Continual Learning Dialogue Systems
– Learning after Model Deployment

Bing Liu
Department of Computer Science
University of Illinois at Chicago
Introduction

- Classic machine learning: Isolated single-task learning

- Key weaknesses
  - Closed-world assumption: nothing new or unexpected in application
  - No knowledge accumulation or transfer: isolated learning
  - Model is fixed after deployment: no learning or adaptation

- Focus of this talk: Learning after model deployment
  Learning on the job or during model application.
Chatbots should learn continually after deployment
(Chen & Liu, 2018, Liu, 2020)

- **Chatbot**: human users may say things a chatbot does not understand.
  - It must learn new knowledge and new language expressions during chatting.
    - E.g., asking the current or other users.
  - Humans learn a great deal in our daily conversations

- Chatbots **should not** solely rely on offline training initiated by engineers.
Outline

- Continual learning with learning after deployment
- Continual learning to ground new language expressions
- Continual learning of factual knowledge
- Dealing with wrong knowledge from users
- Summary
Classic definition of lifelong/continual learning

- Learn a sequence of tasks, $T_1$, $T_2$, ..., $T_N$, ... incrementally. Each task $t$ has a training dataset $D_t = \{x_{t,i}, y_{t,i}\}_{i=1}^{n_t}$.

- **Goal:** learn each new task $T_{N+1}$ incrementally
  1. **with no catastrophic forgetting:** Learning of the new task $T_{N+1}$ should not result in degradation of accuracy for previous $N$ tasks.
  2. **with knowledge transfer:** leveraging the knowledge learned from previous $N$ tasks to learn the new task $T_{N+1}$ better.

- **Assumption:** Both the task $T_{N+1}$ and its training data $D_{N+1}$ are given by the user.
Continual learning with learning after deployment

Orange lines:
Learning after model deployment
- Learning on the job
Characteristics of continual learning
(Chen and Liu, 2018, Liu, 2020)

- **Continuous incremental learning process** (no forgetting)
- **Knowledge accumulation in KB** (long-term memory)
- **Knowledge transfer/adaptation** (across tasks) (Ke, Liu, Huang, 2020)

**Learning after deployment** *(on the job).* *Self-supervision* using the *accumulated knowledge* and *interaction* with *humans* & *environment.*

**Main steps:**
- Identify new tasks to learn (tasks not given)
- Acquire ground-truth training data (training data not given)
- Learn the tasks incrementally (one-shot or few-shot)
Learning on the job (while working)
(Liu, 2020, Chen and Liu, 2018)

- It is estimated that about 70% of our human knowledge comes from ‘on-the-job’ learning.
  - Only about 10% through formal training
  - The rest 20% through observation of others

- An AI agent should learn on the job too as
  - The world is too complex and constantly changing.
    - Have to learn continually and adapt
  - Without this capability, an AI agent is not truly intelligent.

Example 1 – a chatbot system
(Liu and Mazumder, 2021)

- **Session 1**
  - **User-1**: Hey, I visited Stockholm last week. The place is awesome!
  - **Chatbot**: Where is **Stockholm**?
  - **User-1**: Stockholm is the capital of Sweden.

- **Session 2**
  - **User-2**: I am planning a tour to Europe next month.
  - **Chatbot**: Are you visiting **Stockholm**? I heard it is an awesome place.
Example 2 - a greeting bot in a hotel  
(Chen and Liu 2018)

- See an existing guest.
  - Bot: “Hello John, how are you today?”

- See a new guest. Bot recognizes the guest is new. (create a new task)
  - Bot: “Welcome to our hotel! What is your name, sir?” (get class label)
  - Guest: “David” (got class label: David)
  - Bot learns to recognize David automatically (get training data)
    - take pictures of David (learn incrementally)
    - learn to recognize David

- See David next time.
  - Bot: “Hello David, how are you today?” (use the new knowledge)
Learning new knowledge during dialogue

- **Learning to ground language expressions** (Mazumder et al, 2020b)
  - Learning via multi-turn dialogues with the user
  - Learning via user demonstrations

- **Learning factual knowledge** (Mazumder et al, 2019, 2020a)
  - Extracting new knowledge from user utterances via dialogues (Liu and Mei, 2020)
  - Asking and inferring new facts when the bot cannot answer a user query.
  - Asking questions to learn about unknown entities and concepts (Ono et al. 2017).

- **Learning conversation skills** (Hancock et al. 2019; Shuster et al. 2020)
  - Learning user behaviors and preferences
  - Modeling situation-aware conversations
  - Learning the user’s emotion and mood state

Outline

- Continual learning with learning after deployment
- Continual learning to ground new language expressions
- Continual learning of factual knowledge
- Dealing with wrong knowledge from users
- Summary
Learning to ground NL commands

- **Task-oriented chatbots** like virtual assistants (e.g., Siri, Alexa, etc) are **Natural Language (command) Interfaces (NLI)**
  - allow users to issue natural language (NL) commands to be mapped to some actions for execution by the underlying application.

- Existing approaches to building such chatbots:
  - Train an end-to-end deep learning model.
  - Semantic parsing -> logical forms -> translated to executable actions

- We discuss CML (Command Matching and Learning)
Natural language to natural language matching

- An **application-independent approach** to building task-oriented chatbots with interactive continual learning.
  - Based on *natural language to natural language* matching (*NL2NL*)
  - CML to automatically build NLIs for any API-driven applications.

- To build a new NLI (or add a new skill to an existing NLI),
  - the application developer only needs to **write a set \( S_i \) of seed commands (SCs) in NL to represent each action \( i \).**
    - SCs in \( S_i \) are just like paraphrased NL commands from users to invoke \( i \), but the objects to be acted upon in each SC are replaced with variables, the arguments of \( i \).
  - An interactive learning mechanism to enable CML to **continually learn new (paraphrased) SCs from users.**
An example

- **Microsoft Paint tool**: The API action
  `drawCircle(X1, X2)`
  - drawing a circle having color `X1` at coordinate `X2`.
- Let a SC be “draw a X1 circle at X2” for this API,
  - where `X1` and `X2` are variables representing the arguments of the API.
- **User command**: “draw a blue circle at (20, 40)”
  - It can be matched or grounded to this SC, where the grounded API arguments are `X1 = ‘blue’` and `X2 = (20, 40)`.
CML has three components

- **SC (seed command) specification**
  - to enable the application developer to specify a set of SCs for each of their APIs

- **Command grounding module**
  - ground a user command $C$ to an action SC by matching $C$ with the correct SC (whose associated action API is then executed)

- **Interactive learner**
  - It interacts with end-users to learn new SCs and paraphrases of API argument values.
### SC Specification (blocks-world)

Table 2: Action SC specifications for Blocks-World application and some example NL commands from user for each API. (*) denotes that the variable do not take part in command reduction (Utility Constraints), which is automatically detected and marked by CML (see Sec 3.2) (X denotes input).

<table>
<thead>
<tr>
<th>Action API Function</th>
<th>AID</th>
<th>Action SCs (’;’ separated)</th>
<th>Variable: Argument Type</th>
<th>Example commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddBlock (X1)</td>
<td>1</td>
<td>add a block at X1; insert a block at X1</td>
<td>X1: ‘location’ (*)</td>
<td>add a block at (2, 3); put a block at (2, 3)</td>
</tr>
<tr>
<td>Remove (X1)</td>
<td>2</td>
<td>remove X1</td>
<td>X1: ‘block_set’</td>
<td>delete blue block; take away blue block</td>
</tr>
<tr>
<td>Move (X1, X2)</td>
<td>3</td>
<td>move X1 to X2; shift X1 to X2</td>
<td>X1: ‘block_set’, X2: ‘location’ (*)</td>
<td>move blue block to the left of cube; shift green cube to (4, 5)</td>
</tr>
<tr>
<td>MoveByUnits (X1, X2, X3)</td>
<td>4</td>
<td>move X1 along X2 by X3 units</td>
<td>X1: ‘block_set’, X2: ‘direction’, X3: ‘number’</td>
<td>move blue block left by 2 units; shift green cube down by 3 units</td>
</tr>
<tr>
<td>UpdateColor (X1, X2)</td>
<td>5</td>
<td>change color of X1 to X2; color X1 with X2</td>
<td>X1:‘block_set’, X2: ‘color’ (*)</td>
<td>color A red; change color of B to blue</td>
</tr>
<tr>
<td>UpdateShape (X1, X2)</td>
<td>6</td>
<td>change shape of X1 to X2</td>
<td>X1:‘block_set’, X2: ‘shape’ (*)</td>
<td>set the shape of A to cube; make B square</td>
</tr>
<tr>
<td>Rename (X1, X2)</td>
<td>7</td>
<td>rename block X1 to X2</td>
<td>X1: ‘block_set’, X2: ‘name’ (*)</td>
<td>Name the block at (4, 5) as C; rename A to D</td>
</tr>
</tbody>
</table>
Command grounding module (CGM)

- **Rephraser and Tagger (R):**
  - Given the user command $C$, $R$ rephrases $C$ and tags each word or phrase in the rephrased $C$ with either ‘O’ (i.e., not an argument type) or one of the possible argument types of the action SCs.

- **SC Matcher (M):**
  - Given the rephrased and tagged command $C$ and the set $T$ of (action or utility) SCs, Matcher $M$ computes a match score $f(t, C)$ for each $t$ in $T$ and returns the top ranked SC.
  - This work uses an information retrieval (IR) based unsupervised matching model for $M$. 
Figure 1: Working of CGM on a user command for Blocks-World. AID denotes the API IDs (see Tables 2 and 3).
Continual interactive learning

- If CML does not understand a user commend C.
- CML learns a new SC from the user commend C through interactive dialogue.
- It also learn new paraphrased argument values from in C to improve repheaser $R$ over time.

\[
\text{Algorithm 2 Interactive Knowledge Learning}
\]

**Input:** $C'$: Reduced user command by Algorithm 1; $T$: action and utility SC Store; $Q$: Question Template Store; $M$: SC Matcher;

1. $r_1 \leftarrow \text{Verify Pred SC}(Q, C')$ \{r$_i$ is user’s response\}
2. if $r_1 = \text{“no”}$ then
3. \quad $r_2 \leftarrow \text{ShowSC List}(C_{rnk})$ \{$C_{rnk}$ is the action SC rank list returned by $M$\}
4. end if
5. for all variable $x_i$ in $r_2$ do
6. \quad $r_{expr} \leftarrow \text{Ask Ref Expr}(x_i, C)$
7. \quad $r_{prop} \leftarrow \text{Ask Ref Prop}(r_{expr})$
8. \quad $r_{val} \leftarrow \text{Choose Prop Val}(r_{prop}, r_{expr})$
9. \quad $r_{para} \leftarrow \text{Ask Para Expr}(r_{val}, r_{expr})$
10. \quad Update $R$ with all $(r_{val}, r_{para})$ pairs
11. end for
12. Rephrase $C'$ to get a new SC and update $T$
BERT-JISF: joint intent detection and slot filling - fine-tunes a pre-trained BERT model to solve NLU (Chen et al, 2019).

- **A-acc**: action intent prediction
- **Arg-F1**: argument F1

### Datasets
- **BW**: blocks-world
- **WPD**: Webpage design
- **FB**: flight booking

### Table 5: Performance comparison of CML variants and BERT-JISF. Here, CML-vsm(-U) and CML-vsm results are the same for FB as utility APIs are absent in FB specifications.**

<table>
<thead>
<tr>
<th>Models</th>
<th>BW</th>
<th>WPD</th>
<th>FB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A-acc</td>
<td>Arg-F1</td>
<td>A-acc</td>
</tr>
<tr>
<td>BERT-JISF</td>
<td>49.70</td>
<td>59.23</td>
<td>57.87</td>
</tr>
<tr>
<td>CML-jac</td>
<td>68.93</td>
<td>79.35</td>
<td>74.04</td>
</tr>
<tr>
<td>CML-vsm</td>
<td>68.93</td>
<td>79.35</td>
<td>74.46</td>
</tr>
<tr>
<td>CML-embed</td>
<td>68.63</td>
<td><strong>79.94</strong></td>
<td>68.93</td>
</tr>
<tr>
<td>CML-vsm (-R)</td>
<td>64.79</td>
<td>77.42</td>
<td>68.08</td>
</tr>
<tr>
<td>CML-vsm (-U)</td>
<td>14.49</td>
<td>14.49</td>
<td>11.48</td>
</tr>
<tr>
<td>CML-jac + SCL</td>
<td>69.82</td>
<td><strong>81.34</strong></td>
<td>76.17</td>
</tr>
<tr>
<td>CML-vsm + SCL</td>
<td><strong>70.11</strong></td>
<td>80.09</td>
<td><strong>77.02</strong></td>
</tr>
<tr>
<td>CML-jac + SCL + APL</td>
<td>72.78</td>
<td><strong>81.43</strong></td>
<td>80.0</td>
</tr>
<tr>
<td>CML-vsm + SCL + APL</td>
<td><strong>73.07</strong></td>
<td>80.16</td>
<td><strong>80.85</strong></td>
</tr>
</tbody>
</table>
Outline

- Continual learning with learning after deployment
- Continual learning to ground new language expressions
- Continual learning of factual knowledge
- Dealing with wrong knowledge from users
- Summary
Dialogue systems are increasingly using knowledge bases (KBs) storing factual knowledge to help generate responses.

- KBs are inherently incomplete and remain fixed,
- which limit dialogue systems’ conversation capability

**CILK**: *Continuous and Interactive Learning of Knowledge* for dialogue systems

- to continuously and interactively learn and infer new knowledge during conversations
Knowledge learning in conversation

Humans Learn and Leverage Knowledge in Lifelong Manner!

Hey, I visited Stockholm last week. The place is awesome!

Hey, I am planning for a Europe tour soon

Where is Stockholm?

Are you visiting Stockholm? I heard the place has lot of attractions

Stockholm is the capital of Sweden

Knowledge learning happens in a multi-user environment
Opportunities to learn in conversations

1. Extracting knowledge directly from user utterances. E.g.,
   - User: Obama was born in Hawaii.
   - Agent extracts: (Obama, BornIn, Hawaii) – expressed in triples \((h, r, t)\)

2. Asking user questions & expecting correct answers, e.g.,
   - Agent: Where was Obama born?
   - User: Hawaii \(\Rightarrow\) (Obama, BornIn, Hawaii)

3. When the agent cannot answer user questions, it asks the user for some supporting facts and then infers the answers.
   - We focus on this setting (which covers 1 and 2)
Two types of queries or questions

- **Wh-question**
  - E.g., Where was Obama born?
  - (Obama, bornIn, s?)

- **Fact verification question**
  - Was Obama born in Hawaii?
  - (Obama, bornIn? Hawaii)
Components for knowledge learning

Knowledge Base $\mathcal{K}$

Stores acquired Facts (Triples)

KB: Collection of Triples

$\mathcal{T} = \{ (h, r, t) \mid h, t \in E, r \in R \}$

Interaction Module $\mathcal{I}$

Interacts with user to acquire Facts

- decides whether to ask or not, and formulates questions to ask the user for supporting facts

Inference Module $\mathcal{M}$

Infers new Knowledge to answer user’s query

- Performs inference over the acquired Facts and existing KB


Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019
Assumptions

- Focus on developing the core interactive knowledge learning framework
  - Do not build all peripheral components (like fact or relation extraction, entity linking, etc.) which are assumed to be available for use.

- We also assume that the user has good intentions
  - User answers questions with 100% conformity about the veracity of his/her facts (more discussion later)

- User is NOT omniscient
  - We do not assume that the user can answer all questions as opposed to the teacher-student setup - the teacher is assumed to know everything.
(1) When the user asks a Wh-question

- Given a user query / question \((h, r, ?)\) [or \((?, r, t)\)], our goal is two-fold:
  1. **Answering** the user query or **rejecting** the query to remain unanswered if the correct answer is believed to not exist in the KB
  2. **learning / acquiring** some knowledge (supporting facts) from the user to help the answering task.

- We further distinguish two types of queries:
  1. **Closed-world Queries**: \(h\) (or \(t\)) and \(r\) are known to the KB
  2. **Open-world Queries**: Either one or both \(h\) (or \(t\)) and \(r\) are unknown

\[\downarrow\] Proposed Soln.

an engine for **Continuous and Interactive Learning of Knowledge (CILK)**

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019
Interactive knowledge learning in dialogue: example
(Mazumder et al. 2019)

**USER:** In what Country is Boston located?

**CILK:** I do not know what “located in Country” means? Can you provide me an example?

**USER:** (Boston, LocatedInCountry, ?) “In what Country is Boston located?”

**CILK:** Got it. Can you tell me a fact about “Boston”?

**USER:** (Harvard University, UniversityLocatedIn, Boston). “Harvard university is located in Boston.”

**CILK:** (Boston, LocatedInCountry, USA) “Boston is located in USA.”
(2) When the user asks a fact-verification question

- Learn new knowledge interactively from user when the system is unable to answer a user’s factual verification (yes/no) question.

**USER:** (Obama, CitizenOf?, USA) “Is Obama a citizen of USA?”  
**IKAI:** I do not know what “CitizenOf” means? Can you provide me an example?  

**USER:** (David Cameron, CitizenOf, UK). “David Cameron is a citizen of UK.”  
**IKAI:** Got it. Can you tell me how “Obama” and “Honolulu” are related?  
**USER:** (Obama, BornIn, Honolulu). “Obama was born in Honolulu.”  
**IKAI:** Got it. Can you tell also me how “Honolulu” and “Hawaii” are related?  

**USER:** (Honolulu, CapitalOfState, Hawaii). “Honolulu is the state capital of Hawaii.”  
**IKAI:** (Obama, CitizenOf?, USA) holds. “Yes, Obama is a US citizen.”

---

The IKAI approach

- **User question (h, r? s)**

- **Step 1 - Interactive acquisition of supporting facts:**
  - If r is unknown, IKAI asks the user to provide a clue [an example triple r]
  - If s or t is unknown, IKAI asks the user to provide a link/relation to connect the unknown entity s or t with an automatically selected existing entity

- **Step 2 - Knowledge inference (Infer the query answer):**
  - Uses a path-ranking algorithm C-PR (Mazumder and Liu 2017) to build a predictive model (Predictor) to predict whether (s, r?, t) is true.
    - Enumerate relation paths between two entities (s, t) in a KB (encoded as a multi-relation graph) and use those paths as features to train the predictor.
Finite State Machine : Interaction Module (\(I\))

\[(S, A, S_0, S_F, \Delta)\]

- **States**
- **Actions**
- **Initial states**
- **Final states**
- **\(\Delta : (S, A) \rightarrow S\)** Transition Function

<table>
<thead>
<tr>
<th>SB</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QERS</td>
<td>Query entities and relation searched</td>
<td>Whether the query source (s) and target (t) entities and query relation (r) have been searched in KB.</td>
</tr>
<tr>
<td>SEF</td>
<td>Source Entity Found</td>
<td>Whether the source entity (s) has been found in KB.</td>
</tr>
<tr>
<td>TEF</td>
<td>Target Entity Found</td>
<td>Whether the target entity (t) has been found in KB.</td>
</tr>
<tr>
<td>QRF</td>
<td>Query Relation Found</td>
<td>Whether the query relation (r) has been found in KB.</td>
</tr>
<tr>
<td>CLUE</td>
<td>Clue bit set</td>
<td>Whether the triple (to be processed) is a clue from user.</td>
</tr>
<tr>
<td>ILO</td>
<td>Interaction Limit Over</td>
<td>Whether the interaction limit is over for the query.</td>
</tr>
<tr>
<td>PFE</td>
<td>Path Feature Extracted</td>
<td>Whether path feature extraction has been done.</td>
</tr>
<tr>
<td>CPF</td>
<td>Complete Path Found</td>
<td>Whether the extracted path features are complete.</td>
</tr>
<tr>
<td>INFI</td>
<td>Inference Invoked</td>
<td>Whether inference module has been invoked.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Transition Conditions (for current state bits (S_i))</th>
<th>Action Id : Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(QERS = 0)</td>
<td>(a_0) : Search source (s), target (t) entities and query relation (r) in KB.</td>
</tr>
<tr>
<td>(ILO = 0 \land CLUE = 0 \land QERS = 1 \land QRF = 0)</td>
<td>(a_1) : Ask user to provide an clue/example for query relation (r).</td>
</tr>
<tr>
<td>(PFE = 1 \land ILO = 0 \land CPF = 0)</td>
<td>(a_2) : Ask user to provide a missing link for path feature completion.</td>
</tr>
<tr>
<td>(QERS = 1 \land (SEF = 0 \lor TEF = 0) \land ILO = 0)</td>
<td>(a_3) : Ask user to provide a connecting link to add a new entity to the KB.</td>
</tr>
<tr>
<td>(QERS = 1 \land PFE = 0 \land SEF = 1 \land TFE = 1)</td>
<td>(a_4) : Extract path features between source (s) and target (t) entities using C-PF.</td>
</tr>
<tr>
<td>(QRF = 1 \land CPF = 1)</td>
<td>(a_5) : Invoke prediction model for inference.</td>
</tr>
</tbody>
</table>

Working of IKAI’ FSM in a given Session

OKBC query: \( q = (\text{Obama, CitizenOf, USA}) \)

---


NeuCAIR @ ICLR-2021, May 7, 2021

---

**USER:** (Obama, CitizenOf, USA) “Is Obama a citizen of USA?”

**IKAI:** I do not know what “CitizenOf” means. Can you provide me an example?

**USER:** (David Cameron, CitizenOf, UK). “David Cameron is a citizen of UK.”

**IKAI:** Got it. Can you tell me how “Obama” and “Honolulu” are related?

**USER:** (Obama, BornIn, Honolulu). “Obama was born in Honolulu.”

**IKAI:** Got it. Can you tell also me how “Honolulu” and “Hawaii” are related?

**USER:** (Honolulu, CapitalOf, Hawaii). “Honolulu is the state capital of Hawaii.”

**IKAI:** (Obama, CitizenOf, USA) holds. “Yes, Obama is a US citizen.”
Many existing chatbots are written mainly with rules.

We learn with *knowledge distillation pattern*: \((p, F, B)\),

- where \(p\): a dialogue pattern; \(F\): implied facts; \(B\): implied beliefs.
- E.g., \(p\): \(*\) stayed in X at Y \( F\): \{(X, is-a, hotel)\} \( B\): \{(X, has-address, Y)\}

If user says “*I stayed in Hilton at 150 Pine Street,*” the agent gets
- a fact: (Hilton, is-a, hotel) & a belief: (Hilton, has-address, 150 Pine Street).

If user says “*I stayed in Hilton,*” the agent gets
- A fact: (Hilton, is-a, hotel) and also
- a question to ask the user: “What is the address of the hotel?”

With such patterns, a great deal of knowledge can be learned from end-users.
Outline

- Continual learning with learning after deployment
- Continual learning to ground new language expressions
- Continual learning of factual knowledge
- Dealing with wrong knowledge from users
- Summary
Dealing with wrong knowledge from users

- Learn from end-users comes with a major challenge:
  - Acquiring intentional or unintentional wrong knowledge from users.

- Since chatbots normally work in multi-user environments:
  - The issue may be addressed through cross-verification.
  - After acquiring a piece of new knowledge, the agent can store it in an unverified knowledge buffer.
    - Next, while chatting with some other users in future sessions to accomplish related tasks, the chatbot can ask them to verify the unverified knowledge.
Outline

- Continual learning with learning after deployment
- Continual learning to ground new language expressions
- Continual learning of factual knowledge
- Dealing with wrong knowledge from users
- Summary
Summary

- Classic ML: isolated and closed-world offline learning
  - No learning after deployment

- Dialogue systems or any AI agent should continuously learn after deployment or on the job (Chen and Liu, 2018; Liu, 2020)
  - The agent becomes starter and smarter

- Current techniques are still in their infancy, but
  - Some methods are ready for practical applications.
Thank You

Q&A