NPFL123 Dialogue Systems
7. Neural Policies & Natural Language Generation

https://ufal.cz/npfl123

Ondřej Dušek, Patrizia Schmidtova, Vojtech Hudecek & Jan Cuřín
27.3.2023
Deep Reinforcement Learning

- Exactly the same as “plain” RL (see last time)
  - agent & environment, actions & rewards
  - Markov Decision Process
- “deep” = part of the agent is handled by a NN
  - value function (typically $Q$)
  - policy
- NN = function approximation approach
  - such as REINFORCE / policy gradients
  - NN → complex non-linear functions
- assuming huge state space
  - much fewer weights than possible states
  - update based on one state changes many states

(Sutton & Barto, 2018)
Value Function Approximation

• Searching for approximate $V(s)$ or $Q(s, a)$
  • exact values are too big to enumerate in a table
  • **parametric approximation** $V(s; \theta)$ or $Q(s, a; \theta)$

• Regression: **Mean squared value error**
  • weighted over states’ importance
  • useful for gradient descent
  • $\rightarrow \sim$ any supervised learning approach possible
    • not all work well though

• MC = stochastic gradient descent

• TD is not true gradient descent
  • $\leftarrow$ using current weights in target estimate
  • faster than MC, but unstable for NNs!

\[
\overline{VE}(\theta) := \sum_{s \in S} \mu(s) \left( V_\pi(s) - V(s, \theta) \right)^2
\]

states’ importance weight (probability distribution) $\sim$ how likely each state is

**target value** (which we don’t have!)

$\rightarrow$ using $R_t$ in MC

$\rightarrow$ using $r_{t+1} + \gamma V(s', \theta)$ in TD
Deep Q-Networks

- Q-learning with function approximation
  - $Q$ function represented by a neural net

- Causes of poor convergence in basic Q-learning with NNs:
  a) SGD is unstable
  b) correlated samples (data is sequential)
  c) TD updates aim at a moving target (using $Q$ in computing updates to $Q$)
  d) scale of rewards & $Q$ values unknown $\rightarrow$ numeric instability

- Fixes in DQN:
  a) minibatches (updates by averaged $n$ samples, not just one)
  b) **experience replay**
  c) **freezing target Q function**
  d) clipping rewards

---

(Mnih et al., 2013, 2015)
http://arxiv.org/abs/1312.5602
http://www.nature.com/articles/nature14236

common NN tricks
**DQN tricks ~ making it more like supervised learning**

- **Experience replay** – break correlated samples
  - run through some episodes (dialogues, games…)
  - store all tuples \((s, a, r', s')\) in a buffer
  - for training, don’t update based on most recent moves – use buffer
    - sample minibatches randomly from the buffer
    - overwrite buffer as you go, clear buffer once in a while
  - only possible for off-policy

\[
\text{loss} := \mathbb{E}_{(s,a,r',s') \in \text{buf}} \left[ (r' + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta))^2 \right]
\]

- **Target Q function freezing**
  - fix the version of Q function used in update targets
    - have a copy of your Q network that doesn’t get updated every time
    - once in a while, copy your current estimate over

“generate your own ‘supervised’ training data”

“have a fixed target, like in supervised learning”
DQN algorithm

- initialize $\theta$ randomly
- initialize replay memory $D$ (e.g. play for a while using current $Q(\theta)$)
- repeat over all episodes:
  - for episode, set initial state $s$
    - select action $a$ from $\epsilon$-greedy policy based on $Q(\theta)$
    - take $a$, observe reward $r'$ and new state $s'$
    - store $(s, a, r', s')$ in $D$
  - $s \leftarrow s'$
- often → once every $k$ steps:
  - sample a batch $B$ of random $(s, a, r', s')$’s from $D$
  - update $\theta$ using loss $\mathbb{E}_{(s, a, r', s') \in B} \left[ (r' + \gamma \max_{a'} Q(s', a'; \bar{\theta}) - Q(s, a; \theta))^2 \right]$ |
- rarely → once every $\lambda$ steps:
  - $\bar{\theta} \leftarrow \theta$

(Mnih et al., 2013, 2015)
http://arxiv.org/abs/1312.5602
http://www.nature.com/articles/nature14236
https://youtu.be/V1eYniJ0Rnk?t=18
• a simple DQN can drive a dialogue system’s action selection
  • DQN is function approximation – works fine for POMDPs
  • no summary space tricks needed here

rule-based simulator with agenda running on DA level

DQN – feed-forward, 1 hidden ReLU layer

error model controller (simulating ASR/NLU noise)

movie ticket booking: better than rule-based

replay memory initialized using a simple handcrafted policy

NPFL123 L7 2022

Policy Networks

- Learning policy directly – **policy network**
  - can work better than Q-learning
  - NN: input = state, output = prob. dist. over actions
  - extension – actor-critic: network predicts both $\pi$ and $V/Q$

- Training can’t use/doesn’t need the DQN tricks
  - just **REINFORCE** with baseline
    - reward – baseline = **advantage**
    - baseline $\sim$ e.g. 0 (if reward symmetric) or better use $V$
  - these are on-policy $\rightarrow$ no experience replay
    - minibatches used anyway

---

**policy gradient theorem guarantees convergence**
Natural Language Generation

• conversion of **system action semantics → text** (in our case)

• NLG output is well-defined, but input is not:
  • DAs
  • any other semantic formalism
  • database tables
  • raw data streams
  • user model
  • dialogue history
  
  can be any kind of knowledge representation
  
  e.g. “user wants short answers”
  e.g. for referring expressions, avoiding repetition

• general NLG objective:
  • **given input & communication goal**
  • **create accurate + natural, well-formed, human-like text**

• additional NLG desired properties:
  • variation
  • simplicity
  • adaptability
NLG Use Cases

• dialogue systems
  • very different for task/non-task-oriented/QA systems

• standalone
  • data-to-text
  • short text generation for web & apps
    • weather, sports reports
    • personalized letters
    • creative generation (stories)

• machine translation
  • now mostly integrated end-to-end
  • formerly not the case

• summarization
Inputs
- **Content/text/document planning**
  - content selection according to communication goal
  - basic structuring & ordering

Content plan
- **Sentence planning/microplanning**
  - aggregation (facts \(\rightarrow\) sentences)
  - lexical choice
  - referring expressions (e.g., restaurant vs. it)

Sentence plan
- **Surface realization**
  - linearization according to grammar
  - word order, morphology

Text

Deciding what to say
Deciding how to say it

Typically handled by dialogue manager in dialogue systems
Organizing content into sentences & merging simple sentences
This is needed for NLG in dialogue systems
• Few systems implement the whole pipeline
  • All stages: mostly domain-specific data-to-text, standalone
    • e.g. weather reports
  • Dialogue systems: just sentence planning + realization
  • Systems focused on content + sentence planning with trivial realization
    • frequent in DS: focus on sentence planning, trivial or off-the-shelf realizer
  • Surface realization only
    • requires very detailed input
    • some systems: just ordering words

• Pipeline vs. end-to-end approaches
  • planning + realization in one go – popular for neural approaches
  • pipeline: simpler components, might be reusable (especially realizers)
  • end-to-end: no error accumulation, no intermediate data structures
NLG Basic Approaches

• **canned text**
  • most trivial – completely hand-written prompts, no variation
  • doesn’t scale (good for DTMF phone systems)

• **templates**
  • “fill in blanks” approach
  • simple, but much more expressive – covers most common domains nicely
  • can scale if done right, still laborious
  • most production dialogue systems

• **grammars & rules**
  • grammars: mostly older research systems, realization
  • rules: mostly content & sentence planning

• **machine learning**
  • modern research systems
  • pre-neural attempts often combined with rules/grammar
  • neural nets made it work *much* better
Template-based NLG

- Most common in dialogue systems
  - especially commercial systems
- Simple, straightforward, reliable
  - custom-tailored for the domain
  - complete control of the generated content
- Lacks generality and variation
  - difficult to maintain, expensive to scale up
- Can be enhanced with rules
  - e.g. articles, inflection of the filled-in phrases
  - template coverage/selection rules, e.g.:
    - select most concrete template
    - cover input with as few templates as possible
    - random variation

(Facebook, 2015)

Template coverage/selection rules, e.g.:

- select most concrete template
- cover input with as few templates as possible
- random variation

(Facebook, 2019)

- inflection rules

(Alex public transport information rules)

https://github.com/UFAL-DSG/alex
Grammar/Rules for Sentence Planning

- Handcrafted grammar/rules
  - input: base semantics (e.g. dialogue acts)
  - output: detailed sentence representation (=realizer inputs, see \(\rightarrow\))

- Statistical enhancements:
  generate more options & choose the best
  - generate multiple outputs
    - underspecified grammar
    - rules with multiple options...
  - choose the best one
    - train just the selection – learning to rank
    - any supervised approach possible e.g. “best” = 1, “not best” = 0

NB: this is slow!

SpoT trainable planner (RankBoost ranking)

(Walker et al., 2001)
https://www.aclweb.org/anthology/N01-1003
Grammar-based realizers

- Various grammar formalisms
  - production / unification rules in the grammar
  - lexicons to go with it
  - expect very detailed input (sentence plans)
- typically general-domain, reusable
  - **KPML** – multilingual
    - systemic functional grammar
  - **FUF/SURGE** – English
    - functional unification grammar
  - **OpenCCG** – English
    - combinatory categorial grammar

---

**KPML input for** *A dog is in the park.*

![](https://www.academia.edu/download/3459017/bateman97-jnle.pdf)

**FUF/SURGE input for** *She hands the draft to the editor*

![](https://www.aclweb.org/anthology/W03-2316)

**OpenCCG input for** *The cheapest flight is on Ryanair*

**Procedural realizers**

- **SimpleNLG** – no grammar, code to build sentence
  - “do-it-yourself” style – only cares about the grammar
  - system then linearizes
  - built for English, ports to other languages available

- **RealPro** (Meaning-Text-Theory)
  - deep syntax/semantics → surface syntax → morphology

- **Treex** (Functional Generative Description)
  - deep syntax → surface syntax → morphology, linearization
  - Perl code operating over syntax trees

---

(Gatt & Reiter, 2009)
https://www.aclweb.org/anthology/W09-0613
(Lavoie & Rambow, 1997)
http://dl.acm.org/citation.cfm?id=974596
(Popel & Žabokrtský 2010; Dušek et al., 2015)
https://www.aclweb.org/anthology/W15-3009
Trainable Realizers

• **Overgenerate & Rerank**
  • same approach as for sentence planning
  • assuming a flexible handcrafted realizer (e.g., OpenCCG)
  • underspecified input → more outputs possible
  • generate more & use statistical reranker, based on:
    • n-gram language models
    • Tree language models
    • expected text-to-speech output quality
    • personality traits & alignment/entrainment
  • more variance, but at computational cost

• **Grammar/Procedural-based**
  • same as RealPro or TectoMT, but predict each step using a classifier

**References**
- NITROGEN (Langkilde & Knight, 1998) [https://www.aclweb.org/anthology/P98-1116](https://www.aclweb.org/anthology/P98-1116)
- HALOGEN (Langkilde-Geary, 2002) [https://www.aclweb.org/anthology/W02-2103](https://www.aclweb.org/anthology/W02-2103)
- FERGUS (Bangalore & Rambow, 2000) [https://aclweb.org/anthology/C00-1007](https://aclweb.org/anthology/C00-1007)
- CRAG (Isard et al., 2006) [https://www.aclweb.org/anthology/W06-1405](https://www.aclweb.org/anthology/W06-1405)
- StuMaBa (Bohnet et al., 2010) [https://www.aclweb.org/anthology/C10-1012](https://www.aclweb.org/anthology/C10-1012)
- HALOGEN (Langkilde & Knight, 1998) [https://www.aclweb.org/anthology/P98-1116](https://www.aclweb.org/anthology/P98-1116)
- NITROGEN (Langkilde & Knight, 1998) [https://www.aclweb.org/anthology/P98-1116](https://www.aclweb.org/anthology/P98-1116)
- HALOGEN (Langkilde-Geary, 2002) [https://www.aclweb.org/anthology/W02-2103](https://www.aclweb.org/anthology/W02-2103)
- FERGUS (Bangalore & Rambow, 2000) [https://aclweb.org/anthology/C00-1007](https://aclweb.org/anthology/C00-1007)
- CRAG (Isard et al., 2006) [https://www.aclweb.org/anthology/W06-1405](https://www.aclweb.org/anthology/W06-1405)
- StuMaBa (Bohnet et al., 2010) [https://www.aclweb.org/anthology/C10-1012](https://www.aclweb.org/anthology/C10-1012)
Non-Neural End-to-End NLG

• NLG as language models
  • hierarchy of language models (HMM/MEMM/CRF style)
  • DA → slot → word level

• NLG using context-free grammars
  a) “language models” by probabilistic CFGs
     • approximate search for best CFG derivation
  b) synchronous PCFGs – MRs & text
     • “translation” with hierarchical phrase-based system
     • parsing MR & generating text

---

<table>
<thead>
<tr>
<th>Rule</th>
<th>prob./parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. S → R(start)</td>
<td></td>
</tr>
<tr>
<td>2. R(r,t) → FS(r,t,start) R(r,t)</td>
<td>[Pr = 1]</td>
</tr>
<tr>
<td>3. R(r,t) → FS(r,t,start)</td>
<td>[Pr(r,t) \cdot \lambda]</td>
</tr>
<tr>
<td>4. FS(r,t,f) → F(r,t,f) FS(r,t,f)</td>
<td>[Pr(f</td>
</tr>
<tr>
<td>5. FS(r,t,f) → F(r,t,f)</td>
<td>[Pr(f</td>
</tr>
<tr>
<td>6. F(r,t,f) → W(r,t,f) F(r,t,f)</td>
<td>[Pr(w</td>
</tr>
<tr>
<td>7. F(r,t,f) → W(r,t,f)</td>
<td>[Pr(w</td>
</tr>
<tr>
<td>8. W(r,t,f) → (\alpha)</td>
<td>[Pr[\alpha</td>
</tr>
<tr>
<td>9. W(r,t,f) → (g(f,v))</td>
<td>[Pr[g(f,v)</td>
</tr>
</tbody>
</table>

(a) English
(b) CLANG


(Oh & Rudnicky, 2002) https://doi.org/10.1016/S0885-2308(02)00012-8
(Angei et al., 2010) https://www.aclweb.org/anthology/D10-1049
(Liang et al., 2009) https://www.aclweb.org/anthology/P09-1011
(Mairesse et al., 2010) https://www.aclweb.org/anthology/P10-1157
(Mairesse & Young, 2014) https://www.aclweb.org/anthology/J14-4003
(Konstas & Lapata, 2012) https://www.aclweb.org/anthology/P12-1039
Neural Generation: Seq2seq RNNs (see NLU for RNN intro)

- **token representation**: *embeddings* = vectors of ~100-1000 numbers
- **source “word” embeddings**: “hidden states” (=again, vectors of numbers)
- **encoder outputs**: 0, 1, 2, 3, 4, 5, 6, etc.
- **attention**: weighted combination (weights different for each step)
- **encoder**: cell = identical (compound) neural layers
  - input: prev. output + token embedding
- **decoder**: cell
- **target word embeddings**: 0, 1, 2, 3, 4, 5, 6, etc.
- **probability distribution over the whole vocabulary**
- **vocabulary is numbered**
Neural End-to-End NLG: RNNs

• Unlike previous, doesn’t need alignments
  • no need to know which word/phrase corresponds to which slot

  name [Loch Fyne], eatType[restaurant], food[Japanese], price[cheap], familyFriendly[yes]

  Loch Fyne is a kid-friendly restaurant serving cheap Japanese food.

• 1st system: RNN language model conditioned on DA (~decoder only)
  • input: binary-encoded DA
    • 1 if intent/slot-value present, 0 if not
    • delexicalized: much fewer values, shorter vector
  • modified LSTM cells
    • input DA passed in every time step
  • generating delexicalized texts word-by-word
    • i.e. decoder only

Seq2seq NLG with reranking (TGen)

- Encode DAs as sequences, apply standard RNN seq2seq
  - encoder: triples <DA type, slot, value>
  - decoder: words (possibly delexicalized)
- Beam search & reranking
  - DA classification of outputs
  - checking against input DA

(Dušek & Jurčiček, 2016)
https://aclweb.org/anthology/P16-2008
Transformer = seq2seq, with feed-forward & attention nets (instead of RNN)

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

attention over all of input

encoder

decoder

no recurrent connections

attention over all of input & output generated so far (self-attention)

(Vaswani et al., 2017) http://arxiv.org/abs/1706.03762
Transformers & Pretrained Language Models

  - encoder-decoder, but using feed-forward & attention instead of RNNs
  - positional encoding used to indicate sentence position
    - predefined “pattern” functions (based on sin & cos)
    - simply added to word embeddings
  - no RNN → parallel training → faster, allows larger models (more layers)

- **Pretrained language models** – on large data w/o annotation (self-supervised)
  - guess masked word (encoder only: BERT) (Devlin et al., 2019) [https://www.aclweb.org/anthology/N19-1423](https://www.aclweb.org/anthology/N19-1423)
  - generate next word (decoder only: GPTx) (Radford et al., 2019) [https://openai.com/blog/better-language-models/](https://openai.com/blog/better-language-models/)

- Can be **finetuned** for your domain & task
  - less data than w/o pretraining, extremely fluent (Chen et al., 2020) [https://www.aclweb.org/anthology/2020.acl-main.18/](https://www.aclweb.org/anthology/2020.acl-main.18/)
Problems with neural NLG

• Checking the **semantics**
  • neural models tend to forget input / make up irrelevant stuff
  • reranking / decoding changes work, but not perfectly
  • generally **hard to control** (especially LLMs)

• Needs quite a lot of data (except LLMs, with prompting)

• Delexicalization needed (at least some slots)
  • typically OK for pretrained LMs

• Diversity & complexity of outputs
  • still can’t match humans
  • needs specific tricks to improve this

• Still more hassle than writing up templates 😊

(Dušek et al., 2020)
http://arxiv.org/abs/1901.07931

---

open sets, verbatim on the output
(e.g., restaurant/area names)
Summary

Deep Reinforcement Learning

• same as plain RL – agent + states, actions, rewards – just $Q$ or $\pi$ is a NN
  • function approximation for $Q$ – mean squared value error

  • **Deep Q Networks** – Q learning where $Q$ is a NN + tricks
    • experience replay, target function freezing

  • **Policy networks** – policy gradients where $\pi$ is a NN

Natural Language Generation

• steps: content planning, **sentence planning**, **surface realization**
  • not all systems implement everything (content planning is DM’s job in DS)
  • pipeline vs. end-to-end

• approaches: templates, grammars, statistical

• **templates** work great

• neural: **RNN** / **Transformer**, pretrained models
Contact us:

https://ufaldsg.slack.com/
{odusek,schmidtova,hudecek}@ufal.mff.cuni.cz
Skype/Meet/Zoom (by agreement)

Get these slides here:

http://ufal.cz/npfl123

References/Inspiration/Further:

- David Silver’s course on RL (UCL): http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Milan Straka’s course on RL (Charles University): http://ufal.mff.cuni.cz/courses/npfl122/