Neural Conversational AI

Ondřej Dušek
MLSS^N Summer School
30 June 2022
About

Ondřej

• Charles University, Prague
• ‘16-18 at Heriot-Watt Uni Edinburgh
• working mostly on language generation
• often in/with dialogue systems

This lecture

• relatively vague/high-level (focus on main ideas)
• focusing on what I work with (pretrained language models)
• trying to avoid digressions
• expecting you know NNs, but haven’t necessarily worked in NLP
• probably much more applied than other talks here
  • most of you probably know more about ML theory than I do
• slightly improvised (depending on timing, I might skip stuff)
1. Intro: “Conversational AI” = “Dialogue Systems”

2. Transformer & pretrained language models

3. Neural models for dialogue system components
   • language understanding
   • state tracking
   • dialogue policy

4. End-to-end neural models

5. Evaluation metrics
1. Introduction
What’s Conversational AI = Dialogue System?

• Definition: A *(spoken)* dialogue system is a **computer system designed to interact** with users in *(spoken)* **natural language**
  • Wide – covers lots of different cases
    • “smart speakers” / phone OS assistants
    • phone hotline systems (even tone-dial ones)
    • in-car systems
    • assistive technologies: therapy, elderly care, companions
    • entertainment: video game NPCs, chatbots

• DSs are cool:
  • ultimate natural interface: say what you want
  • lots of active research – far from solved
  • already used commercially

Real-life dialogue systems: virtual assistants

- Google, Amazon, Apple & others, Mycroft, Rhasspy: open-source
- Really good microphones
  - and not much else – listen for wake word, processing happens online
- Huge knowledge bases
  - combined with web search
- Lots of domains programmed in, but all by hand
  - integration with a lot of services
    (calendar, music, shopping, weather, news…)
  - you can add your own (with limitations)
- Can keep some context
- Conversational capabilities limited


https://homealarmreport.com/smart-home/amazon-echo-vs-google-home/
Dialogue System Types

Task-oriented
• focused on completing a certain task/tasks
  • booking restaurants/flights, finding bus schedules, smart home…
• most actual DS in the wild
  • also our main focus in this course
• (typically) **single/multi domain**
  • talk about 1/more topics

Non-task-oriented
• chitchat – social conversation, entertainment
  • persona, gaming the Turing test
• typically **open-domain** – talk about anything

Comm. Modes: voice / text / multimodal (face, graphics…)
• traditional DS pipeline:
  • ASR: voice $\rightarrow$ text
  • NLU: text $\rightarrow$ meaning
  • DM: meaning $\rightarrow$ reaction
  • NLG: reaction $\rightarrow$ text
  • TTS: text $\rightarrow$ voice
• backend
  • needed for anything better than basic chit-chat
• text-based systems (here): NLU$\rightarrow$DM$\rightarrow$NLG

Dialogue Systems Architecture
Task-oriented Dialogue Example

MultiWOZ: benchmark for task-oriented dialogue in multiple domains
- hotels, restaurants, attractions, trains, taxi, police, hospital
- domains are connected (e.g. taxi from hotel to attraction)
- 10k dialogues, extensive annotation (but noisy!)

1. user: I am looking for a train from Cambridge to London Kings Cross.
   state: {train {departure = cambridge, destination = london kings cross}}
   DB: {train (70) [...]}
   system: There are 70 trains departing from Cambridge to London Kings Cross. What day would you like to travel?

2. user: I would like to leave on Saturday after 18:45.
   state: {train {day = saturday, departure = cambridge, destination = london kings cross, leave at = 18:45}}
   DB: {train (3) {arrive by = 19:51,21:51,23:51; id = TR0427,TR0925,TR4898; leave at = 19:00,21:00,23:00; … }}
   system: TR0427 leaves at 19:00 and arrives by 19:51. Would you like to book this train?

3. user: Yes, I would like to book it for eight people.
   system: I have booked it for you. Your reference number is 00000057. Is there anything else I can help you with?

4. user: I am also looking for an expensive restaurant in the centre.
   belief: {restaurant {area = centre, price range = expensive} train {…}}
   DB: {restaurant (33) {area = centre (33); name=Curry Garden, …; …}, …}
   system: There are 33 expensive restaurants in the centre. Is there a particular type of food you would like?
2. Transformer & Pretrained Models
Representing Language: Embeddings

• distributed representation
  • each word = a vector of floats
  • basically an easy conversion of 1-hot → numeric
  • a dictionary of trainable features

• part of network parameters – trained
  a) pretraining (optional)
  b) training for the target task

• the network learns which words are used similarly – for the given task
  • they end up having close embedding values
  • different embeddings for different tasks

• embedding size: ~100s-1000
• vocab size: ~50-100k
Subwords

- vocabulary is unlimited, embedding matrix isn’t
  - + the bigger the embedding matrix, the slower your models
- Special **out-of-vocabulary token** `<unk>`
  - loses information, we don’t want it on the output
- **Subwords**: groups of characters that
  - make shorter sequences than using individual characters
  - cover everything
  - 20-50k subwords for 1 language, ~250k subwords multilingual
- **Byte-pair Encoding** (=one way to get subwords)
  - start from individual characters
  - iteratively merge most frequent bigram, until you get desired # of subwords

(Sennrich et al., 2016)  
https://www.aclweb.org/anthology/P16-1162/  
(Kudo, 2018)  
https://aclanthology.org/P18-1007
Encoder-Decoder Networks (Sequence-to-sequence)

- Default RNN paradigm for sequences/structure prediction
  - **encoder** RNN: encodes the input token-by-token into **hidden states** $h_t$
    - next step: last hidden state + next token as input
  - **decoder** RNN: constructs the output token-by-token **autoregressively**
    - initialized by last encoder hidden state
    - output: hidden state & softmax over output vocabulary + argmax
    - next step: last hidden state + last generated token as input
- LSTM/GRU cells=layers over vectors of $\sim$ embedding size
- used for many NLP tasks

$h_0 = 0$
$h_t = \text{cell}(x_t, h_{t-1})$
$s_0 = h_T$
$p(y_t|y_1, \ldots y_{t-1}, x) = \text{softmax}(s_t)$
$s_t = \text{cell}(y_{t-1}, s_{t-1})$

https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129
Attention

• Encoder-decoder is too crude for complex sequences
  • the whole input is crammed into a fixed-size vector (last hidden state)

• **Attention** = “memory” of all encoder hidden states
  • weighted combination, re-weighted for every decoder step
    → can focus on currently important part of input
  • fed into decoder inputs + decoder softmax layer

• **Self-attention** – over previous decoder steps
  • increases consistency when generating long sequences

https://skymind.ai/wiki/attention-mechanism-memory-network
token representation: **embeddings**
- vectors of ~100-1000 numbers

source “word” embeddings

vocabulary is numbered

source “word” embeddings

encoder outputs
- “hidden states”
- vectors of numbers

attention = weighted combination
- weights different for each step

probability distribution over the whole vocabulary

target word embeddings

**cells**: identical (compound) neural layers
input: prev. output + token embedding

Transformer

(Waswani et al., 2017)
https://arxiv.org/abs/1706.03762

• getting rid of recurrences
  • faster to train, allows bigger nets
  • replace everything with **attention** + **feed-forward** networks
  • ⇒ needs more layers
  • ⇒ needs to encode positions

• positional encoding
  • adding position-dependent patterns to the input

• attention – simple dot-product
  • scaled by $\frac{1}{\sqrt{\text{#dims}}}$ (so values don’t get too big)
  • **more heads** (attentions in parallel)
    – focus on multiple inputs

\[
\sin\left(\frac{pos \times 10000}{2 \cdot \text{#dims}}\right) \quad \cos\left(\frac{pos \times 10000}{2 \cdot \text{#dims}}\right)
\]

one of these for each word

Transformer

- feed-forward (fully connected) network
- ReLU activations
- tricks for better training

**attention** over all of input

Positional encoding (indicate position in sentence)

**no recurrent connections**

**encoder**

**decoder**

http://arxiv.org/abs/1706.03762

(Vaswani et al., 2017)
Pretrained Language Models

• Transformer Architecture
  • Encoder-only (= good for classification/token tagging)
  • Decoder-only (= good for generation)
  • Encoder-Decoder (= RNN seq2seq equivalent)

• Self-supervised pretraining
  • standard supervised training, but without annotation
    • naturally occurring labels
    • automatic labels ~ fix artificially corrupted data
  • typically simple language tasks (→)
  • used with huge amounts of data – many GBs of text (e.g. CommonCrawl)
  • models not useful for much, but can be finetuned for the target task
    • just train further, use data for target task
Pretrained Language Models

• Pretraining Tasks
  • Masked word prediction
  • Next-word prediction
  • Fixing corrupt sentences
  • Sentence order prediction

• Models
  • BERT encoder only, variants: multilingual, RoBERTa (optimized)
  • GPT(-2/-3/-j/-neo): decoder only, next-word prediction
  • (m)BART, (m)T5: encoder-decoder
  • ByT5: enc-dec, byte-level (instead of subwords)

• a lot of pretrained models released plug-and-play
  • you only need to finetune (and sometimes, not even that)

(Devlin et al., 2019) https://www.aclweb.org/anthology/N19-1423
https://github.com/google-research/bert
(Liu et al., 2019) http://arxiv.org/abs/1907.11692
(Lewis et al., 2020) http://arxiv.org/abs/1910.13461

https://github.com/huggingface/transformers

(Devlin et al., 2019) https://openai.com/blog/better-language-models/
(Xue et al., 2022) https://doi.org/10.1162/tacl_a_00461
3. Component Models
Natural/Spoken Language understanding (NLU/SLU)

- **Words → meaning**: Extracting the meaning from user utterance
- **dialogue acts** (or other structured semantic representation):
  - act type/intent (*inform, request, confirm*)
  - slot/attribute (*price, time...*)
  - value (*11:34, cheap, city center...*)
  - typically intent classification + slot-value tagging
  - (other, more complex representations – e.g. trees, predicate logic)

- Specific steps:
  - **named entity resolution** (NER)
    - identifying task-relevant names (*London, Saturday*)
  - **coreference resolution**
    - (“it” → “the restaurant”)
NLU Challenges

• non-grammaticality  
  find something cheap for kids should be allowed

• disfluencies
  • hesitations – pauses, fillers, repetitions  
    uhm I want something in the west the west part of town
  • fragments
  • self-repairs (~6%!)  
    uhm I’m looking for a cheap
    uhm find something uhm something cheap no I mean moderate

• ASR errors  
  I’m looking for a for a chip Chinese rest or rant

• synonymy  
  Chinese city centre
  I’ve been wondering if you could find me a restaurant that has Chinese food close to the city centre please

• out-of-domain utterances  
  oh yeah I’ve heard about that place my son was there last month
NLU basics

- You can get far with keywords/regexes (for a limited domain)
- **Intent classification**
  - RNN: last hidden state
  - Transformers, PLMs: typically over 1st input element (start-of-sentence token)
- **Slot value detection**
  - classification (binary: “is slot value X present?”)
  - **slot tagging** – classify every token
    - BIO/IOB scheme: beginning (+slot) – inside (+slot) – outside
- **Delexicalization**: replacing slot values by placeholders
  - essentially named entity recognition
  - essentially tagging, but typically done by dictionaries

I'm looking for a Japanese restaurant in Notting Hill.  
I'm looking for a &lt;food&gt; restaurant in &lt;area&gt;.

I need a flight from Boston to New York tomorrow.

I need to leave after 12:00.

I need to leave after &lt;time&gt;.

(= not necessarily 1:1 with slots)
BERT-based NLU

- combined intent-slot
- slot tagging on top of pretrained BERT
  - standard **IOB approach**
  - feed last BERT layers to **softmax over tags**
    - classify only at 1st subword in case of split words (don’t want tag changes mid-word)
- special start token tagged with intent
  - again, softmax on top of last BERT layer
- finetune both tasks at once
  - essentially same task, just having different labels on the 1st token 😊

(Chen et al., 2019)
http://arxiv.org/abs/1902.10909
Dialogue Pretrained Models

- Pretraining on dialogue tasks can do better (& smaller) than BERT
  - ConveRT: Transformer-based **dual encoder**
    - 2 Transformer encoders: context + response
    - feed forward + cosine similarity on top
  - training objective: **response selection**
    - response that actually happened = 1
    - random response from another dialogue = 0
  - trained on a large dialogue dataset (Reddit)
- can be used as a base to train models for:
  - **slot tagging** (top self-attention layer → CNN → CRF)
  - **intent classification** (top feed-forward → more feed-forward → softmax)
  - Transformer layers are fixed, not fine-tuned
  - works well for little training data (**few-shot**)
• pre-finetuning BERT on vast task-oriented dialogue data  
  • basically combination of 2 previous approaches

• **BERT + user/sys tokens +** train for:  
  • masked language modelling  
  • response selection (dual encoder style)  
    • over [CLS] tokens from whole batch  
    • other examples in batch = negative

• result: “better dialogue BERT”  
  • can be finetuned for various dialogue tasks  
    • intent classification  
    • slot tagging  
  • good performance even few-shot  
    • just 1 or 10 examples per class

(Wu et al., 2020)  
https://www.aclanthology.org/2020.emnlp-main.66
Dialogue Manager (DM)

• Given NLU input & dialogue so far, responsible for **deciding on next action**
  • keeps track of what has been said in the dialogue
  • keeps track of user profile
  • interacts with backend (database, internet services)

• Dialogue so far = **dialogue history**, modelled by **dialogue state**
  • managed by **dialogue state tracker**

• System actions decided by **dialogue policy**
Dialogue state / State tracking

• Stores (a summary of) dialogue history
  • User requests + information they provided so far
  • Information requested & provided by the system
  • User preferences

• Implementation
  • **handcrafted** – e.g. replace value for slot with last-mentioned
    • good enough in some circumstances
  • **probabilistic (belief state)**
    – keep an estimate of per-slot preferences based on NLU
      • more robust, more complex
      • accumulates probability over time & n-best lists
      • → handles NLU/ASR errors
        – e.g. 3x same low-confidence input = prob. high enough to react

| price: | cheap
| food:  | Chinese
| area:  | riverside
| price: | 0.8 cheap
|        | 0.1 moderate
|        | 0.1 <null>
| food:  | 0.7 Chinese
|        | 0.3 Vietnamese
| area:  | 0.5 riverside
|        | 0.3 <null>
|        | 0.2 city center
a) **Always trust the NLU**

    for null value:
    \[ p = \text{prev} \cdot p(\text{null}) \sim \text{user didn’t mention this slot} \]

    **non-null** value \( v \):
    \[ p = \text{prev} \cdot p(\text{null}) + p(v) \]
    \sim \text{didn’t mention = carry from previous}
    \sim \text{did mention = add new NLU probability}

    • basically rule-based (but good if NLU is good)

b) **“NLU” over whole dialogue**

    • typically classification (“is slot value \( v \) present?”)
      • option: limit to some candidates (from NLU/delexicalization), rank them
    • may not need NLU, may be better, but slower

---

(Žilka et al., 2013)
http://www.aclweb.org/anthology/W13-4070
BERT & Span Selection

a.k.a. Span Tagging
(≈ question answering/reading comprehension)

• BERT over previous system & current user utterance
• from 1st token’s representation, get a decision: none/dontcare/span
  • per-slot (BERT is shared, but the final decision is slot-specific)
• span = need to find a concrete value as a span somewhere in the text
  • predict start & end token of the span using 2 softmaxes over tokens
• rule-based update:
  • if none is predicted, keep previous value
  • essentially similar to NLU & update rule
Break
Action Selection / Policy

- Deciding what to do next
  - action based on the current belief state
  - following a policy (strategy) towards an end goal (e.g. book a flight)
  - controlling the coherence & flow of the dialogue
  - actions: linguistic & non-linguistic (backend access)
  - actions represented by system dialogue acts

- DM/policy should:
  - manage uncertainty from belief state
  - recognize & follow dialogue structure
  - plan actions ahead towards the goal

Did you say Indian or Italian?

follow convention, don’t be repetitive
e.g. ask for all information you require

confirm(food=Chinese)
inform(name=Golden Dragon, food=Chinese, price=cheap)

(from Milica Gašić’s slides)
Action Selection Approaches

- Finite-state machines
  - simplest possible
  - dialogue state is machine state

- Frame-based/flowcharts (e.g. VoiceXML)
  - slot-filling + providing information – basic agenda
  - rule-based in essence

- Rule-based
  - any kind of rules (e.g. Python code)

- Statistical
  - typically trained with reinforcement learning
Why Reinforcement Learning

- **Action selection ~ classification →** use supervised learning?
  - set of possible actions is known
  - belief state should provide all necessary features

- Yes, but…
  - You’d need sufficiently large **human-human data** – hard to get
    - human-machine would just mimic the original system
  - Dialogue is ambiguous & complex
    - there’s **no single correct next action** – multiple options may be equally good
    - but datasets will only have one next action
    - **some paths will be unexplored** in data, but you may encounter them
  - DSs won’t behave the same as people
    - ASR errors, limited NLU, limited environment model/actions
    - **DSs should behave differently** – make the best of what they have
  - supervised classification **doesn’t plan ahead**
    - RL optimizes for the whole dialogue, not just the immediate action
Reinforcement learning: Definition

- MDP formalism: agent in an environment, **state-action-reward**
  - Agent
  - Environment
  - State transition is stochastic → maximize expected return

- RL = finding a **policy that maximizes long-term reward**
  - unlike supervised learning, we don’t know if an action is good
  - immediate reward might be low while long-term reward high

\[
R_t = \sum_{t=0}^{T} \gamma^t r_{t+1}
\]

\(\gamma \in [0,1] = \text{discount factor}\)

(Immediate vs. future reward trade-off)

- state transition is stochastic → maximize **expected return**

\[E[R_t | \pi, s_0]\]

expected \(R_t\) if we start from state \(s_0\) and follow policy \(\pi\)
Policy Gradients

• Train a **network to represent the policy** $\pi(a|s, \theta)$ – $\theta$ are parameters

• To optimize, we need a **performance metric**: $J(\theta) = \mathbb{E}[R_t|\pi, s_0]$
  • expected return in starting state when following $\pi_\theta$
  • we want to directly optimize this using gradient ascent

• **Policy Gradient Theorem**:
  • expresses $\nabla J(\theta)$ in terms of $\nabla \pi(a|s, \theta)$

$$
\nabla J(\theta) \propto \sum_s \mu(s) \sum_a Q^\pi(s, a) \nabla \pi(a|s, \theta) = E_\pi \left[ \sum_a Q^\pi(s, a) \nabla \pi(a|s, \theta) \right]
$$

$\mu(s)$ is state probability under $\pi$ – this is the same as expected value $E_\pi$

$Q^\pi(s, a)$ = “Q-function”
  – value of taking action $a$ in state $s$, then following policy $\pi$

(Sutton & Barto, 2018; p. 324ff)
REINFORCE: Monte Carlo Policy Gradients

• direct search for policy parameters by stochastic gradient ascent
  • looking to maximize performance $J(\theta) = \mathbb{E}[R_t | \pi, s_0]$
• choose learning rate $\alpha$, initialize $\theta$ arbitrarily
• loop forever:
  • generate an episode $s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T$, following $\pi(\cdot | \cdot, \theta)$
  • for each $t = 0, 1, ..., T$: $\theta \leftarrow \theta + \alpha \gamma^t R_t \nabla \ln \pi(a_t | s_t, \theta)$

variant – **advantage** instead of returns:
  discounting a **baseline** $b(s)$ (predicted by any model)
  $A_t = R_t - b(s_t)$ instead of $R_t$
gives better performance

returns $R_t = \sum_{i=t}^{T-1} \gamma^{i-t}r_{i+1}$
this will guarantee the right state distribution/frequency $\mu(s)$
this is stochastic $\nabla J(\theta)$:
  • from policy gradient theorem
  • using single action sample $a_t$
  • expressing $Q^\pi$ as $R_t$ (under $E_\pi$)
  • using $\nabla \ln x = \frac{v_x}{x}$

(Sutton & Barto, 2018; p. 327f)
Rewards in RL

• Typical setup – **handcrafted rewards:**
  - every turn: -1 (encourage fast dialogues)
  - successful dialogue: + 20
  - unsuccessful: - 10 (~center around 0)

• Problems:
  - domain knowledge needed to detect dialogue success
  - **need simulated and/or paid users** (known goal)
    - simulated = essentially another dialogue system
    - paid users = costly + often fail to follow pre-set goals
  - needs a lot of dialogues to train (1000s) → simulated users, supervised pretraining

• Solutions:
  - trained rewards
    - provided by a network, can be turn-level
  - corpus-based RL (supervised/RL hybrid)
    - follow dataset, just assign rewards like RL (→)
• Representing system dialogue act in natural language (text)
  • reverse NLU

• How to express things might depend on context
  • Goals: fluency, naturalness, avoid repetition (…)

• Traditional approach: **templates**
  • Fill in (=**lexicalize**) values into predefined templates (sentence skeletons)
  • Works well for limited domains

```
inform(name=Golden Dragon, food=Chinese, price=cheap) +

<name> is a <price>-ly priced restaurant serving <food> food

= Golden Dragon is a cheaply priced restaurant serving Chinese food.
```

• Statistical approach: **seq2seq**/pretrained language models
  • input: system dialogue act, output: sentence (operation similar to →)
4. End-to-end models
End-to-End Systems

• experimental, research state-of-the-art
  • but not ready for practical deployment

• the whole system (NLU/DM/NLG) is a single neural network
  • joint training (“end-to-end”)  
  • more elegant
  • potentially easily retrainable

• typically still needs annotation
  • same as individual modules
  • can be less predictable

• connecting the database is a problem
  • typically this step is done separately

(Wen et al., 2017)
https://www.aclweb.org/anthology/E17-1042/
End-to-end vs. separate components

- Traditional architecture – separate components:
  - more flexible (replace one, keep the rest)
  - error accumulation
  - improved components don’t mean improved system
  - possibly joint optimization by RL
  - more explainable

- End-to-end:
  - joint supervised optimization, RL still works
  - still needs DA-level annotation
  - typically needs a lot of data
  - less control of outputs: hallucination, dull/repetitive

Neural Conv AI

Inform(food=Chinese)

Food: Chinese
Price: cheap
Area: ?

What area would you prefer?

Many results

How are you? I am good

I’m looking for a cheap Chinese place
Sequicity: Two-stage Copy Net

(Lei et al., 2018) [https://www.aclweb.org/anthology/P18-1133](https://www.aclweb.org/anthology/P18-1133)

- fully **RNN/seq2seq**-based, not much structure
  - still explicit dialogue state
  - DB is external (as in most systems)
- operation:
  1) **encode**
     - previous dialogue state
     - prev. system response
     - current user input
  2) **decode new dialogue state** first
     - attend over whole encoder
  3) **decode system output** (delexicalized)
     - attend over state only
       + use DB output (one-hot vector added to each generator input)
         - DB: 0/1/more results – vector of length 3
         - **delexicalized** decoding: use placeholders (replaced based on full DB result)
End-to-end Dialogue with GPT-2

- Multiple recent DSs are based on GPT-2 (SOLOIST, UBAR, SimpleTOD, NeuralPipeline)
  - decoder-only PLM

- Similar to Sequicity, everything recast as sequence generation
  - dialogue context, belief state, database outputs represented as sequences
  - GPT-2 **prompting**: force-decode some input (ignore softmaxes, feed your tokens)
    - allows attention over it, conditions following text
    - essentially works like an encoder

- Multi-step operation:
  1) prompt with context & decode belief state
  2) query DB (external)
  3) prompt with DB output & decode response

---

**SOLOIST**
(Peng et al., 2021)

1. **Task 1**: Belief State Prediction
   - User: I would like to find an ... of town
   - Belief State: Restaurant (...), [EOB]

2. **Task 2**: Grounded Response Generation
   - DB: ...<EOKB>
   - [The restaurant name] is a great ... [EOS]

3. **Task 3**: Contrastive Objective

---

**Neural Conv AI**
(Hosseini-Asl et al., 2020)
Yang et al., 2021

- [Link](https://www.aclweb.org/anthology/2020.acl-main.54)
- [Link](http://arxiv.org/abs/2012.03539)
• Same idea as ↑, multiple improvements

• Operation:
  1) context → belief state
     • prompt w. context & user utterance
     • greedy decoding of state
     • text-like belief state representation
  2) belief state → DB
     • text-like DB results
  3) DB → response
     • top-p sampling (diversity)
     • delexicalized (slot placeholders)

• Training:
  • belief/response prediction + consistency (Y/N)
Consistency task

• **Additional training task** – generating & classifying at the same time
  • additional classification layer on top of last decoder step logits
  • incurs additional loss, added to generation loss

• **Aim: robustness** – detecting problems
  • ½ data **artificially corrupted** – state or target response don’t fit context
  • prev. work: corrupted state sampled randomly
  • **AuGPT**: corrupted state sampled from the **same domain** – harder!

<table>
<thead>
<tr>
<th>context</th>
<th>state</th>
<th>response</th>
<th>consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>i want a cheap italian restaurant</td>
<td>{ price range = cheap , food = Italian }</td>
<td>ok which area ?</td>
<td>✅</td>
</tr>
<tr>
<td>i want a cheap Italian restaurant</td>
<td>{ price range = cheap , food = Italian }</td>
<td>thanks, goodbye !</td>
<td>❌ bad response</td>
</tr>
<tr>
<td>i want a cheap italian restaurant</td>
<td>{ destination = Cambridge , leave at = 19:00 }</td>
<td>ok which area ?</td>
<td>❌ bad state</td>
</tr>
<tr>
<td>i want a cheap italian restaurant</td>
<td>{ area = north , food = Chinese }</td>
<td>ok which area ?</td>
<td>❌ bad state (same domain)</td>
</tr>
</tbody>
</table>

(Kulhánek et al., 2021)
http://arxiv.org/abs/2102.05126
Further improvements

(Kulhánek et al., 2021)
http://arxiv.org/abs/2102.05126

• **Data augmentation** via backtranslation (en → xx → en)
  • MT between English and 40 languages from the ELITR project ([https://elitr.eu/](https://elitr.eu/))
  • we chose 10 best languages
  • user inputs chosen at random from original & 10 backtranslated texts

• **Data cleaning**
  • checking consistency of user goal with database
  • ~30% MultiWOZ data discarded

• **Unlikelihood loss** for output diversity
  • repeated tokens are penalized

• **Sampling** for output diversity
MinTL: Diff dialogue states

- 2-step decoding, same as ↑
  - based on T5 or BART here
  - explicit 2 decoders
    (for state, for response)

- “Levenshtein states”
  - don’t decode full state each time
  - just decode a diff
    (“Levenshtein distance from previous”)
  - better consistency over dialogue

Obtain diffs from state annotation

Encode previous state & context

DB queried based on updated state

Response decoder starting token = # of DB results

Neural Conv AI

(Lin et al., 2020)
Training end-to-end systems: RL?

• Supervised
  • sometimes components still trained separately
    • e.g. hard knowledge base lookup
  • sometimes all in one
  • can’t learn from users
  • problems with train-test mismatch

• RL
  • can learn from users, can learn all-in-one
  • doesn’t work great if done on word-level
    • RL doesn’t care about fluency/naturalness
    • either avoid word-level, or mix with supervised


Facebook abandoned an experiment after two artificially intelligent programs appeared to be chatting to each other in a strange language only they understood.
• Making system actions latent, learning them implicitly

• **Discrete latent space** here ($M$ $k$-way variables)
  • using Gumbel-Softmax trick for backpropagation
  • trained using Full ELBO (KL divergence vs. a prior network)
    or “Lite ELBO” (KL divergence vs. uniform)

• RL over latent actions, not words
  • avoids producing disfluent language
  • **corpus-based RL**
    • generate outputs, but use original contexts from a dialogue from training data
    • success & RL updates based on generated responses

• ignores DB & state tracking
  • takes gold annotation from data (assumes external model for this)
HDNO: Hierarchical RL End-to-end Dialogue

- Similar to (↑), but tries word-level RL
  - corpus-level RL
  - RNN architecture
  - dialogue state not tracked

- hierarchical RL:
  - **top level**: latent actions, like LARL
    - latent actions Gaussian here
    - standard reward based on success
  - **bottom level**: words
    - reward based on fluency
    - language model probability
  - both rewards weighted (word level much lower)
  - levels updated asynchronously

(Wang et al., 2021)
5. Evaluation
Corpus-based evaluation
(Nekvinda & Dušek, 2021)
https://aclanthology.org/2021.gem-1.4

• Task: take real dialogue history from corpus + generate 1 response
  • repeat over whole dialogue, collect responses

• Metrics:
  • Inform rate – last offered entity matches user constraints
  • Success rate – ↑ + system provided all requested information about it
  • Joint goal accuracy – % turns where all user constraints are captured correctly
  • BLEU – n-gram precision (matching sub-phrases of 1-4 words against reference)

• Problems:
  • really artificial setting, but easiest to use (just need test data)
  • Inf/Succ/JGA: matching the provided entities (more ways to do it)
  • BLEU: tokenization, measuring over delexicalized text
Simulator Evaluation

• **User Simulator** – works as a user, tries to follow goals

• **Dialogue-level** – good over 1 turn ≠ good over whole dialogue
  - especially for end-to-end systems, errors may accumulate over time
  - simulator is the only automatic way to assess this

• Main metric: **Success rate**: was the simulated user’s goal reached?
  - i.e. did the system give a correct entity & all information
  - technically same as corpus-based, but now over real dialogues

• Problems:
  - the simulator needs to be built for a given domain
  - it’s essentially another dialogue system (👤 × 🐔)
  - simulator behavior will bias the evaluation
Human Evaluation

Metrics (objective – measuring):

• **Task success** (boolean): did the user get what they wanted?
  • (paid) testers with known goal → check if they found what they were supposed to
    • [warning] sometimes people go off script
  • basic check: did we provide any information at all?

• **Duration**: number of turns (fewer is better)

Metrics (subjective – questionnaires):

• **Success rate**: Did you get all the information you wanted?
  • typically different from objective measures!

• **Future use**: Would you use the system again?

• Component-specific questions

---

<table>
<thead>
<tr>
<th>System</th>
<th># calls</th>
<th>Subjective Success Rate</th>
<th>Objective Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDC</td>
<td>627</td>
<td>82.30% (±2.99)</td>
<td>62.36% (±3.81)</td>
</tr>
<tr>
<td>NBC</td>
<td>573</td>
<td>84.47% (±2.97)</td>
<td>63.53% (±3.95)</td>
</tr>
<tr>
<td>NAC</td>
<td>588</td>
<td>89.63% (±2.46)</td>
<td>66.84% (±3.79)</td>
</tr>
<tr>
<td>NABC</td>
<td>566</td>
<td>90.28% (±2.44)</td>
<td>65.55% (±3.91)</td>
</tr>
</tbody>
</table>

(Jurčiček et al., 2012)
https://doi.org/10.1016/j.csl.2011.09.004
Final Remarks
Further Research Areas

• Multi/open domains
  • reusability, domain transfer
  • training from little data
  • using less annotation
  • connecting task-oriented systems and chatbots

• Context dependency
  • understand/reply in context (grounding, speaker alignment)

• Incrementality
  • don’t wait for the whole sentence to start processing

• Evaluation
  • neural-net-based metrics
Multimodal/Visual Dialogue

- adding other modalities
- specific components
  - parallel to NLU
    - vision – image classification networks
    - face identification/tracking
  - parallel to NLG
    - mimics/gesture generation
    - gaze
    - image retrieval
  - vision – typically CNN
    - often off-the-shelf stuff
  - specific classifiers/rules

(Agarwal et al., 2018)

http://aclweb.org/anthology/W18-6514

https://youtu.be/5fhjuGu3d0I?t=137
https://vimeo.com/248025147

Neural Conv AI
Thanks

Contact me:
MLSS^N Slack
in person till tomorrow
odusek@ufal.mff.cuni.cz

I’m looking for a postdoc
& will be looking for PhD students
(know someone?)
http://ufal.cz/ng-nlg/postdoc

Get the slides here:
http://ufal.cz/ondrej-dusek/bibliography (under “Talks”)

References/Inspiration/Further:
Apart from materials referred directly, these slides are based on slides and syllabi by:

- Pierre Lison (Oslo University): https://www.uio.no/studier/emner/matnat/ifi/INF5820/h14/timeplan/index.html
- Oliver Lemon & Verena Rieser (Heriot-Watt University): https://sites.google.com/site/olemon/conversational-agents
- Milica Gašić (University of Cambridge): http://mi.eng.cam.ac.uk/~mg436/teaching.html
- David DeVault & David Traum (Uni. of Southern California): http://projects.ict.usc.edu/nld/cs599s13/schedule.php
- Luděk Bártĕk (Masaryk University Brno): https://is.muni.cz/el/1433/jaro2018/PA156/um/
- Gina-Anne Levow (University of Washington): https://courses.washington.edu/ling575/
Recommended Reading

Best:

- McTear: Conversational AI. Morgan & Claypool 2021. ([https://doi.org/10.2200/S01060ED1V01Y202010HLT048](https://doi.org/10.2200/S01060ED1V01Y202010HLT048)) – a bit more advanced & focused, pretty new
- Sutton & Barto: Reinforcement Learning: An Introduction, 2018 (freely online) – specifically on RL, pretty advanced
- recent papers from the field (linked on individual slides)

Also good (but more outdated):