Continual Learning Dialogue Systems
– Learning during Conversation

Sahisnu Mazumder
Multimodal Dialogue & Interactions
Intel Labs, USA

Bing Liu
Department of Computer Science
University of Illinois at Chicago

Introduction

- Classical 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 tutorial**: Learning after/post model deployment on the job, particularly, in the dialogue domain.
  - In the open-world with unknowns and distribution changes.
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.
    - must learn and adapt continually to achieve autonomy.
  - Without this capability, an AI agent is not truly intelligent.
Self-driving cars: A motivating example

- Self-driving cars cannot reach human-level of driving with only rules and off-line training.
  - Impossible to cover all corner cases
  - Real-world is full of unknowns or novelties.
- Has to learn & adapt continuously in its interaction with humans and the environment by itself.
A personal experience with a self-driving car

- I consulted for a self-driving car company for a year.
- Once we took a self-driving car for a field test on the road.
  - At a T-junction, the car suddenly stopped and refused to move.
    - Every direction was clear, and nothing was on the road.
- We had to take over manually and drove the car to the lab.
  - Debugging found that a sensor detected a pebble on the road.
    - If the car could say "I detected an unknown object here. What should I do?" we would have said "It is safe. Go ahead."
  - The car can then learn the new object so that it will have no issue next time.
    - That is, learning on the fly or on the job.
AI in open-world & Learning on the job
(Liu 2020, Liu & Mazumder 2021)

- AI has to learn **continually** and **interactively** on the job (after/post deployment) in the open-world
  ✓ Needs to communicate with humans, e.g., **take human instructions**, ask when it has difficulty in the open world with **unknowns**, and learn from humans.
  ✓ Communicating in natural language (NL) is a natural choice.

- **Dialogue systems** with on-the-job learning (i.e., **learning during conversation**) capability is essential for the next-generation dialogue systems

Dialogue & Interactive Systems: Tasks

- **Question-answering**: provide concise, direct answers to user queries based on rich knowledge drawn from various data sources *(QA Bots)*
  - text collections such as Web documents
  - pre-compiled KBs such as sales and marketing database, factual KBs, etc.

- **Task completion**: help user accomplish their tasks *(task completion bots)*
  - restaurant reservation, meeting scheduling, trip planning

- **Social chats**: converse seamlessly and appropriately with users and provide useful recommendations *(social chatbots)*

---

Example – Human-agent dialogues for business decision making

<table>
<thead>
<tr>
<th>Turn</th>
<th>Dialogue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>usr</em>: Good morning!</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: Good morning, I hope you had a great weekend.</td>
</tr>
<tr>
<td>2</td>
<td><em>usr</em>: not really.</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: You sound depressed. What can I do for you?</td>
</tr>
<tr>
<td>3</td>
<td><em>usr</em>: I’d like to know where sales are lagging behind our forecast?</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: The worst region is [country], where sales are 15% below projections.</td>
</tr>
<tr>
<td>4</td>
<td><em>usr</em>: Do you know why?</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: The forecast for [product] growth was overly optimistic.</td>
</tr>
<tr>
<td>5</td>
<td><em>usr</em>: How can we turn this around?</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: Here are the 10 customers in [country] with the most growth potential, per our CRM model.</td>
</tr>
<tr>
<td>6</td>
<td><em>usr</em>: Can you set up a meeting with the CTO of [company]?</td>
</tr>
<tr>
<td></td>
<td><em>agt</em>: Yes, I’ve set up a meeting with [person name] for next month when you are in [location].</td>
</tr>
<tr>
<td>7</td>
<td><em>usr</em>: Thanks!</td>
</tr>
</tbody>
</table>

Dialogue & Interactive Systems: Broad Categories

- **Task-oriented chatbots**
  - Complete tasks based on users’ requests, e.g., providing the requested information and taking actions.
  - Personal assistants: Alexa, Siri, Google Home, etc.

- **Chit-chat systems**
  - Conduct chit-chat type of conversation in wide range of topics without having a specific goal to complete.
  - Example – ELIZZA, Microsoft XiaoIce.
Task-oriented Dialogue Systems

- Often designed as a Modular system –
  - **Natural Language Understanding (NLU):** Identify user intents and extract associated information
  - **State Tracking:** Track the dialogue state to capture all essential information in conversation so far
  - **Dialogue Policy:** Select the next action based on the current state
  - **Natural Language Generation (NLG):** Convert agent actions to natural language responses
Natural Language Understanding (NLU)

- **Domain Classification**: Classifying the domain of the task.
  - E.g., is this user talking about airlines, programming an alarm clock, or dealing with their calendar?

- **Intent Classification**: What general task or goal is the user trying to accomplish?
  - E.g., Find a Movie, or Show a Flight, or Remove a Calendar Appointment.

- **Slot filling**: Extract the slots and fillers that the user intends the system to understand from their utterance with respect to their intent  → a sequence labeling problem
Natural Language Understanding (NLU): Examples

“Find me an italian restaurant for dinner”

Domain: RESTAURANT
Intent: SearchRestaurant
Slots: {CUISINE: Italian, TIME: dinner}

“Wake me up tomorrow at 6”

Domain: ALARM-CLOCK
Intent: Set-Alarm
Slots: {DATE: 2022-07-11, TIME: 06:00}
State Tracking

User: Find me an italian restaurant for dinner.
   inform(cuisine=Italian; time=dinner)
System: Okk. What price range do you prefer?
User: may be in moderate
   inform(cuisine=Italian; time=dinner; price_range= $$)
System: Sure. There an nice italian restaurant at moderate price range nearby.
User: Awesome! What’s the name and address?
   inform(cuisine=Italian; time=dinner; price_range= $$);
   request(name; address)
System: The restaurant name is Franco’s Ristorante and address is
   300 W 31st St, Chicago, IL 60616.

dialogue-state: the entire state of the frame at this point (the filler of each slot), as well as the user’s most recent dialogue act, summarizing all of the user’s constraints.
Dialogue Policy (DP) Learning

- Decide what action the system should take next, that is, what dialogue act to generate.
  - At turn $i$ in the conversation, we want to predict which action $A_i$ to take, based on the entire dialogue state [entire sequence of dialogue acts from the system (A) and from the user (U)].

$$
\hat{A}_i = \text{argmax}_{A_i \in A} P(A_i | (A_1, U_1, ..., A_{i-1}, U_{i-1})
$$
Natural Language Generation (NLG)

Once the policy has decided what speech act to generate, the NLG generates the text response

- modeled in two stages, content planning (what to say) and sentence realization (how to say it).

Sentence realization is commonly achieved through delexicalization

- Mapping from frames to delexicalized sentences using encoder decoder models
Social Chatbots
(Jurafsky et.al. 2020)

- Often Implemented using a unitary (non-modular) system
  - Rule-based systems
    - Works based on pattern/transform rules
    - Examples - ELIZA (Weizenbaum, 1966) and PARRY (Colby et al., 1971)
  - Corpus-based systems
    - Mimic human conversations by training on large amounts of human-human conversational data
    - Example - Microsoft XioIce
Social Chatbots: Rule-based systems

- **ELIZA (Weizenbaum, 1966)** worked by pattern/transform rules like the following one:

  (Ø YOU Ø ME) [pattern]
  ->
  (WHAT MAKES YOU THINK I 3 YOU) [transform]

  You hate me
  into:
  WHAT MAKES YOU THINK I HATE YOU

- Each ELIZA pattern/rule is linked to a keyword that might occur in a user sentence

  Example dialogue by ELIZA

  Men are all alike.
  IN WHAT WAY
  They’re always bugging us about something or other.
  CAN YOU THINK OF A SPECIFIC EXAMPLE
  Well my boyfriend made me come here.
  YOUR BOYFRIEND MADE YOU COME HERE
  He says I’m depressed much of the time.
  I AM SORRY TO HEAR YOU ARE DEPRESSED
  ..
Social Chatbots: Corpus-based systems
(Jurafsky et.al. 2020)

- **Response by retrieval**: Considering user’s turn as a query $q$, the goal is to retrieve and repeat some appropriate turn $r$ as the response from a corpus of conversations $C$ *(training set for the system)*.

- Score each turn in $C$ as a potential response to the context $q$ and select the highest-scoring one.

\[
\text{response}(q, C) = \arg\max_{r \in C} \frac{q \cdot r}{|q||r|}
\]
Social Chatbots: Corpus-based systems

- **Response by Generation**: Models response production as an encoder-decoder task

  - transducing from the user’s prior turn to the system’s turn (Ritter et al., 2011; Sordoni et al., 2015b; Vinyals and Le, 2015; Shang et al., 2015).

\[
\hat{r}_t = \arg\max_{w \in V} P(w|q,r_1...,r_{t-1})
\]
Response Generation in Dialogue Systems: weaknesses

- Dull responses
  - e.g., "I don’t know", "I don’t have a clue".

- Out-of-context responses

- Semantic Conflicts

- Inconsistent response

- Lack of context-awareness

- so on ....
Knowledge grounding in Conversation: Example

I watched ``The Dark Knight`` yesterday. It was awesome.

Did you like Christian Bale’s acting?

Knowledge grounding makes conversation interesting and intelligent!
Knowledge-grounded Conversation Modeling

Recently researchers have begun to explore how to ground the chitchat in world knowledge to make the conversation more contentful and interesting.


Ghazvininejad et. al., AAAI-2018

Young et.al., AAAI-2018
Dialogue systems in the open-world: Challenges

- Built with **pre-collected training data, fixed rules** and **pre-compiled knowledge bases** (KBs)
  - Great deal of manual effort is needed
  - No matter how much data is collected, can’t cover all possible variations of natural language.

- Pre-compiled KB **can’t cover all rich knowledge** needed in practice
  - Knowledge bases are incomplete *(West et. al. 2014)*
  - KB of existing systems does not grow over time!

*West et. al. Knowledge base completion via search-based question answering. WWW-2014*
Chatbots should learn continually after deployment (Chen & Liu 2018, Liu 2020, Liu & Mazumder 2021)

- **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.
Learning during Conversation: **Scopes**
(Liu 2020, Liu & Mazumder 2021)

- **Passive learning**
  - Learning by reading web corpus, web tables or past conversation
    [information extraction]

- **Interactive learning**
  - Learning through interactive multi-turn dialogue **[our focus]**
Goals of this Tutorial

- Introducing the paradigm of lifelong or continual learning and discuss various related problems and challenges in the context of conversational AI applications.

- Recent advancements in continual learning in Chatbots after model deployment via interactions with end-users.

- A discussion on the future scope for continual conversational learning and open challenges.
Outline

I. **Lifelong and Continual Learning: An Introduction**
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Classic definition of lifelong/continual learning

- Learn a sequence of tasks, $T_1, T_2, \ldots, T_N, \ldots$ incrementally. Each task $t$ has a training dataset $D_t = \{(x_i, y_i)\}_{i=1}^n$.

- **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.

- **Question:** Where do the task $T_{N+1}$ and its training data $D_{N+1}$ come from?
  - Currently, they are given by the user.
Continual learning with learning after deployment

(Chen & Liu, 2018, Liu, 2020, Liu & Mazumder 2021)

Orange lines:
Learning after model deployment
(Learning on the job)

(more details)
Liu, Robertson, Grigsby, and Mazumder. Self-Initiated Open World Learning for Autonomous AI Agents.
AAAI Spring Symposium, 2022.

SIGIR-2022, Madrid, July 11, 2022
Characteristics of continual learning
(Chen and Liu, 2018, Liu, 2020)

- **Continuous incremental learning process** (no forgetting)
  - *Without forgetting*: Learning a new task should not forget the past.

- **Knowledge accumulation in KB** (long-term memory)

- **Knowledge transfer/adaptation** (across tasks) (Ke, Liu, Huang, 2020)
  - *Using/adapting past knowledge* to help learn new tasks

- **Learning after deployment** (on the job). *Self-supervision*
  learning using the *accumulated knowledge* and *interaction* with *humans & environment.*
Closed-world assumption and open-world
(Fei et al, 2016; Shu et al., 2017)

- **Supervised learning:**
  - **Training data:** \( D^{train} = \{D_1, D_2, \ldots, D_t\} \) of class labels \( Y^{train} = \{l_1, l_2, \ldots, l_t\} \).
  - **Test data:** \( D^{test}, Y^{test} \in \{l_1, l_2, \ldots, l_t\} \)

- **Classic paradigm:** closed-world assumption: \( Y^{test} \subseteq Y^{train} \)
  - Classes appeared in testing must have appeared in training, **nothing new**

- **Open-world** (with out-of-distribution data, OOD) \( Y^{test} \neq Y^{train} \)
  - **Test data:** \( D^{test}, Y^{test} \in \{l_1, l_2, \ldots, l_t, L_0\} \)
  - **L_0:** novel or unseen classes
  - A system **unable to identify anything new/novel** cannot learn by itself

- **Novelty is a key motivation** for lifelong or continual learning

---

SIGIR-2022, Madrid, July 11, 2022
- Shu, Hu and Liu. DOC: Deep Open Classification of Text Documents. EMNLP 2017
On the job continual learning (CL): Main steps
(Chen and Liu, 2018, Liu, 2020)

- **Identify a new task to learn** (tasks not given)
  - **Discover novel instances (OOD):** e.g., classify instances in $D^{test}$ to $Y^{train}$ and detect *novel instances* $D^{novel} \subseteq D^{test}$ belonging to unknown classes $L_0$
  - **Identify the unseen/new classes** in $D^{novel}$, $L_0 = \{l_{t+1}, l_{t+2}, \ldots\}$
  - **Create a task with classes** in $L_0 = \{l_{t+1}, l_{t+2}, \ldots\}$ to be learned.

- **Acquire ground-truth training data** (training data not given)
  - Gather additional ground-truth data if needed

- **Learn the task incrementally** (one-shot or few-shot CL)
Lifelong Interactive Learning in Conversation (LINC)

(Liu and Mazumder 2021)

• The tasks have to be self-discovered on-the-fly and the training data has to be found by the agent.
  ✓ A new learning task $T_{N+1}$ is formed when the agent needs to or can learn a piece of knowledge or encounters a problem in conversation.
  ✓ Could not understand a user utterance or could not answer a user question

• In order to learn the new task $T_{N+1}$, it needs to formulate a plan to obtain the ground truth training data $D_{N+1}$ on the job.
  ✓ That is, to interact with the user and ask the user questions and learn from it.
  ✓ This learning process is like human on-the-job learning.
Example - a learning greeting bot in a hotel
(Chen and Liu 2018)

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

- See a new guest. Bot recognizes the guest as new
  - Bot: “Welcome to our hotel! What is your name, sir?”
  - Guest: “David”
  - Bot learns to recognize David automatically
    - take pictures of David, and
    - learn to recognize David

- See David next time.
  - Bot: “Hello David, how are you today?”
Example - a learning greeting bot in a hotel
(Chen and Liu 2018)

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

- See a new guest. Bot recognizes the guest as new.
  - Bot: “Welcome to our hotel! What is your name, sir?”
  - Guest: “David”
  - Bot learns to recognize David automatically
    - take pictures of David, and
    - learn to recognize David

- See David next time.
  - Bot: “Hello David, how are you today?”

Example - knowledge learning in conversation

Humans Learn and Leverage Knowledge in Lifelong Manner!

Knowledge learning happens in a multi-user environment


Session 1

USER1

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

Where is Stockholm?

Stockholm is the capital of Sweden

USER2

Hey, I am planning for a Europe tour soon

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

USER3

USER2 / agent

Session 2

USER2 / agent
Outline

I. Lifelong and Continual Learning: An Introduction
II. **Continuous Knowledge Learning during Conversation**
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Knowledge learning in chatbots: Opportunities
(Liu and Mazumder 2021)

1. Extracting facts from user utterances
   - Did you watch anything yesterday?
   - Watched Forest Gump. The movie was awesome. Liked Tom Hanks' performance a lot!
   - Extracted Facts:
     - (Forest Gump, is a, movie)
     - (Tom Hanks, acted in, Forest Gump).

2. Ask questions to learn about unknown entities and concepts.
   - Hey, is there any good place around for having sushi?
   - What is sushi?
   - Japanese dish.
   - Learned new concepts/entities:
     - (Sushi, is, food)
     - (Sushi, has cuisine, Japanese).

3. Ask and infer new facts in conversation
   - When the chat-bot cannot answer an user query, it can ask for some related supporting facts and then infer the answer.
Continuous Factual knowledge learning in dialogues
(Mazumder et. al. 2019; 2020)

- 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* (Mazumder et. al. 2019); **IKAI**: *Interactive Knowledge Acquisition and Inference* (Mazumder et. al. 2020)
  - to continuously and interactively learn and infer new knowledge during conversations
Two types of queries or questions
(Mazumder et. al. 2019; 2020)

- **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

Stores acquired Facts (Triples)

KB: Collection of Triples

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

- **Interaction Module** \( \mathcal{I} \)
  - decides whether to ask or not, and formulates questions to ask the user for supporting facts
- **Inference Module** \( \mathcal{M} \)
  - Performs inference over the acquired Facts and existing KB

**Knowledge Base** \( \mathcal{K} \)

Stores acquired Facts (Triples)

**Triple Store**

(\textit{Boston, LocatedInCountry, USA})

**Knowledge Graph**

- Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019
Assumptions – Knowledge learning in dialogues
(Mazumder et. al. 2019; 2020)

- Focus on 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
    - Opposed to the teacher-student setup - the teacher is assumed to know everything.
When the user asks a Wh-question
(Mazumder et. al. 2019)

Given a user query \((h, r, ?)\) [or \((?, r, t)\)], the 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**

**Proposed Soln.**

an engine for Continuous and Interactive Learning of Knowledge (CILK)
Interactive knowledge learning in dialogue: example
(Mazumder et al. 2019)

USER: In what Country is Boston located?

CILK: Knowledge Base

CILK: I don't know what "located in Country" means? Can you provide me an example?

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

[Query]

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

[Ask for Clue]

USER: (London, LocatedInCountry, UK). “London is located in UK.”

[SF1]

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

[Ask for Entity Fact]

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

[SF2]

CILK: (Boston, LocatedInCountry, USA) “Boston is located in USA.”

[Answer]
CILK: The Inference Module

- We use the neural knowledge base embedding (KBE) [Bordes et al., 2011, 2013; Yang et al., 2014] for learning $\mathcal{M}$

- For evaluation, we adopt DistMult [Yang et al., 2014]

\[
\begin{align*}
\text{Triple: } & (h, r, t) \\
\text{x}_h, x_t \text{ and } x_r \\
\text{Entity Embed.} & \begin{cases}
\mathbf{v}_h = W_E \mathbf{x}_h \\
\mathbf{v}_t = W_E \mathbf{x}_t \\
\mathbf{v}_r = W_R \mathbf{x}_r
\end{cases} \\
\text{Relation Embed.} & \\
\text{DisMult Scoring Function} & S(h, r, t) = \mathbf{v}_h^T \text{diag}(\mathbf{v}_r) \mathbf{v}_t = \sum_{i=1}^{N} v_h[i] v_r[i] v_t[i] \\
\text{Max-margin Raking Loss} & \mathcal{L} = \sum_{d \in D^+} \sum_{d' \in D^-} \max\{S(d') - S(d) + 1, 0\}
\end{align*}
\]
CILK maintains a **threshold buffer** $T$ that stores entity and relation specific prediction thresholds and updates it continuously over time.

$$T[z] = \frac{1}{2|D_{vd}|} \sum_{(q, E^+, E^-) \in D^z_{vd}} S(q, e^+_i)$$

$\mu^+_E = \frac{1}{|E^+|} \sum_{e^+_i \in E^+} S(q, e^+_i)$

$\mu^-_E = \frac{1}{|E^-|} \sum_{e^-_i \in E^-} S(q, e^-_i)$

For tail query triple $S(e, r, e^+_i)$

For head query triple $S(e^+_i, r, e)$

**mean scores of triples involving +ve entities**

**mean scores of triples involving -ve entities**

Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019
Designing the Interaction Module: CILK’s Interaction Strategy

- CILK has to acquire supporting facts to learn embeddings of $e$ and $r$

  - user can only provide very few supporting facts per session → may not be sufficient for learning good embeddings of $e$ and $r$

  - Asking for too many SFs can be annoying and also, is unnecessary for entity and/or relation with good embeddings.

  - Need a sufficiently good validation set for learning $\mathcal{T}[e]$ and $\mathcal{T}[r]$

  ↓

  Ask for SFs for the known entities and/or relations for which CILK is not confident enough, besides the unknown ones.

---

- Mazumder, Liu, Wang, and Ma. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. SIGDIAL-2019
Acquiring Knowledge with limited Interaction: Improving Skillset over time

- A **performance buffer** $\mathcal{P}$ is used to store the performance statistics of $M$.

  $\mathcal{P}[e]$ and $\mathcal{P}[r]$ denote the **MRR achieved by** $M$ while answering queries involving $e$ and $r$ respectively, *evaluated on the validation dataset* $D_{vd}$.

- At the end of each dialogue session, CILK **detects bottom $\rho$% of query relations and entities** in $\mathcal{P}$ based on MRR scores.
  - **diffident relation and entity sets** for the next dialogue session.

**Strategy**

Ask for supporting facts for **diffident** and/or **unknown** query relation or entity
Performance improves with the increase in (acquired) entity fact triples (specially, for WordNet).

<table>
<thead>
<tr>
<th>(#C, #EF)</th>
<th>WordNet</th>
<th></th>
<th>Nell</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>H@1</td>
<td>H@10</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0.30</td>
<td>22.09</td>
<td>37.83</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>0.32</td>
<td>23.00</td>
<td>39.25</td>
</tr>
<tr>
<td>(1, 3)</td>
<td>0.33</td>
<td>25.27</td>
<td>40.95</td>
</tr>
<tr>
<td>(1, 3)-U</td>
<td>0.31</td>
<td>23.52</td>
<td>38.15</td>
</tr>
<tr>
<td>(2, 2)</td>
<td>0.32</td>
<td>23.43</td>
<td>39.05</td>
</tr>
</tbody>
</table>

# clues, # entity facts acquired per session

Performance Buffer disabled
(2) When the user asks a fact-verification question
(Mazumder et. al. 2020)

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?” [Query]
IKAI: I do not know what “CitizenOf” means? Can you provide me an example? [Ask for Clue]
USER: (David Cameron, CitizenOf, UK). “David Cameron is a citizen of UK.” [SF1]
IKAI: Got it. Can you tell me how “Obama” and “Honolulu” are related? [CLQ]
USER: (Obama, BornIn, Honolulu). “Obama was born in Honolulu.” [SF2]
IKAI: Got it. Can you tell also me how “Honolulu” and “Hawaii” are related? [MLQ]
USER: (Honolulu, CapitalOfState, Hawaii). “Honolulu is the state capital of Hawaii.” [SF3]
IKAI: (Obama, CitizenOf?, USA) holds. “Yes, Obama is a US citizen.” [Answer]
The IKAI approach
(Mazumder et. al. 2020)

- 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.
C-PR + Compositional Vector Space*: Inference Module of IKAI
*(Neelakantan et. al. 2015)

**Query:** CitizenOf (Obama, USA)?

- Encodes the path feature \( p \in P_c \) enumerated by C-PR using RNN to learn a vector representation of \( v_p \).

- Uses same RNN to encode the query relation \( r \) as \( v_r \).

- Inference -

\[
P(r|s, t) = \text{sigmoid}\left(\frac{1}{|P_c|} \sum_{p \in P_c} \cos(v_r, v_p)\right).
\]

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. *Compositional vector space models for knowledge base completion*. In ACL, 2015.
Finite State Machine: Interaction Module \( \mathcal{I} \)

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

- **States**
- **Actions**
- **Initial states**
- **Final states**
- \( \Delta : (S, A) \rightarrow S \) \hspace{1cm} 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>

**State Transition Conditions**

\( \text{for current state bits } S_i [\cdot] \)

<table>
<thead>
<tr>
<th>Action Id : Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_0 ) : Search source ( (s) ), target ( (t) ) entities and query relation ( (r) ) in KB.</td>
<td></td>
</tr>
<tr>
<td>( a_1 ) : Ask user to provide an clue/example for query relation ( r ).</td>
<td></td>
</tr>
<tr>
<td>( a_2 ) : Ask user to provide a missing link for path feature completion.</td>
<td></td>
</tr>
<tr>
<td>( a_3 ) : Ask user to provide a connecting link to add a new entity to the KB.</td>
<td></td>
</tr>
<tr>
<td>( a_4 ) : Extract path features between source ( (s) ) and target ( (t) ) entities using C-PR.</td>
<td></td>
</tr>
<tr>
<td>( a_5 ) : Invoke prediction model for inference.</td>
<td></td>
</tr>
</tbody>
</table>
Working of IKAI’ FSM: a given Session

**OKBC query:** q = (Obama, CitizenOf?, USA)

**Query strategy formulation postpones; clue strategy formulation starts**

![Diagram showing the FSM for query processing](image)

**State Bits**

<table>
<thead>
<tr>
<th>S0</th>
<th>0000000000</th>
<th>a0</th>
<th>S1</th>
<th>1010000000</th>
<th>a1</th>
<th>S2</th>
<th>1011000000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>110100111</td>
<td>a5</td>
<td>S5</td>
<td>11100110</td>
<td>a2</td>
<td>S4</td>
<td>11100100</td>
</tr>
<tr>
<td></td>
<td>(q, s0)</td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(q, s0)</td>
<td></td>
</tr>
</tbody>
</table>

**Processing Stack (PS)**

- Top

**Processing of remaining clues**

**Query strategy formulation resumes**

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

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

**[Ask for Clue]**

**USER:** (David Cameron, CitizenOf?, UK). “David Cameron is a citizen of UK.” [SF1]

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

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

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

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

**IKAI:** (Obama, CitizenOf?, USA) holds. “Yes, Obama is a US citizen.” [Answer]

- **a0:** Search source (s), target (t) entities and query relation (r) in KB.
- **a1:** Ask user to provide an clue/example for query relation r.
- **a2:** Ask user to provide a missing link for path feature completion.
- **a3:** Ask user to provide a connecting link to add a new entity to the KB.
- **a4:** Extract path features between source (s) and target (t) entities using C-PR.
- **a5:** Invoke prediction model for inference.

**SIGIR-2022, Madrid, July 11, 2022**


54
IKAI - Performance Evaluation

IKAI achieves best performance overall

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Models</th>
<th>% TTO</th>
<th>Freebase</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50%</td>
<td>0.0</td>
<td>0.492</td>
<td>0.507</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>0.545</td>
<td>0.580</td>
<td>0.578</td>
</tr>
</tbody>
</table>

IKAI Performance improvement due to User Interaction

Path completion via Blind Guessing

+ve F1 score

Macro F1 score

% of test Triples observed

Continuous learning past tasks (relations) Is disabled

IKAI achieves best performance overall

Lexical knowledge acquisition in dialogues
(Otsuka et. al. 2013)

- **Goal**: acquire the **attributes of unknown concepts** from users during dialogues

- propose a method for **generating more specific questions** than simple *wh*-questions to acquire the attributes

  - **Task**: *estimate the cuisine type of a restaurant from its name*, which is assumed to be unknown to the system
  - well-distributed **confidence measure** (CM) on the attributes to generate more specific questions.
  - Two basic CMs: (1) *character and word distributions* in the target database and (2) frequency of occurrence of restaurant attributes on Web pages.
Question Generation
(Otsuka et. al. 2013)

- Determines a question type on the basis of CM.
  - The CM is estimated for each cuisine type $c_j$ in the target database.

$$\text{num} = \min(n) \text{ s.t. } \sum_{j=1}^{n} CM(c_j) > \theta,$$

$CM(c_j)$ is a confidence measure for cuisine type $c_j$ its descending order

<table>
<thead>
<tr>
<th>num</th>
<th>Question form</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes-No question</td>
<td>Is it cuisine $c_1$?</td>
</tr>
<tr>
<td>2</td>
<td>Alternative question</td>
<td>Which cuisine is it, $c_1$ or $c_2$?</td>
</tr>
<tr>
<td>3</td>
<td>3-choice question</td>
<td>Which cuisine is it, $c_1$, $c_2$, or $c_3$?</td>
</tr>
<tr>
<td>$\geq$4</td>
<td>Wh-question</td>
<td>What cuisine is it?</td>
</tr>
</tbody>
</table>
CM_D Calculation: using Word and Character Distribution in Database (Otsuka et. al. 2013)

\[
CM_D(s_i, c_j) = \frac{1}{Z} \exp \left[ \tilde{\lambda}(c_j) \cdot \tilde{\phi}(s_i) \right]
\]

Input: Restaurant name

Azuma Sushi (東寿司)

Output: CM_D

Japanese restaurant: 0.9
Japanese pub: 0.05
Cafe: 0.0006

Classifier: Maximum entropy (ME) model

Training data

DB

<table>
<thead>
<tr>
<th>Restaurant name</th>
<th>Cuisine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maru Sushi (まる寿司)</td>
<td>Japanese restaurant</td>
</tr>
<tr>
<td>Tanaka Sushi (田中寿司)</td>
<td>Japanese restaurant</td>
</tr>
<tr>
<td>Koikoi (こいこい)</td>
<td>Japanese pub</td>
</tr>
<tr>
<td>Sushi Goichi (寿司 五一)</td>
<td>Japanese restaurant</td>
</tr>
<tr>
<td>Quinci CENTRARE</td>
<td>Italian</td>
</tr>
<tr>
<td>Hyakuraku (百楽)</td>
<td>Chinese restaurant</td>
</tr>
<tr>
<td>C’s ave cafe</td>
<td>Cafe</td>
</tr>
</tbody>
</table>

feature vector (n-gram, words etc.) obtained from a restaurant name

cuisine type

restaurant name
**CM_W Calculation: Using the Web**

(Otsuka et. al. 2013)

\[
P_{freq}(c_j) = \frac{\sum_i w_i \cdot freq_i(c_j)}{\sum_{c_j} \sum_i w_i \cdot freq_i(c_j)} \quad w_i = \frac{1}{rank(i) \cdot cuisine(i)}
\]

\[
CM_W(c_j) = \frac{\alpha_j P_{freq}(c_j)}{\sum_{c_j} \alpha_j P_{freq}(c_j)} \quad \alpha_j = \frac{P_{freq}(c_j)}{P_{freq}(c_1)}
\]
Lexical acquisition in dialogues: **Performance**

(Otsuka et. al. 2013)

Distribution of estimation results by CM values

<table>
<thead>
<tr>
<th>CM range</th>
<th>CM_D Correct</th>
<th>CM_D Incorrect</th>
<th>CM_W Correct</th>
<th>CM_W Incorrect</th>
<th>CM_I Correct</th>
<th>CM_I Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 – 0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>0.1 – 0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>0.2 – 0.3</td>
<td>1</td>
<td>16</td>
<td>14</td>
<td>22</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>0.3 – 0.4</td>
<td>6</td>
<td>19</td>
<td>28</td>
<td>19</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>0.4 – 0.5</td>
<td>11</td>
<td>25</td>
<td>29</td>
<td>21</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>0.5 – 0.6</td>
<td>21</td>
<td>29</td>
<td>56</td>
<td>9</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>0.6 – 0.7</td>
<td>22</td>
<td>28</td>
<td>85</td>
<td>7</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>0.7 – 0.8</td>
<td>41</td>
<td>16</td>
<td>42</td>
<td>3</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>0.8 – 0.9</td>
<td>21</td>
<td>9</td>
<td>19</td>
<td>1</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>0.9 – 1.0</td>
<td>131</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>184</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>254</td>
<td>146</td>
<td>274</td>
<td>124</td>
<td>297</td>
<td>103</td>
</tr>
</tbody>
</table>
Knowledge acquisition in a rule-based system

(Liu and Mei, 2020)

- 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] \(F\): \{\((X, \text{is-a, hotel})\)\} \(B\): \{\((X, \text{has-address, Y})\)\}
- If user says, “\textit{I stayed in Hilton at 150 Pine Street},” the agent gets
  - a fact: \((\text{Hilton, is-a, hotel})\) & a belief: \((\text{Hilton, has-address, 150 Pine Street})\).
- If user says, “\textit{I stayed in Hilton},” the agent gets
  - A fact: \((\text{Hilton, is-a, hotel})\) 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

I. Lifelong and Continual Learning: An Introduction
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Learning to ground natural language (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.

- **Interactive Language Learning after deployment:**
  - via user demonstrations *(Wang et. al. 2016)*.
  - via multi-turn NL dialogues with the user *(Mazumder et. al. 2020)*
Learning Language Games through Interaction
(Wang et. al. 2016)

- A language learning setting for building adaptive natural language interfaces.
- inspired by Wittgenstein’s language games:
  - a human wishes to accomplish some task but can only communicate with a computer, who performs the actual actions.
  - The computer initially knows nothing about language and therefore must learn it from scratch through interaction, while the human adapts to the computer’s capabilities.

The SHRDLURN game
Interactive learning through language game (ILLG) Setting (Wang et al. 2016)

- **Goal**: transform a start state into a goal state, but the only action the human can take is entering an utterance.
  - The computer parses the utterance and produces a ranked list of possible interpretations based on its current model.
  - Human scrolls through the list and chooses the intended one.
  - For the computer to be successful, it **has to learn** the human’s language **quickly** over the course of the game.

The SHRDLURN game
## Compositional action space for SHRDLURN
(Wang et. al. 2016)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Semantics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>all()</td>
<td>all stacks</td>
</tr>
<tr>
<td>Color</td>
<td>cyan</td>
<td>brown</td>
</tr>
<tr>
<td>Color → Set</td>
<td>with(c)</td>
<td>stacks whose top block has color c</td>
</tr>
<tr>
<td>Set → Set</td>
<td>not(s)</td>
<td>all stacks except those in s</td>
</tr>
<tr>
<td>Set → Set</td>
<td>leftmost</td>
<td>rightmost(s)</td>
</tr>
<tr>
<td>Set Color → Act</td>
<td>add(s, c)</td>
<td>add block with color c on each stack in s</td>
</tr>
<tr>
<td>Set → Act</td>
<td>remove(s)</td>
<td>remove the topmost block of each stack in s</td>
</tr>
</tbody>
</table>

*‘remove rightmost orange block’*

→ remove(rightmost(with(orange))))
ILLG as a semantic parser
(Wang et. al. 2016)

- Maps natural language utterances (e.g., ‘remove red’) into logical forms (e.g., remove(with(red))).
  - Uses a log-linear model over logical forms (actions) $z \in Z$ given utterance $x$:
    $$p_{\theta}(z | x) \propto \exp(\theta^T \phi(x, z))$$
  - Parser generates many candidate logical forms.
  - Based on the human’s feedback, it performs online gradient updates on the parameters corresponding to simple lexical features.
    - n-grams (including skip-grams) conjoined with tree-grams on the logical form side.

Features defined over utterance and logical form
Prob. Assigned to possible mappings
**ILLG: Performance** (Wang et. al. 2016)

### Evaluation on 100 players on Mechanical Turk

<table>
<thead>
<tr>
<th>Most successful players (1st–20th)</th>
<th>Average players (21th–50th)</th>
<th>Least successful players (51th–)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rem cy pos 1, stack or blk pos 4, rem blk pos 2 thru 5, rem blk pos 2 thru 4, stack bn blk pos 1 thru 2, fill bn blk, stack or blk pos 2 thru 6, rem cy blk pos 2 fill rd blk (3.01)</td>
<td>remove the brown block, remove all orange blocks, put brown block on orange blocks, put orange blocks on all blocks, put blue block on leftmost blue block in top row (2.78)</td>
<td>Remove the center block, Remove the red block, Remove all red blocks, Remove the first orange block, Put a brown block on the first brown block, Add blue block on first blue block (2.72)</td>
</tr>
<tr>
<td>reinsert pink, take brown, put in pink, remove two pink from second layer, Add two red to second layer in odd intervals, Add five pink to second layer, Remove one blue and one brown from bottom layer (9.17)</td>
<td>remove red, remove 1 red, remove 2 4 orange, add 2 red, add 1 2 3 4 blue, emove 1 3 5 orange, add 2 4 orange, add 2 orange, remove 2 3 brown, add 1 2 3 4 5 red, remove 2 3 4 5 6, remove 2, add 1 2 3 4 6 red (8.37)</td>
<td>move second cube, double red with blue, double first red with red, triple second and fourth with orange, add red, remove orange on row two, add blue to column two, add brown on first and third (7.18)</td>
</tr>
<tr>
<td>holdleftmost, holdbrown, holdleftmost, blueonblue, brownonblue1, blueonorange, holdblue, holdorange2, blueonred2 , holdends1, holdrightend, hold2, orangeonorangerrightmost (14.15)</td>
<td>‘add red cubes on center left, center right, far left and far right’, ‘remove blue blocks on row two column two, row two column four’, remove red blocks in center left and center right on second row (12.6)</td>
<td>laugh with me, red blocks with one aqua, aqua red alternate, brown red red orange aqua orange, red brown red brown red brown, space red orange red, second level red space red space red space (14.32)</td>
</tr>
</tbody>
</table>
ILLG: Performance
(Wang et. al. 2016)

- **memorize**: featurize entire utterance and logical form non-compositionally;
- **half model**: featurize the utterances with unigrams, bi-grams, and skip-grams but conjoin with the entire logical form;
- **full model**: the proposed full model
- **full+prag**: the proposed full model with online pragmatics algorithm

<table>
<thead>
<tr>
<th>Method</th>
<th>top 10</th>
<th>top 20</th>
<th>top 50</th>
<th>all 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>memorize</td>
<td>25.4</td>
<td>24.5</td>
<td>22.5</td>
<td>17.6</td>
</tr>
<tr>
<td>half model</td>
<td>38.7</td>
<td>38.4</td>
<td>36.0</td>
<td>27.0</td>
</tr>
<tr>
<td>half + prag</td>
<td>43.7</td>
<td>42.7</td>
<td>39.7</td>
<td>29.4</td>
</tr>
<tr>
<td>full model</td>
<td>48.6</td>
<td>47.8</td>
<td>44.9</td>
<td>33.3</td>
</tr>
<tr>
<td>full + prag</td>
<td>52.8</td>
<td>49.8</td>
<td>45.8</td>
<td>33.8</td>
</tr>
</tbody>
</table>

Average online accuracy under various settings
Natural Language to Natural Language (NL2NL) matching (Mazumder et. al. 2020)

- **Goal:** An adaptable system to automatically serve as NLI to API-based applications.
  - One system for many diverse API-driven applications
  - Learning continuously or lifelong from users via interactions.

- **Approach:** Natural Language to Natural Language (NL2NL) matching

  - Each action (API) is attached with one or more natural language (NL) representation - a set of one or more API seed commands (SCs) just like a NL command from the user to invoke the API.
  - When the user issues a NL command, the system simply matches the command with one of the system’s SCs.

<table>
<thead>
<tr>
<th>API (arg : arg type)</th>
<th>Seed Commands (SCs)</th>
<th>Example NL command</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwitchOnLight(X1: location)</td>
<td>Switch on the light at X1; Put on light on X1</td>
<td>Power on the light at bedroom (X1)</td>
</tr>
<tr>
<td>SwitchOffLight(X1: location)</td>
<td>Switch off the light at X1; Power off the light at X1</td>
<td>Turn off the light at living room (X1)</td>
</tr>
<tr>
<td>ChangeLightColor (X1: location, X2: color)</td>
<td>Change the X1 light to X2; I want X1 light as X2</td>
<td>Please make the color of bedroom (X1) light blue (X2)</td>
</tr>
</tbody>
</table>
Command Matching & Learning (CML)  
(Mazumder et. al. 2020)

- CML works based on NL2NL matching idea.
- Consists of Three components
  - **SC (seed command) specification**
    - enable 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 in multi-turn dialogues to continually learn new SCs and paraphrases of API argument values.
**SC Specification (blocks-world)**

(Mazumder et. al. 2020)

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 Z1 {Z1=X1}</td>
<td>X1: ‘location’ (*)</td>
<td>add a block at (2, 3); put a block at (2, 3) delete blue block; take away blue block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Z1={X1}</td>
<td></td>
<td>move blue block to the left of cube; shift green cube to (4, 5) move blue block left by 2 units; shift green cube down by 3 units</td>
</tr>
<tr>
<td>Remove (X1)</td>
<td>2</td>
<td>remove Z1 {Z1=X1}</td>
<td>X1: ‘block_set’</td>
<td></td>
</tr>
<tr>
<td>Move (X1, X2)</td>
<td>3</td>
<td>move Z1 to Z2 {Z1=X1, Z2=X2}</td>
<td>X1: ‘block_set’, X2: ‘location’ (*)</td>
<td></td>
</tr>
<tr>
<td>MoveByUnits (X1, X2, X3)</td>
<td>4</td>
<td>move Z1 along Z2 by Z3 units {Z1=X1, Z2=X2, Z3=X3}</td>
<td>X1: ‘block_set’, X2: ‘direction’ (*), X3: ‘number’</td>
<td></td>
</tr>
<tr>
<td>UpdateColor (X1, X2)</td>
<td>5</td>
<td>change color of Z1 to Z2 {Z1=X1, Z2=X2}; color Z1 with Z2 {Z1=X1, Z2=X2}</td>
<td>X1: ‘block_set’, X2: ‘color’ (*)</td>
<td>set the shape of A to cube; make B square</td>
</tr>
<tr>
<td>Rename (X1, X2)</td>
<td>6</td>
<td>rename block Z1 to Z2 {Z1=X1, Z2=X2}</td>
<td>X1: ‘block_set’, X2: ‘name’ (*)</td>
<td>Name the block at (4, 5) as C; rename A to D</td>
</tr>
<tr>
<td>UpdateShape (X1, X2)</td>
<td>7</td>
<td>change shape of Z1 to Z2 {Z1=X1, Z2=X2}</td>
<td>X1: ‘block_set’, X2: ‘shape’ (*)</td>
<td></td>
</tr>
</tbody>
</table>
Command grounding module (CGM)  
(Mazumder et. al. 2020)

- **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.
Command grounding module (contd.)

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 (Mazumder et. al. 2020)


Experiment results
(Mazumder et. al. 2020)

**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 A-acc</th>
<th>BW Arg-F1</th>
<th>WPD A-acc</th>
<th>WPD Arg-F1</th>
<th>FB A-acc</th>
<th>FB Arg-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-JISF</td>
<td>57.56</td>
<td>52.42</td>
<td>57.26</td>
<td>50.36</td>
<td>33.64</td>
<td>54.41</td>
</tr>
<tr>
<td>CML-jac</td>
<td>66.44</td>
<td>63.70</td>
<td>72.68</td>
<td>81.26</td>
<td>88.58</td>
<td>96.75</td>
</tr>
<tr>
<td>CML-vsm</td>
<td>66.44</td>
<td>63.70</td>
<td>71.80</td>
<td>81.26</td>
<td>87.15</td>
<td>96.75</td>
</tr>
<tr>
<td>CML-embed</td>
<td>66.77</td>
<td>64.03</td>
<td>63.87</td>
<td>76.42</td>
<td>83.00</td>
<td>96.75</td>
</tr>
<tr>
<td>CML-vsm (-R)</td>
<td>63.48</td>
<td>61.15</td>
<td>66.07</td>
<td>72.56</td>
<td>75.48</td>
<td>84.15</td>
</tr>
<tr>
<td>CML-vsm (-U)</td>
<td>3.94</td>
<td>2.96</td>
<td>5.72</td>
<td>5.72</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>CML-jac + SCL</td>
<td>68.75</td>
<td>66.00</td>
<td>76.21</td>
<td>82.14</td>
<td>92.47</td>
<td>96.75</td>
</tr>
<tr>
<td>CML-vsm + SCL</td>
<td>68.09</td>
<td>65.35</td>
<td>74.89</td>
<td>81.71</td>
<td>93.32</td>
<td>96.75</td>
</tr>
<tr>
<td>CML-jac + SCL + APL</td>
<td>71.05</td>
<td>70.72</td>
<td>79.73</td>
<td>84.02</td>
<td>93.74</td>
<td>97.72</td>
</tr>
<tr>
<td>CML-vsm + SCL + APL</td>
<td>70.06</td>
<td>69.73</td>
<td>78.85</td>
<td>83.59</td>
<td>94.50</td>
<td>97.72</td>
</tr>
</tbody>
</table>
Outline

I. Lifelong and Continual Learning: An Introduction
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. **Open-Domain Dialogue Learning**
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Open-domain Dialogue Learning after Deployment

- Learning by extracting new training examples from conversations
  - **Self-feeding Chatbot (Hancock et. al. 2019):** extracts (context, response) pairs from the conversations and use it for continual training.

- Dialogue learning via role-playing games
  - **LIGHT WILD (Shuster et. al. 2020):** Human players converse with agents situated in an open-domain fantasy world and showed that by training agents on in-game conversations they progressively improve.
Self-feeding Chatbot
(Hancock et. al. 2019)

- Learning by extracting new training examples from conversations
  - If the conversation appears to be going well, the user’s responses become new training examples to imitate.
  - Otherwise, on making a mistake, it asks the user for feedback to obtain a relevant response.
- The agent is retrained periodically using all available data

Hancock, Bordes, Mazare, and Weston. Learning from Dialogue after Deployment: Feed Yourself, Chatbot!. ACL 2019.
Self-feeding Chatbot: Learning (Hancock et. al. 2019)

- **Initial training phase**
  - The agent is trained on two tasks using supervised Human-Human (HH) dialogue examples
    - **Task-1: DIALOGUE**
      - next utterance prediction, or what should I say next?
    - **Task-2: SATISFACTION**
      - how satisfied is my speaking partner with my responses?

---

Hancock, Bordes, Mazare, and Weston. Learning from Dialogue after Deployment: Feed Yourself, Chatbot!. ACL 2019.
Self-feeding Chatbot: Learning
(Hancock et. al. 2019)

- **Deployment phase**
  - The agent engages in multi-turn conversations with users, extracting new *deployment* examples of two types.
    - Each turn, the agent *observes the context* $x$ (i.e., the conversation history) and uses it to *predict its next utterance* $\hat{y}$ and its partner’s *satisfaction* $\hat{s}$.
    - If $\hat{s} > t$ *(threshold)*, the agent *extracts a new Human-Bot (HB) DIALOGUE example* using context $x$ and human’s response $y$ and continues the conversation.
    - If $\hat{s} < t$, the agent *requests feedback with a question* $q$, and the resulting feedback response $f$ is used to create a new example for the TASK-3: FEEDBACK task (what feedback am I about to receive?).

---

Hancock, Bordes, Mazare, and Weston. Learning from Dialogue after Deployment: Feed Yourself, Chatbot!. ACL 2019.
Self-feeding Chatbot: Working
(Hancock et. al. 2019)

Self-feeding Chatbot: Model Architecture
(Hancock et. al. 2019)

- **Dialogue agent:** built on the traditional Transformer architecture

- **SATISFACTION task:** The context $x$ is encoded with a Transformer and converted to the scalar satisfaction prediction $\hat{s}$.

- **DIALOGUE and FEEDBACK tasks:** set up as ranking problems: the model ranks a collection of candidate responses and returns the top-ranked one as its response.
  - Context $x$ is encoded with one Transformer and $\hat{y}$, $\hat{f}$ candidates encoded with another.
  - The score is calculated as the dot product of the encoded context and encoded candidate.

---

Hancock, Bordes, Mazare, and Weston. Learning from Dialogue after Deployment: Feed Yourself, Chatbot!. ACL 2019.
## Self-feeding Chatbot: Performance
(Hancock et. al. 2019)

<table>
<thead>
<tr>
<th>Human-Bot (HB)</th>
<th>Human-Human (HH) DIALOGUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIALOGUE</td>
<td>20k</td>
</tr>
<tr>
<td>Feedback</td>
<td>20k</td>
</tr>
<tr>
<td>-</td>
<td>30.3 (0.6)</td>
</tr>
<tr>
<td>20k</td>
<td>32.7 (0.5)</td>
</tr>
<tr>
<td>40k</td>
<td>34.5 (0.5)</td>
</tr>
<tr>
<td>60k</td>
<td>35.4 (0.4)</td>
</tr>
<tr>
<td>-</td>
<td>35.0 (0.5)</td>
</tr>
<tr>
<td>20k</td>
<td>36.7 (0.7)</td>
</tr>
<tr>
<td>60k</td>
<td>37.8 (0.6)</td>
</tr>
<tr>
<td>60k</td>
<td>39.7 (0.6)</td>
</tr>
</tbody>
</table>

**Accuracy (hits@1/20) on the DIALOGUE task’s hidden test set**

Dialogue learning via role-playing games
(Shuster et. al. 2020)

**Goal:** study the ability of an open-domain dialogue model to iteratively learn from conversations with intrinsically motivated humans.

- Built and deploy a role-playing game
  - Human players role-play characters and converse with other characters (that are learning models) situated within the game world.

- System iterates between collecting data of human-model interactions, retraining updated models on newly collected data, and redeploying them.

- Provides a natural metric to evaluate and compare models online using the continuation rate of players (how long they continue playing).

Shuster, Urbanek, Dinan, Szlam, and Weston. Dialogue in the wild: Learning from a deployed role-playing game with humans and bots. *In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*
Open-domain dialogue as a game
(Shuster et. al. 2020)

- **Core Game:** involves pairing two agents in a given setting
  - Where one is a human and other is a dialogue agent (with an underlying machine learning model).
  - Two players are assigned characters, with given names, backstories (personas), their current location and its description.
  - Each player’s goal is simply to act out (role-play) their character’s dialogue in the given situation.

Shuster, Urbanek, Dinan, Szlam, and Weston. Dialogue in the wild: Learning from a deployed role-playing game with humans and bots." In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021
Open-domain dialogue as a game
(Shuster et. al. 2020)

- **Role-Playing (Acting) Score:** an (automated) dungeon master (DM), to assess the quality of the player’s role-playing.

  - For each dialogue turn, apply a learned model to the human player’s dialogue, to **assess how likely their utterance is given the context**.
  - **convert the score in between 1 and 5 stars**, that is presented to the human player, to **reward them for good acting**.

- Acting scores are accumulated, and a player’s total score is presented on a leaderboard for further motivation.
Open-domain dialogue as a game: Example
(Shuster et. al. 2020)

Human (grounds keeper)

Location: Arch Hall. The main beauty of this room is it's Arches. Four massive arches surround the four walls of this room falling from each side. There are book shelves covered in old books...

Persona: I am a groundskeeper at the castle. My job is to keep the King's land immaculate. I have worked here for many years, since I was a young boy. I do hard labor.

grounds keeper: Good morning sir

assistant chef: Hello, keeper. What may I call you?

grounds keeper: My name is Skylar

assistant chef: Excellent choice. I have a new sauce recipe from bredwell across the sea. Would you like to try it?

Model (assistant chef)

grounds keeper: How kind of you, I would love to try it.

assistant chef: Of course. I will prepare it right away. You won't be disappointed.

grounds keeper: I don't believe you could ever disappoint me, Chef. Your food is always delicious.

assistant chef: I am glad you appreciate it. The king is very particular about the kinds of sauces that he wants to serve the guests.

grounds keeper: Thank you for allowing me to try it. I do believe the guests and the King will enjoy the sauce as well. I myself, am eager to see how it tastes.

assistant chef: It should be ready within the hour.

grounds keeper: I will be back then, I must check upon the grounds in the mean time.

Shuster, Urbanek, Dinan, Szlam, and Weston. Dialogue in the wild: Learning from a deployed role-playing game with humans and bots.” In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021
**Dialogue learning via role-playing games: Performance**

(Shuster et. al. 2020)

<table>
<thead>
<tr>
<th>Round</th>
<th>Model</th>
<th>Retrieval Model (Hits@1/20 ↑)</th>
<th>Generative Model (PPL ↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIGHT Test</td>
<td>LIGHT Test Unseen</td>
<td>WILD Test</td>
</tr>
<tr>
<td>Round 1</td>
<td>87.12</td>
<td>82.43</td>
<td>81.61</td>
</tr>
<tr>
<td>Round 2</td>
<td>87.65</td>
<td>82.70</td>
<td>84.60</td>
</tr>
<tr>
<td>Round 3</td>
<td>87.72</td>
<td>83.48</td>
<td>87.63</td>
</tr>
</tbody>
</table>

- **Round 1**: models trained on LIGHT MTurk data only.
- **Round 2**: models trained on LIGHT MTurk data + 50k WILD examples collected from Round 1
- **Round 3**: models trained on LIGHT MTurk data + 50k examples from Round 1 + an additional 180k examples collected from Round 2 deployment

*test locations do not overlap with the training set locations*
Outline

I. Lifelong and Continual Learning: An Introduction
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning After Deployment
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
A progressive slot filling model, *ProgModel*.

- Gradually expands at each batch by using a context gate for knowledge transfer.
  - *Word Embeddings Transfer* and *Gated Utterance Context Transfer*

- Using the transferred knowledge, each newly expanded component is trained in a progressive manner with new data.

- Previously trained components remain untouched to avoid catastrophic forgetting.
ProgModel: Inference Decision Engine
(Shen et. al. 2019)

- non-trainable separate component to avoid the potential catastrophic forgetting.

- consider two types of decision engines:
  - t-IDE: ProgModel using only the output $M^t$ as decision engine
  - c-IDE: for $i^{th}$ word, it combines all outputs from each component $M^t$

\[
\sum_{k=0}^{t} P^k(i) I^k(i)
\]

- $I^k(i) \rightarrow$ indicator function which is 1 if $i^{th}$ word is in the vocabulary of $M^k$
- $P^k_j(i) \rightarrow$ output probability of slot $j$ for the $i^{th}$ word from $M^k$
- The label with maximum probability is selected.

Aggregate (slot label) prob. for each word from all components
ProgModel Performance: ATIS Dataset (Shen et. al. 2019)

- **FT-AttRNN**: fine tunes current model only using new training data $U_t$
- **FT-Lr-AttRNN**: fine tunes current model using adjusted lower learning rate on $U_t$
- **FT-Cp-AttRNN**: copies the previous model and fine tunes the new copied model on $U_t$.
- **t-ProgModel**: using only output of $M_t$ as decision engine
- **c- ProgModel**: ProgModel using combined inference decision engine.

<table>
<thead>
<tr>
<th>Approach</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttRNN (upper bound)</td>
<td>92.12</td>
<td>92.89</td>
<td>93.04</td>
<td>93.56</td>
<td>95.13</td>
</tr>
<tr>
<td>FT-AttRNN</td>
<td>91.85</td>
<td>89.98</td>
<td>91.25</td>
<td>88.03</td>
<td></td>
</tr>
<tr>
<td>FT-Lr-AttRNN</td>
<td>91.96</td>
<td>86.46</td>
<td>88.03</td>
<td>86.58</td>
<td></td>
</tr>
<tr>
<td>FT-Cp-AttRNN</td>
<td>92.12</td>
<td>92.10</td>
<td>90.06</td>
<td>91.98</td>
<td>89.67</td>
</tr>
<tr>
<td>t-ProgModel</td>
<td>92.33</td>
<td>92.43</td>
<td>92.57</td>
<td>92.58</td>
<td></td>
</tr>
<tr>
<td>c-ProgModel</td>
<td>92.40</td>
<td>92.64</td>
<td>92.71</td>
<td>93.91</td>
<td></td>
</tr>
</tbody>
</table>
### ProgModel Performance: \textbf{Snips Dataset} (Shen et. al. 2019)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Approach</th>
<th>Batch 0</th>
<th>Batch 1</th>
<th>Batch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add To Playlist</strong></td>
<td>AttRNN (upper bound)</td>
<td>79.58</td>
<td>86.74</td>
<td>88.89</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>81.23</td>
<td>87.07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>78.99</td>
<td>86.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>79.58</td>
<td>84.67</td>
<td>87.15</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>86.12</td>
<td>88.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>85.51</td>
<td>87.25</td>
<td></td>
</tr>
<tr>
<td><strong>Book Restaurant</strong></td>
<td>AttRNN (upper bound)</td>
<td>79.49</td>
<td>89.78</td>
<td>90.03</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>88.71</td>
<td>88.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>88.57</td>
<td>87.89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>79.49</td>
<td>89.06</td>
<td>88.14</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>89.45</td>
<td>89.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>89.40</td>
<td>89.40</td>
<td></td>
</tr>
<tr>
<td><strong>Get Weather</strong></td>
<td>AttRNN (upper bound)</td>
<td>76.48</td>
<td>91.12</td>
<td>93.56</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>89.52</td>
<td>88.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>89.09</td>
<td>88.56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>76.48</td>
<td>89.82</td>
<td>90.09</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>90.73</td>
<td>93.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>89.92</td>
<td>90.95</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Approach</th>
<th>Batch 0</th>
<th>Batch 1</th>
<th>Batch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Play Music</strong></td>
<td>AttRNN (upper bound)</td>
<td>77.48</td>
<td>87.79</td>
<td>89.13</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>84.71</td>
<td>84.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>84.53</td>
<td>84.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>77.48</td>
<td>84.85</td>
<td>86.10</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>86.05</td>
<td>87.26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>87.00</td>
<td>88.45</td>
<td></td>
</tr>
<tr>
<td><strong>Rate Book</strong></td>
<td>AttRNN (upper bound)</td>
<td>92.64</td>
<td>98.45</td>
<td>99.07</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>96.87</td>
<td>96.83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>96.20</td>
<td>96.86</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>92.64</td>
<td>97.06</td>
<td>97.93</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>97.50</td>
<td>98.89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>98.19</td>
<td>98.20</td>
<td></td>
</tr>
<tr>
<td><strong>Search Creative Work</strong></td>
<td>AttRNN (upper bound)</td>
<td>66.32</td>
<td>89.01</td>
<td>89.67</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>85.93</td>
<td>85.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>84.69</td>
<td>84.45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>66.32</td>
<td>87.25</td>
<td>86.36</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>88.21</td>
<td>88.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>88.79</td>
<td>88.83</td>
<td></td>
</tr>
<tr>
<td><strong>Search Screening Event</strong></td>
<td>AttRNN (upper bound)</td>
<td>89.30</td>
<td>95.68</td>
<td>97.34</td>
</tr>
<tr>
<td></td>
<td>FT-AttRNN</td>
<td>93.40</td>
<td>94.53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Lr-AttRNN</td>
<td>91.87</td>
<td>93.56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT-Cp-AttRNN</td>
<td>89.30</td>
<td>93.81</td>
<td>94.56</td>
</tr>
<tr>
<td></td>
<td>t-ProgModel</td>
<td>95.01</td>
<td>96.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c-ProgModel</td>
<td>93.62</td>
<td>94.31</td>
<td></td>
</tr>
</tbody>
</table>
A continual learning benchmark for ToDS with 37 domains -

✓ four settings: intent recognition, state tracking, natural language generation, and end-to-end

✓ implement and compare multiple existing continual learning baselines
  ➢ regularization, rehearsal and architectural.

✓ propose a simple and effective architectural method based on residual adapters (Houlsby et al., 2019).
Continual Learning in ToDs: Problem Formulation
(Madotto et. al. 2020)

- Modularized setting by their input-out pairs:
  \[ H \rightarrow I \] (INTENT)
  \[ H \rightarrow I(s_1 = v_1, \ldots, s_k = v_p) \] (DST)
  \[ I(s_1 = v_1, \ldots, s_k = v_p) \rightarrow S \] (NLG)

- End-to-End (E2E) formulation:
  \[ H \rightarrow I(s_1 = v_1, \ldots, s_k = v_p) \]
  \[ H + I(s_1 = v_1, \ldots, s_k = v_p) \rightarrow S \]

- Dialogue Dataset: input-out pair from one of the four settings in consideration

\[ \text{H: Dialogue History} \]
\[ \text{I: intent or the api-name} \]
\[ \text{S: NL response} \]
AdapterCL
(Madotto et. al. 2020)

- Employ a decoder-only pre-trained Language Models (e.g. GPT-2)
  - trained by minimizing the negative log-likelihood
  - only the task-specific parameter are trained while the original weights are left frozen.

- parameterizes each task using Residual Adapters (Houlsby et al., 2019) and uses a perplexity-based classifier to select which adapter to use at testing time.
  - **Residual adapter**: trainable parameters added on top of each transformer layer, which steer the output distribution of a pre-trained model without modifying its original weights.
AdapterCL
(Madotto et. al. 2020)

- Residual adapter computation:

  \[ \text{Adapter}_{\mu_t}(H) = \text{ReLU}((\text{LN}(x)W^E_t)W^D_t + H) \]

  \[ H \in \mathbb{R}^{p \times d} \]

- To learn new task (Dataset \( D_t \)), spawn a new Adapter (\( \mu_t \)) and train its parameters.

- Loss is optimized over \( \mu_t \) to guarantee that each task is independently learned.
**AdapterCL**: Perplexity-Based Classifier  
(Madotto et. al. 2020)

- In the CL setting →
  - during **training** task-id is provided → $\mu_t$ is optimized over $D_t$
  - during **testing** task-id is **not** provided → model has to predict which adapter to use for accomplishing the task.

- Following (Wortsman et al. 2020), utilize the perplexity of each adapter over the input $X$ as a measure of uncertainty.
  - selecting the adapter with lowest perplexity → select the most confident model to generate the output sequence.

\[
\alpha_t = \text{PPL}_{\mu_t}(X) \quad \forall t \in 1, \cdots, N
\]

\[
t^* = \arg\min \alpha_0, \cdots, \alpha_N
\]
## Continual Learning in Task-Oriented Dialogue Systems: Performance (Madotto et. al. 2020)

<table>
<thead>
<tr>
<th>Method</th>
<th>+Param.</th>
<th>Mem.</th>
<th>INTENT</th>
<th>DST</th>
<th>NLG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy↑</td>
<td>JGA↑</td>
<td>EER↓</td>
<td>BLEU↑</td>
<td></td>
</tr>
<tr>
<td>VANILLA</td>
<td>-</td>
<td>Ø</td>
<td>4.08 ± 1.4</td>
<td>4.91 ± 4.46</td>
<td>48.73 ± 3.81</td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td></td>
<td>3.74 ± 1.4</td>
<td>3.81 ± 3.44</td>
<td>55.68 ± 7.09</td>
</tr>
<tr>
<td>EWC</td>
<td>2</td>
<td>Ø</td>
<td>3.95 ± 1.3</td>
<td>5.22 ± 4.46</td>
<td>58.2 ± 3.66</td>
</tr>
<tr>
<td>AGEM</td>
<td>-</td>
<td>t</td>
<td>τ</td>
<td></td>
<td>34.04 ± 6.36</td>
</tr>
<tr>
<td>LAMOL</td>
<td>-</td>
<td>Ø</td>
<td>7.49 ± 6.35</td>
<td>4.55 ± 3.48</td>
<td>66.11 ± 6.97</td>
</tr>
<tr>
<td>REPLAY</td>
<td>-</td>
<td>t</td>
<td>τ</td>
<td></td>
<td>81.08 ± 1.37</td>
</tr>
<tr>
<td>ADAPT</td>
<td>t</td>
<td>μ</td>
<td></td>
<td>90.46 ± 0.6</td>
<td><strong>35.06 ± 0.52</strong></td>
</tr>
<tr>
<td>MULTI</td>
<td>-</td>
<td>-</td>
<td>95.45 ± 0.1</td>
<td>48.9 ± 0.2</td>
<td>12.56 ± 0.2</td>
</tr>
</tbody>
</table>

E2E results in terms of Intent accuracy, Joint-Goal-Accuracy (JGA), Slot-Error-Rate (EER) and BLUE. +Param shows the additional number of parameters per task, and Mem the episodic memory size need per task.
Outline

I. Lifelong and Continual Learning: An Introduction
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning After Deployment
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Learning user behaviors and preferences

- Given a conversation context, the chatbot can learn the user’s behavioral and preference profile.
  - whether a user feels more excited or gets annoyed while conversing on a particular topic, what his/her likes and dislikes are etc.
  - PERSONALIZED MEMN2N (Luo et. al. 2019); $P^2$ BOT (Liu et. al. 2020)

Utilize user profile knowledge in modeling future conversations to make them more engaging with the user.

Example – Personalized goal-oriented dialogues

**Dialog 2**

*User Profile*

*Gender: Male, Age: Young, Dietary: Non-vegetable*

Good morning.

Hey dude, what is up?

Can you make a restaurant reservation in a cheap price range in Rome for six people with Spanish food?

Ok looking for options.

Searching in knowledge base...See results in Figure 1b

Is this one cool: The_Fancy_Pub

Let’s do it.

**Dialog 3**

*User Profile*

*Gender: Female, Age: Elderly, Dietary: Vegetable*

Good day madam, how could I assist you today?

Can you make a restaurant reservation in a cheap price range in Rome for six people with Spanish food?

Thank you madam I shall provide you with options shortly.

Searching in knowledge base...See results in Figure 1b

Would you consider this option: The_Place

Let’s do it.

---


Learning emotions, moods and sentiments

- Recognizing emotional state and sentiments (Zhou et al. 2018; Pamungkas 2019) of the user and leverage it to generate empathetic responses can be useful to building therapeutic chatbots.

- Emotional Chatting Machine (Zhou et al. 2018): can generate appropriate responses not only in content (relevant and grammatical) but also in emotion (emotionally consistent).

<table>
<thead>
<tr>
<th>User: Worst day ever. I arrived late because of the traffic.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Seq2Seq: You were late.</td>
</tr>
<tr>
<td>ECM (Like): I am always here to support you.</td>
</tr>
<tr>
<td>ECM (Happy): Keep smiling! Things will get better.</td>
</tr>
<tr>
<td>ECM (Sad): It’s depressing.</td>
</tr>
<tr>
<td>ECM (Disgust): Sometimes life just sucks.</td>
</tr>
<tr>
<td>ECM (Angry): The traffic is too bad!</td>
</tr>
</tbody>
</table>

Modeling situation-aware conversations

- Understanding the situation and spatial-temporal context of a person to decide the conversation strategy is a key characteristic of the human conversation process
  - Continuously learning from the conversation history of the user provides a scope for chatbots to learn user’s conversation profile, e.g.,
    - what time of a day the user generally likes to talk or remains busy
    - whether the user is in a meeting or not …
Outline

I. Lifelong and Continual Learning: An Introduction
II. Continuous Knowledge Learning during Conversation
III. Continual Language Learning and Grounding
IV. Open-Domain Dialogue Learning After Deployment
V. Continual Learning for Task-oriented Dialogue Systems
VI. Continual Learning of Conversational Skills
VII. Other Challenges & Summary
Other Challenges of CL in Dialogues

- **Dealing with Wrong Knowledge from Users**

  - How to deal with the issue of acquiring intentional or unintentional wrong knowledge from them?
  
  - Can be addressed through a cross-verification strategy.

  - 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**.
Other Challenges of CL in Dialogues

- **Revision of Knowledge**
  
  ✓ How to revise or correct the wrong knowledge once it is detected?
  
  ✓ Requires -
    - a **knowledge monitoring system**: detect **contradictions** in the knowledge base
    - a **knowledge revision method**: revise the wrong knowledge and also all the consequences inferred from it.
Other Challenges of CL in Dialogues

- **Learning New Task Completion Skills from Users**
  - Modern task-oriented chatbots are deployed with a finite set of task completion skills which they have been preprogrammed with to perform.
  - Can end users use natural language dialogues to program their own chatbots and endow them with new skills after deployment?
    - lead to personalization of virtual assistants.

Other Challenges of CL in Dialogues

- One-shot or few shot continual learning
  - The amount of ground-truth data that can be acquired during interaction with human users is often very small, one or a few.
  - To learn continually and effectively, we need one-shot or few-shot continual learning methods.
    - Current methods are still very weak.

- In general, the current deep learning-based Continual Learning methods still have serious catastrophic forgetting problems.
  - Not ready for real-world applications. Some engineering hacks or data augmentations will be needed to get around of it.

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 (Liu, 2020; Liu and Mazumder, 2021)
  - The agent becomes smarter and smarter

- **Current techniques are still in their infancy**, but
  - Some methods are ready for practical applications.
Reading List [non-exhaustive]

- **Lifelong and Continual Learning: Introduction**
  - Sebastian Thrun. *Is learning the n-th thing any easier than learning the first?* In NIPS, 1996.
  - Geli Fei, Shuai Wang, and Bing Liu. *Learning Cumulatively to Become More Knowledgeable.* In KDD-2016
  - Zixuan Ke, Bing Liu, and Xingchang Huang. *Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks.* In NeurIPS 2020
Reading List [non-exhaustive]

**On the job Continual Learning**

**Continuous Knowledge Learning during Conversation**
Reading List [non-exhaustive]

- Continuous Knowledge Learning during Conversation (contd..)
  - Sahisnu Mazumder, Bing Liu, Shuai Wang, and Nianzu Ma. *Lifelong and Interactive Learning of Factual Knowledge in Dialogues*. In SIGDIAL-2019
Reading List [non-exhaustive]

- **Continuous Knowledge Learning during Conversation (contd..)**

- **Continual Language Learning and Grounding**
Reading List [non-exhaustive]

- **Continual Language Learning and Grounding (Contd..)**

- **Open-Domain Dialogue Learning**
Reading List [non-exhaustive]

- **Continual Learning for Task-oriented Dialogue Systems**
Reading List

- **Conversational Skill Learning**
Thank You

Q&A