

3) Compute gradients \( \nabla_\theta L \) and \( \nabla_\phi L \) using Monte Carlo integration to estimate expectations.

4) Update parameters

Note that \( \nabla_\phi L \) is very noisy and variance reduction techniques are required in practice.
"What area would you like?"

"Somewhere in the centre."

$s_t = u_t \oplus b_t \oplus x_t$

$a_t^{(n)} \sim \pi_\Theta(a_t | s_t)$

Sample an action from policy

"Try Pizza Express in the centre of town."

Training the LIDM

In practice, learning latent actions in a completely unsupervised manner is extremely difficult. Hence, a multi-stage approach was taken for training the full end-to-end dialogue system:

- the SLU Component was pre-trained on labelled data
- Part of the training corpus was clustered to provide a subset of automatically labelled actions
- Variational lower bound maximisation was interleaved with supervised learning on the automatically labelled data
- Reinforcement learning was used to fine tune policy parameters

Sample Dialogue

<table>
<thead>
<tr>
<th>Human:</th>
<th>Hi, I’m hungry for some Indonesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine:</td>
<td>(0 0.71) <strong>i am sorry, there are no indonesian restaurants in the area. would you like something else?</strong> (46 0.19) there are no restaurants serving indonesian kind of food. (34 0.07) what location would you like?</td>
</tr>
<tr>
<td>Human:</td>
<td>Oh no. How about Indian?</td>
</tr>
<tr>
<td>Machine:</td>
<td>(0 0.59) there are 9 restaurants in the east side of town. would you like their address? (30 0.15) curry prince serves indian cuisine. (34 0.10) <strong>what area would you like?</strong></td>
</tr>
<tr>
<td>Human:</td>
<td>East side please</td>
</tr>
<tr>
<td>Machine:</td>
<td>(0 0.62) <strong>curry prince is a indian restaurant in the east side of town. would you like their telephone number?</strong> (84 0.19) curry prince is an indian restaurant in the east of town and the number is 01223 566388. (81 0.08) curry prince serves indian type of cuisine and is located in the east part of town.</td>
</tr>
<tr>
<td>Human:</td>
<td>yes, the address and phone number please</td>
</tr>
<tr>
<td>Machine:</td>
<td>(3 0.46) the address is 451 newmarket road fen ditton and the phone is 01223 566388. (11 0.16) <strong>the address for curry prince is 451 newmarket road fen ditton and the phone number is 01223 566388.</strong> (0 0.15) curry prince is located at 451 newmarket road fen ditton and their phone is 01223 566388. is there anything else i can help you with?</td>
</tr>
<tr>
<td>Human:</td>
<td>Okay, thank you goodbye</td>
</tr>
<tr>
<td>Machine:</td>
<td>(0 0.57) thank you, goodbye. (2 0.13) goodbye. (1 0.11) <strong>you are welcome. goodbye.</strong> (6 0.10) thank you for using the cambridge restaurant system. goodbye.</td>
</tr>
</tbody>
</table>

actual outputs selected in dialogue shown in bold
Summary

- DNNs provide a flexible building block for all stages of the dialogue system pipeline, though training is rarely as straightforward as research papers suggest!
- Labelled data is expensive and each stage of a multi-component pipeline requires its own labelled data set.
- “End-2-End” multi-component training has potential to reduce labelled data requirement and potentially avoid hand-crafting internal interfaces.
- Users can provide feedback for free but the feedback signal is weak and noisy. Reinforcement Learning provides a framework for exploiting this mostly untapped resource.
Credits

All members of the Cambridge Dialogue Systems Group Past and Present:

<table>
<thead>
<tr>
<th>Milica Gasic</th>
<th>Matt Stuttle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catherine Breslin</td>
<td>Martin Szummer</td>
</tr>
<tr>
<td>Pawel Budzianowski</td>
<td>Eddy Su</td>
</tr>
<tr>
<td>Matt Henderson</td>
<td>Blaise Thomson</td>
</tr>
<tr>
<td>Filip Jurcicek</td>
<td>Pirros Tsiakoulis</td>
</tr>
<tr>
<td>Simon Keizer</td>
<td>Stefan Ultes</td>
</tr>
<tr>
<td>Dongho Kim</td>
<td>David Vandyke</td>
</tr>
<tr>
<td>Fabrice Lefevre</td>
<td>Karl Weilhammer</td>
</tr>
<tr>
<td>Francois Mairesse</td>
<td>Shawn Wen</td>
</tr>
<tr>
<td>Nikola Mrksic</td>
<td>Jason Williams</td>
</tr>
<tr>
<td>Lina Rojas Barahona</td>
<td>Hui Ye</td>
</tr>
<tr>
<td>Jost Schatzmann</td>
<td>Kai Yu</td>
</tr>
</tbody>
</table>