Neural Conversational AI

Ondřej Dušek, Petr Schwarz, Ondřej Plátek, Santosh Kesiraju

JSALT Summer School

20 June 2022

Ondrej Platek is supported by
Charles University project GAUK 40222
& EU H2020 project no. 952026, HumanE-AI-Net

Topics

1. Intro: “Conversational AI” = “Dialogue Systems”
2. Neural network-based language models, Transformer, Pretrained Models
3. Neural models for dialogue system components
   - language understanding
   - state tracking
   - dialogue policy
4. End-to-end neural models

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

https://www.digitaltrends.com/mobile/5-things-you-need-to-know-about-microsofts-chinese-girlfriend-chatbot-xiaoice/

Real-life dialogue systems: virtual assistants

- Google, Amazon, Apple & others, Mycroft, Rhasspy: open-source
- The devices are really good microphones (microphone arrays)
  - 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://www.lifehacker.com.au/2018/02/specs-showdown-google-home-vs-amazon-echo-vs-apple-homepod/

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 dialog systems in the wild
  - also our main focus
- (typically) **single/multi domain**
  - talk about 1/more topics

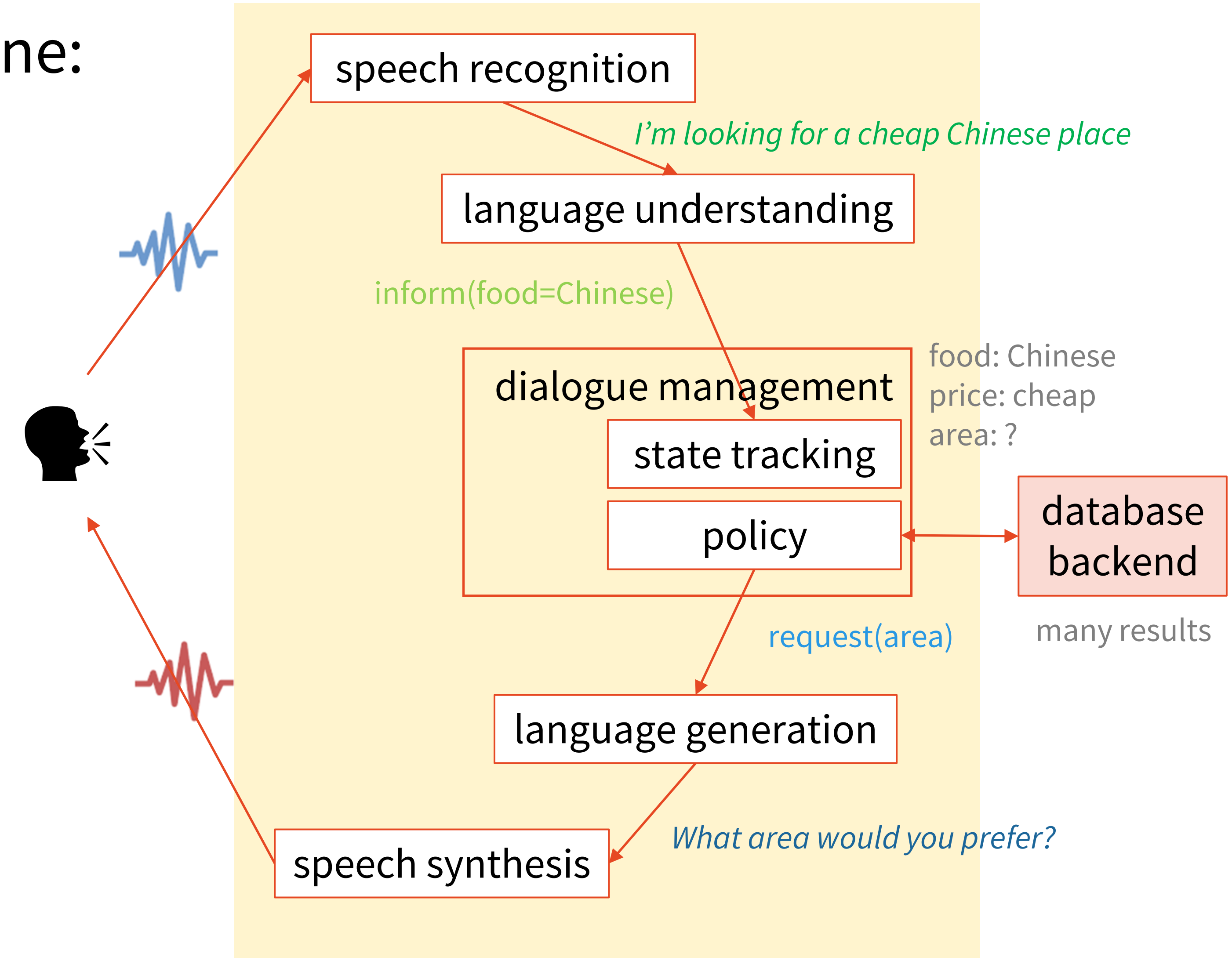
Non-task-oriented

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

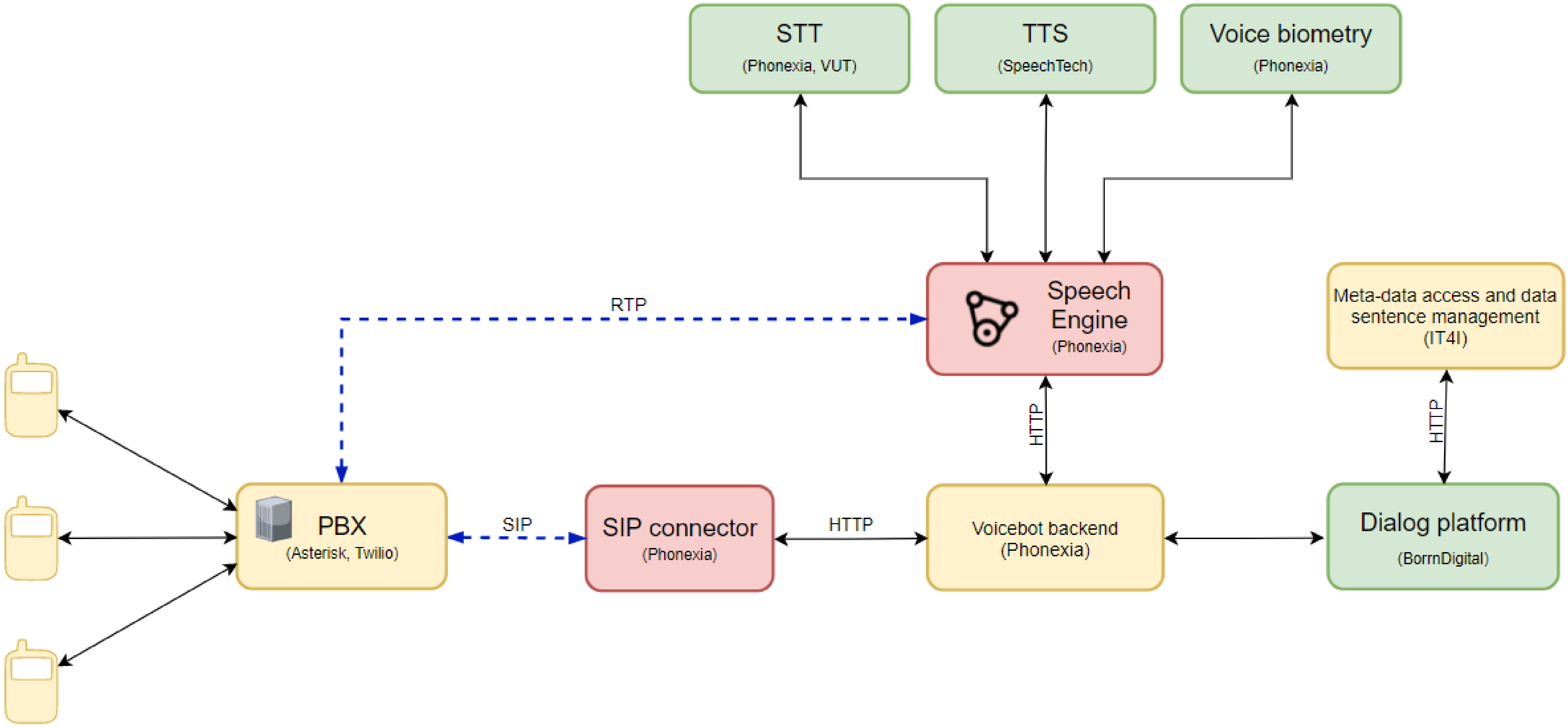
Modalities: voice / text / multimodal (face, graphics…)

Dialogue Systems Architecture

- traditional dialog system pipeline:
  - ASR: voice → text
  - **NLU: text → meaning**
  - **DM: meaning → reaction**
  - **NLG: reaction → text**
  - TTS: text → voice
- backend
  - needed for anything better than basic chit-chat
- text-based systems (here): NLU→DM→NLG



Voice dialog system (project with emergency lines)



Task-oriented Dialogue Example

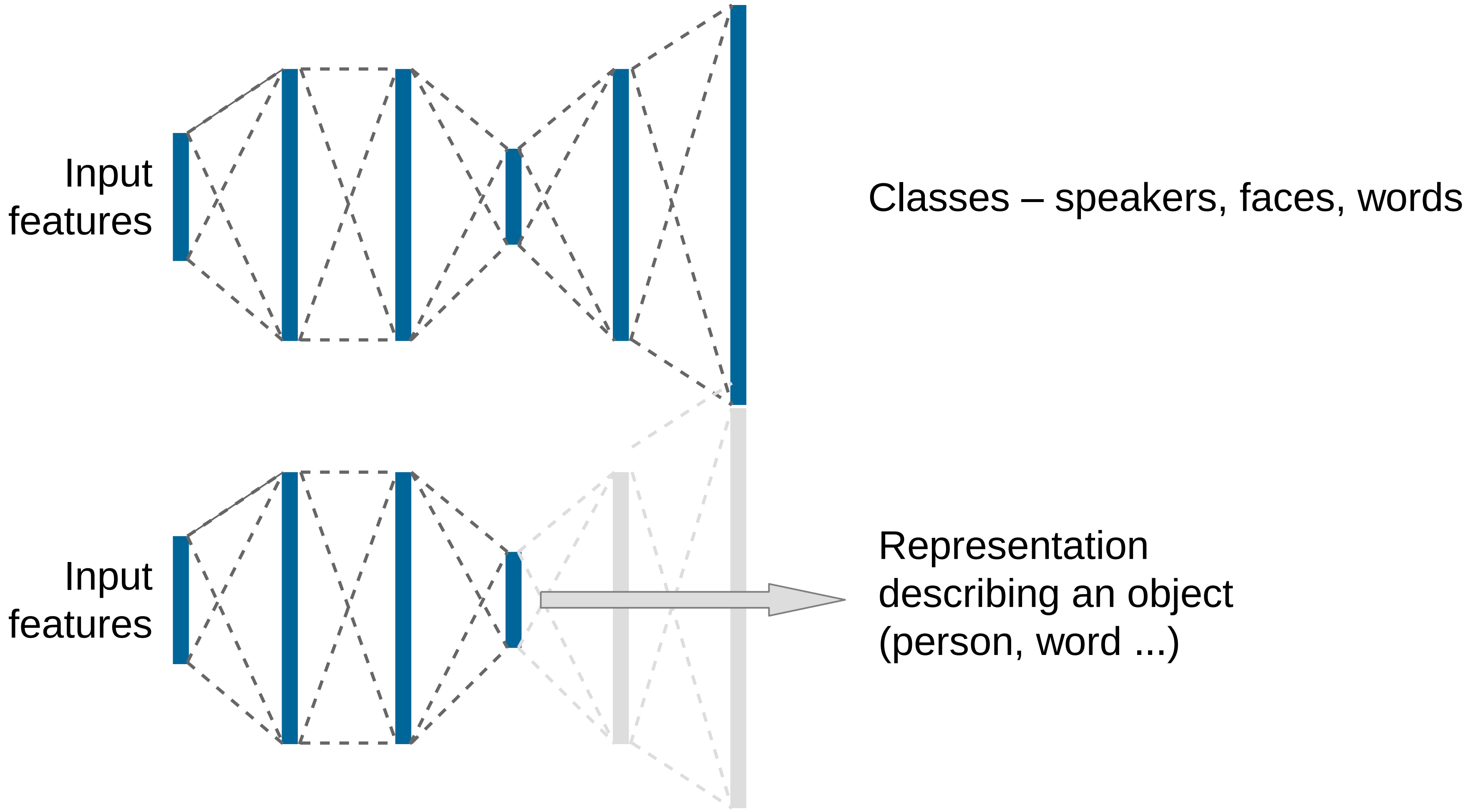
(Budzianowski et al., 2018)
https://aclanthology.org/D18-1547/

- 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!)

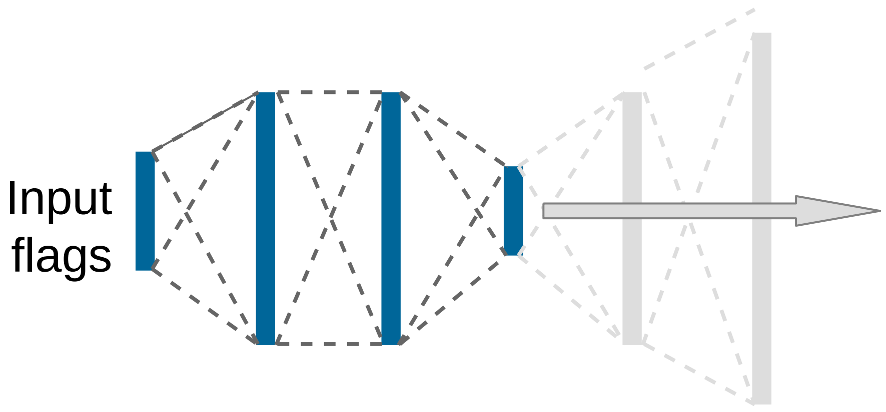
1.	user:	I am looking for a train from Cambridge to London Kinks Cross.
	state:	*{train {departure = cambridge, destination = london kings cross}}*
	DB:	*{train (70) {…}}*
		[count] **[departure]** **[destination]**
	system:	There are 70 trains departing from Cambridge to London Kings Cross. What day 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; … }}*
		[id] **[leave_at]** **[arrive_by]**
	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.
		[reference]
	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, ...; ...}, ...}*
		[count] **[price_range]** **[area]**
	system:	There are 33 expensive restaurants in the centre. Is there a particular type of food you would like?

2. Neural network-based language models, Transformer, Pretrained Models

Neural network embedding

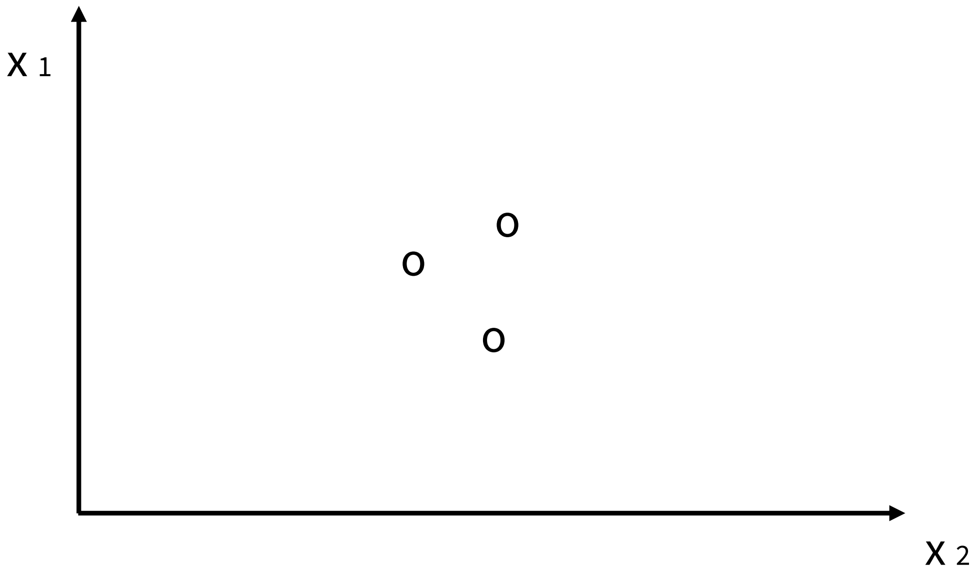


Embedding can be seen as dot in vector space



Representation describing an object (speaker, word...)

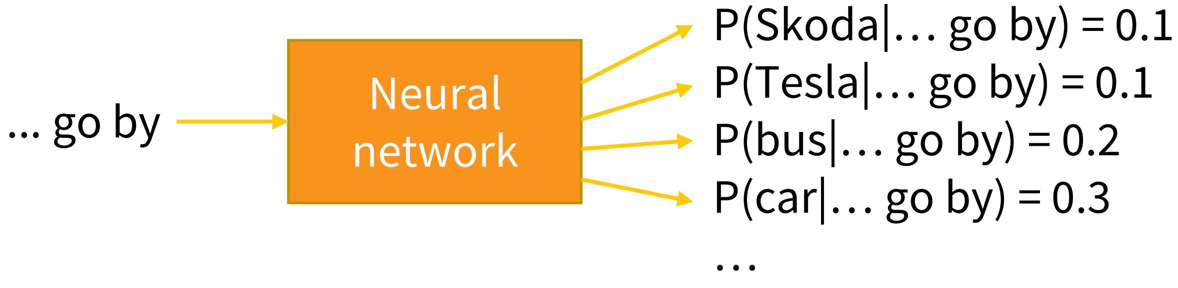
$$\boldsymbol{x} = \begin{bmatrix} 1.8364 \\ 2.5534 \\ -0.5327 \\ \cdots \\ 1.1397 \end{bmatrix}$$

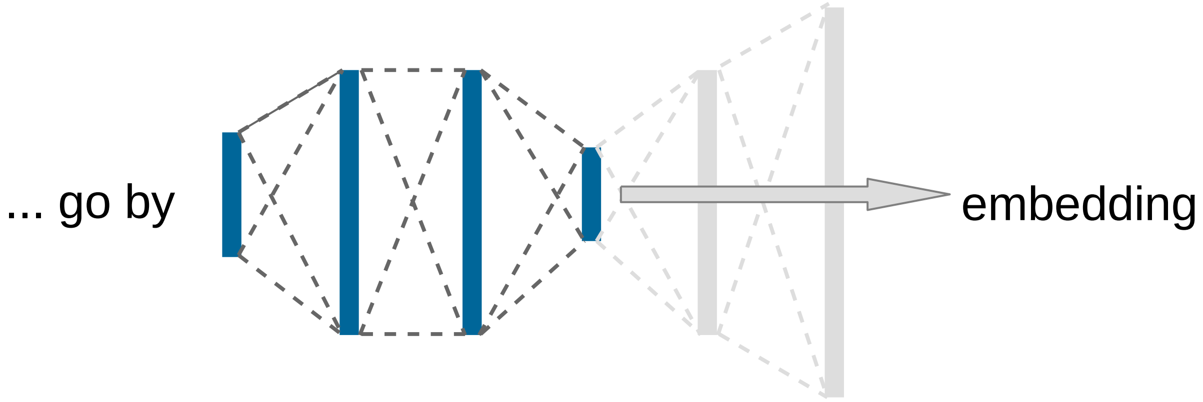


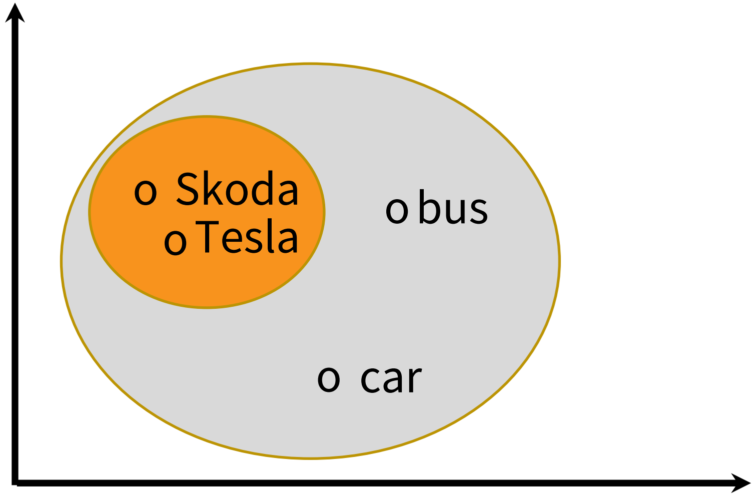
Neural network-based language models

- A neural network can be trained to predict the next word in a sentence (Tomáš Mikolov BUT/CIIRC)
- Input can be 1-hot encoding (1 at the index of word, else 0)

I go by Skoda
I go by Tesla
I go by bus
I go by car
I go to a service by Tesla

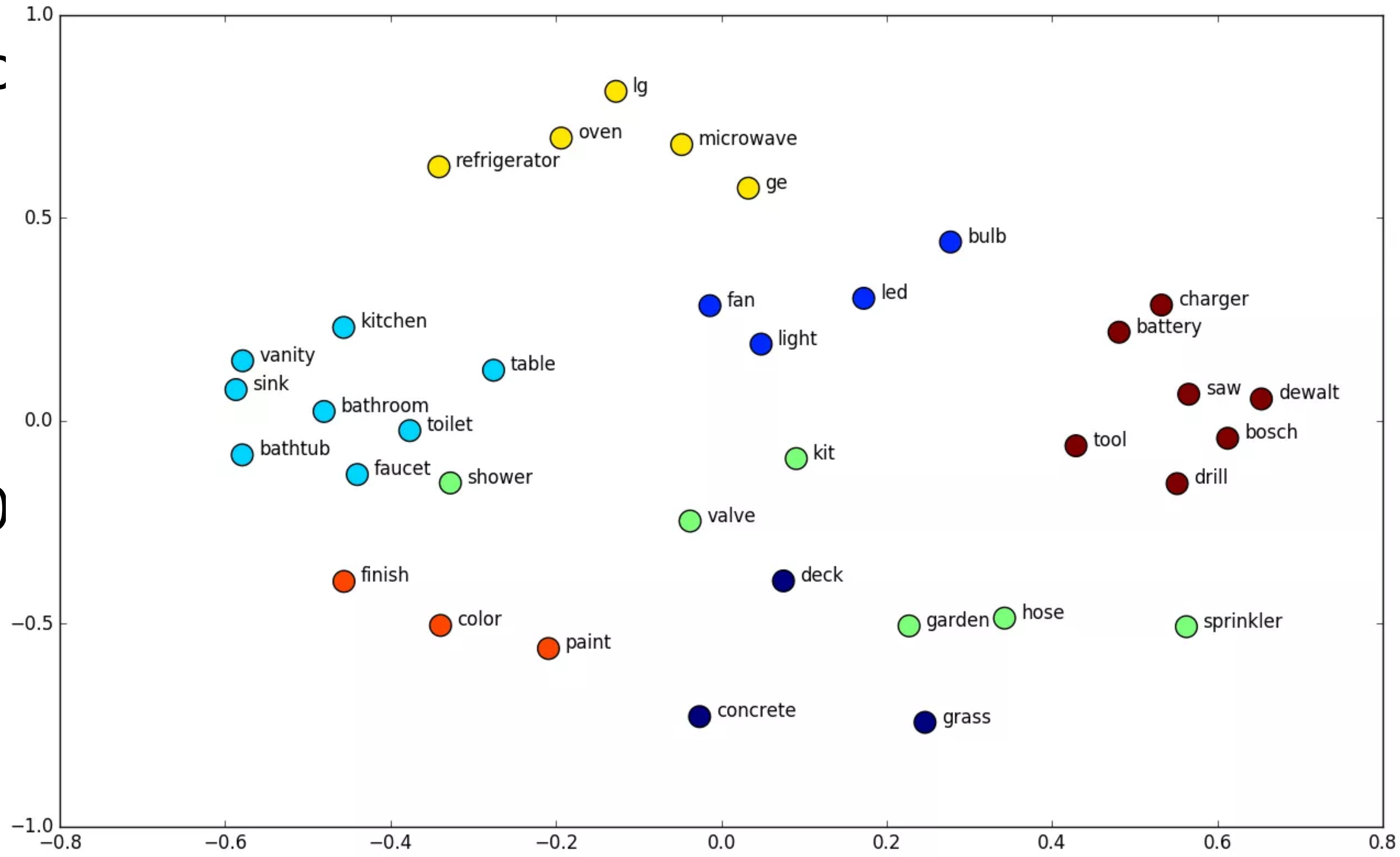






Representing Language: Embeddings

- the network learns which words are used similarly – fc the given task
  - they end up having close embedding values
  - different embeddings for different tasks
- embedding size: ~100s-1000
- vocab size: ~50-100k



http://blog.kaggle.com/2016/05/18/home-depot-product-search-relevance-winners-interview-1st-place-alex-andreas-nurlan/

Subword units

- vocabulary is unlimited, our word list (and one-hot encoding vector dimension) is not
  - + the bigger the dimension is, the sparser and the slower the model is
- 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
- Another possibilities: **Word-pieces, Characters**

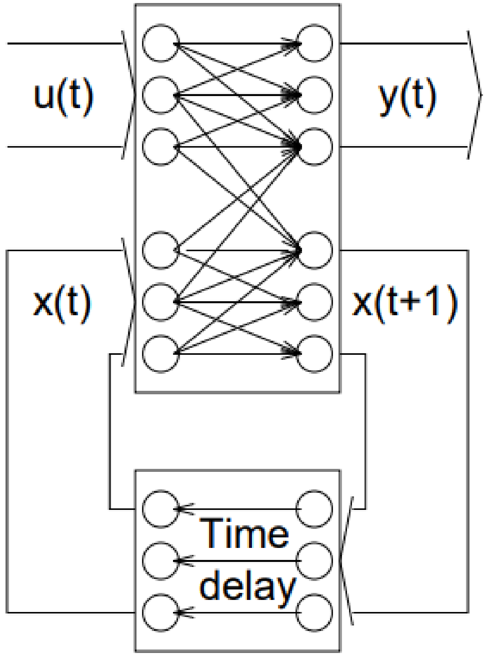
f a s t _	*fast er _*
f a s t e r _	*tall er _*
t a l l _	*s l o w er _*
t a l l e r _	*tall e s t _*

(Sennrich et al., 2016) https://www.aclweb.org/anthology/P16-1162/

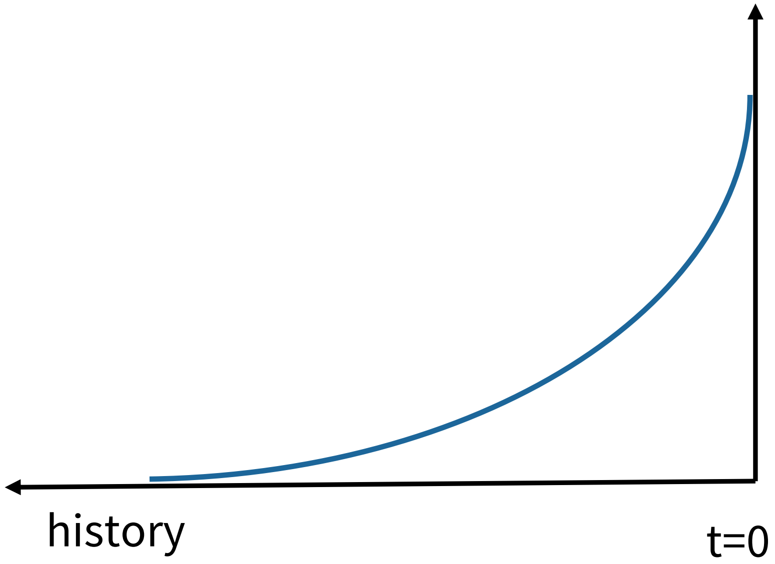
(Kudo, 2018) https://aclanthology.org/P18-1007

Neural networks and word context

- Recurrent Neural Network (RNN)
  - We would like to model arbitrary long word history
  - Problem with Gradient Vanishing during training

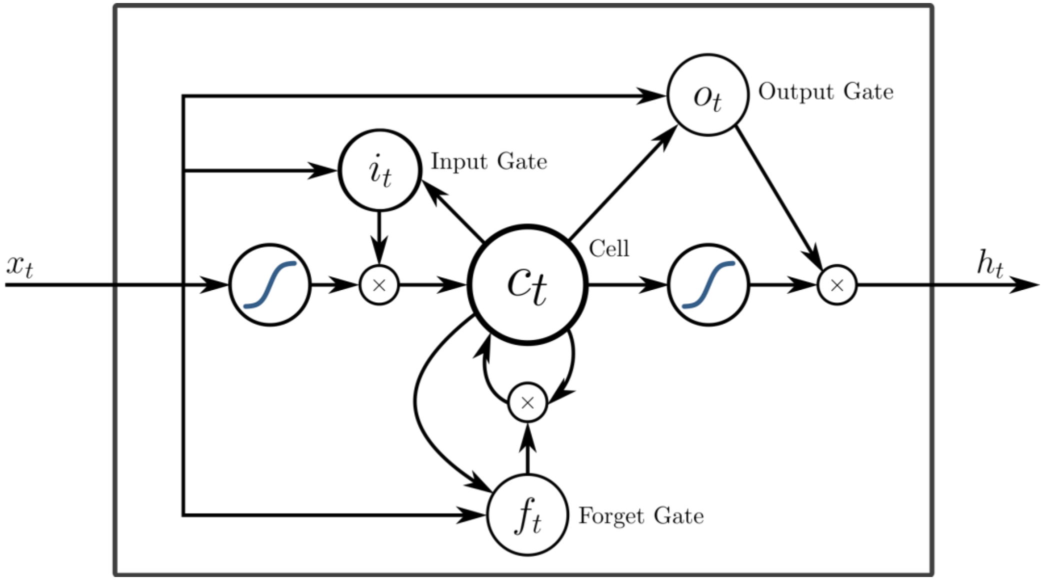


How is the context information seen?



Neural networks and word context

- Long short term memory (LSTM)
  - Can work better with longer histories
  - The stored information is controlled by gates

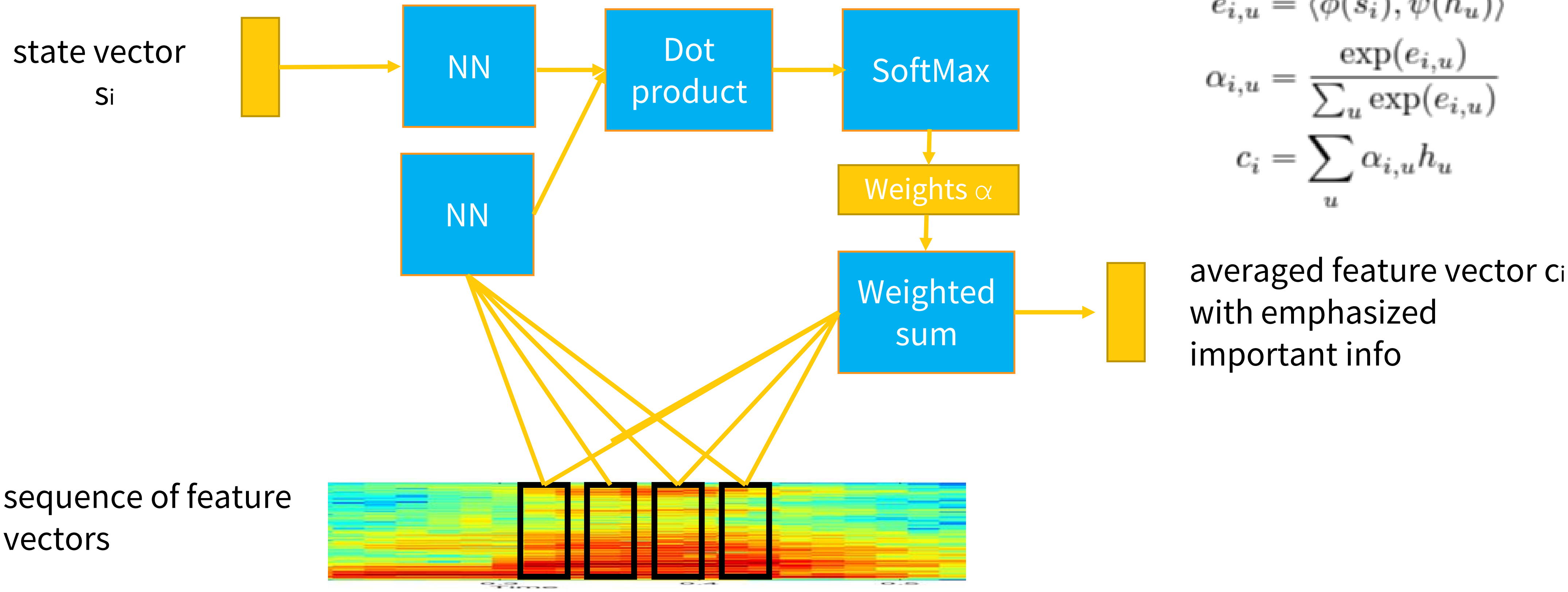


$$
\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}
$$

- B-LSTM – adds also a run in the opposite direction (from future to past)

Neural networks and word context

- Can be seen as an evolution of LSTMs
- Attention is a mechanism that enables us to focus on arbitrary place in the time (can be input sequence of features, output sequence, or both)



$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}$$

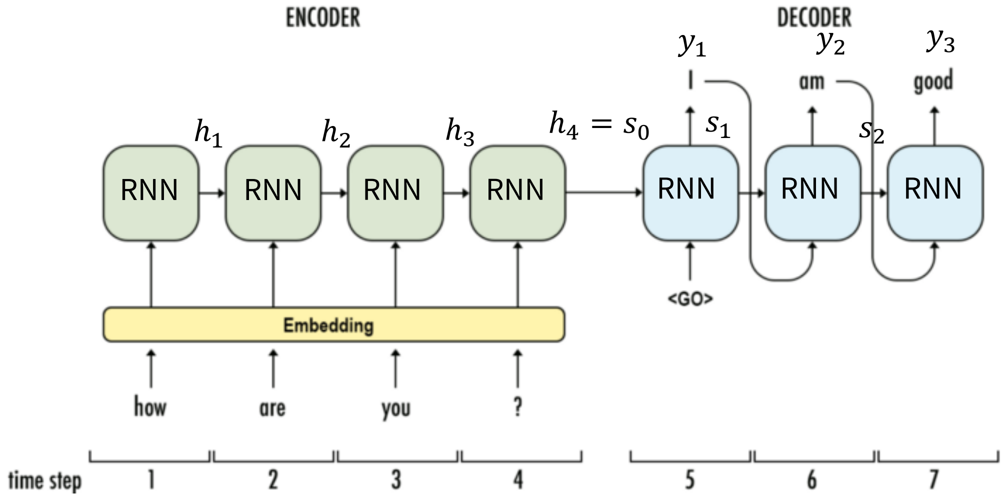
$$c_i = \sum_u \alpha_{i,u} h_u$$

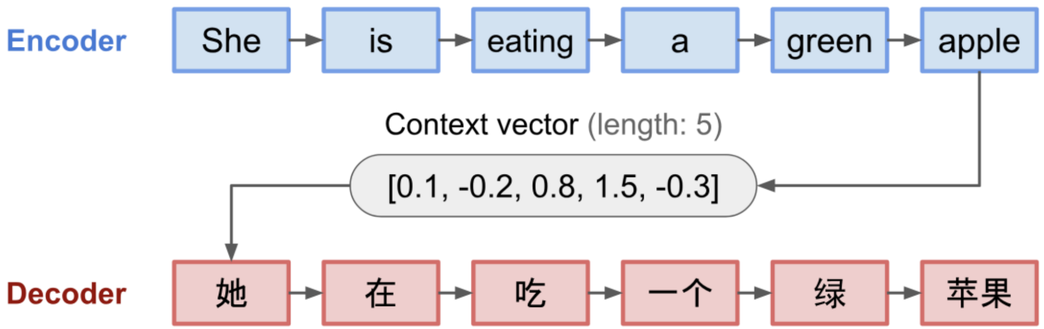
Encoder-Decoder Networks (Sequence-to-sequence)

- 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 ~ embedding size
  - used for many NLP tasks

$$\boldsymbol{h}_0 = \mathbf{0}$$
$$\boldsymbol{h}_t = \mathrm{cell}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{s}_0 = \boldsymbol{h}_T$$
$$p(y_t | y_1, \dots y_{t-1}, \mathbf{x}) = \mathrm{softmax}(\boldsymbol{s}_t)$$
$$\boldsymbol{s}_t = \mathrm{cell}(\boldsymbol{y}_{t-1}, \boldsymbol{s}_{t-1})$$

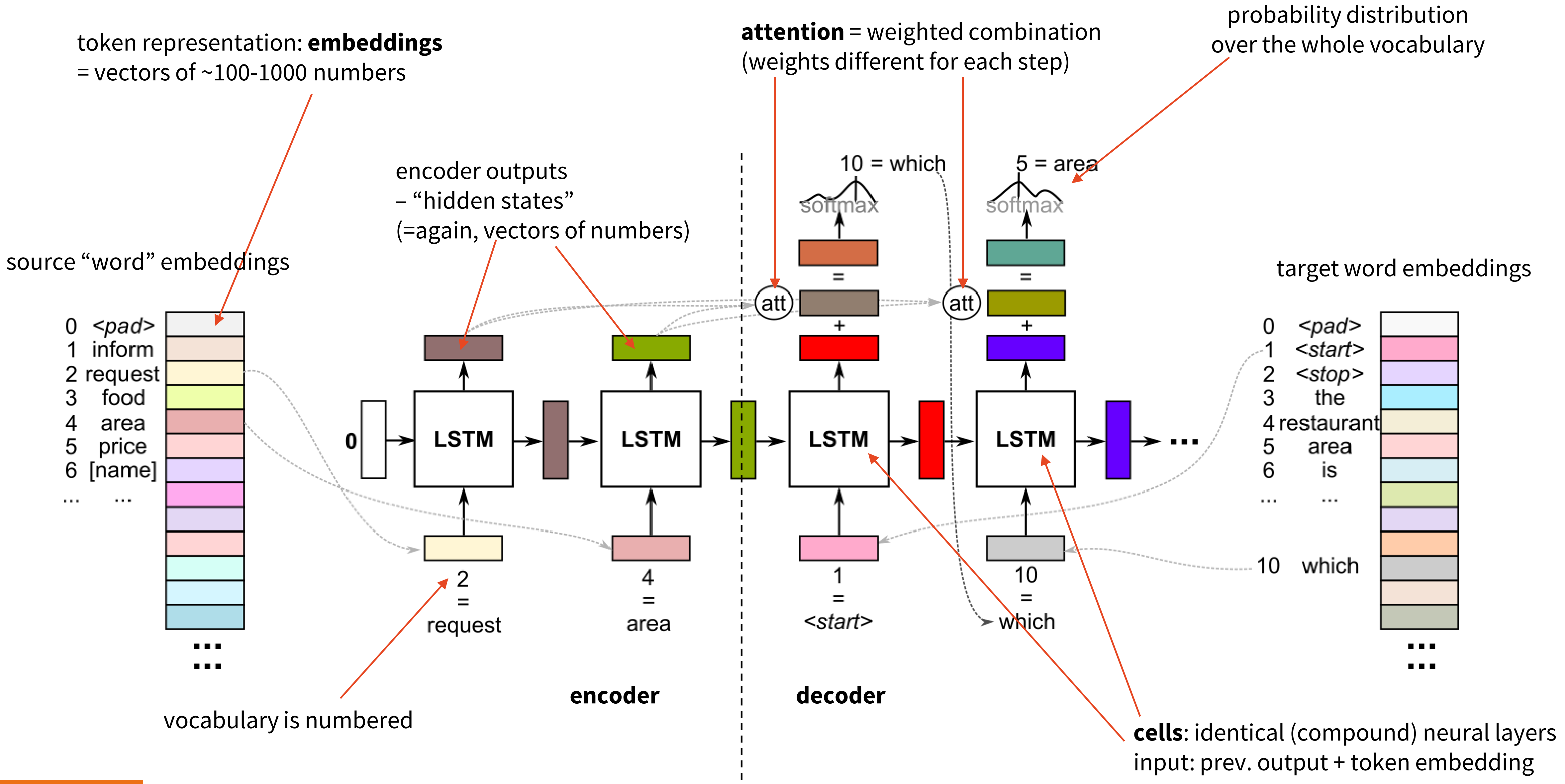




https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

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

Seq2seq RNNs with Attention



(Bahdanau et al., 2015) http://arxiv.org/abs/1409.0473

Transformer

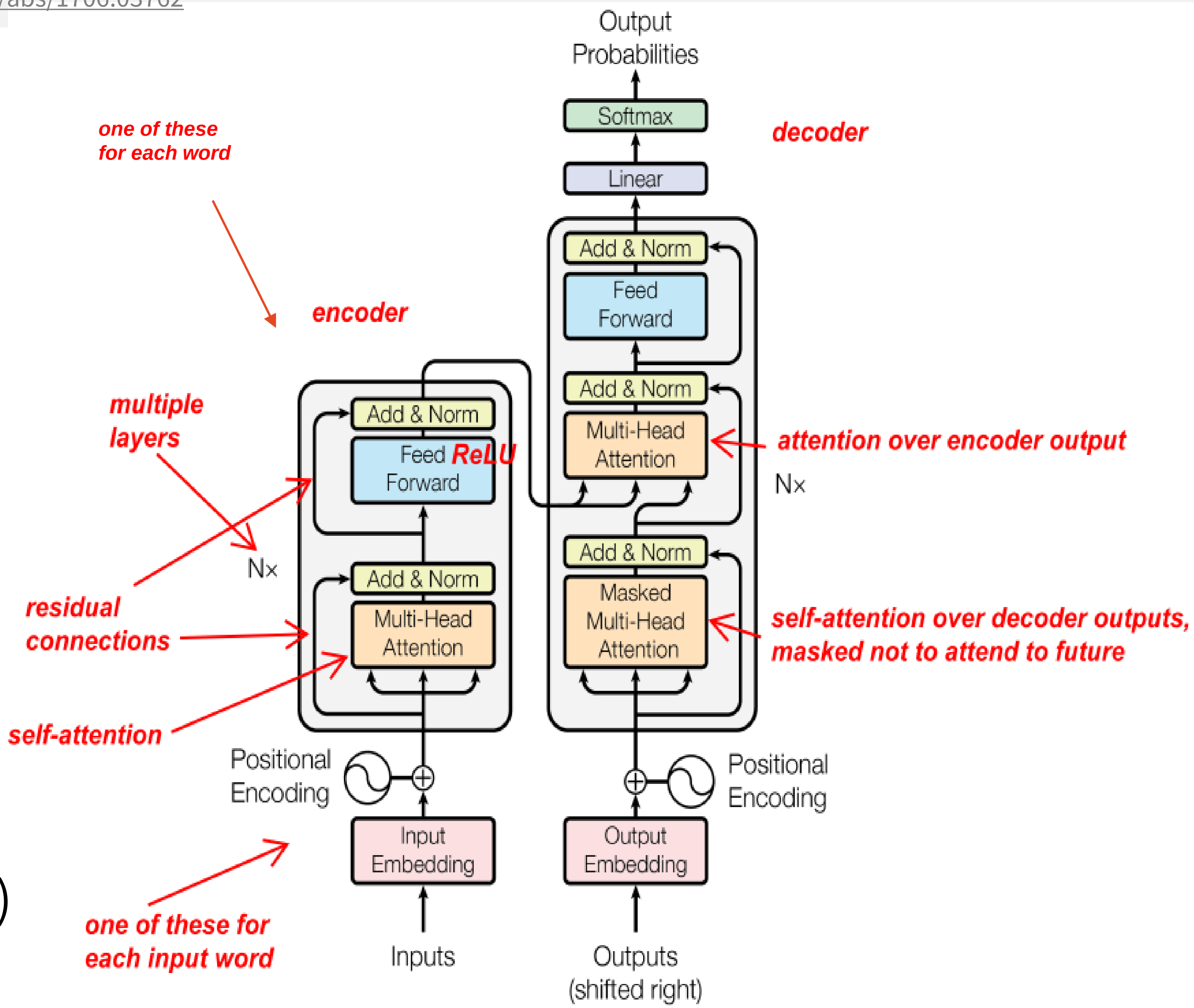
(Waswani et al., 2017)
https://arxiv.org/abs/1706.03762

- getting rid of recurrences
  - faster to train, allows bigger nets
  - replace everything with **attention + feed-forward** networks

  ⇒ needs more layers

  ⇒ uses positional encoding
- positional encoding
  - adding position-dependent patterns to the input
- attention
  - Implemented through dot product
  - **more heads** (attentions in parallel)

  – focus on multiple inputs



http://jalammar.github.io/illustrated-transformer/ https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Pretrained Language Models

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

- **Self-supervised pretraining**
  - standard supervised training, but without annotation
    - naturally occurring labels are used (text, waveform samples)
    - the task can be to fix artificially corrupted data, predict masked labels
  - used with huge amounts of data – many GBs of text (e.g. CommonCrawl)
  - models not useful for much themselves, but **can be finetuned** for the target task
    - trained further with the use of target task data

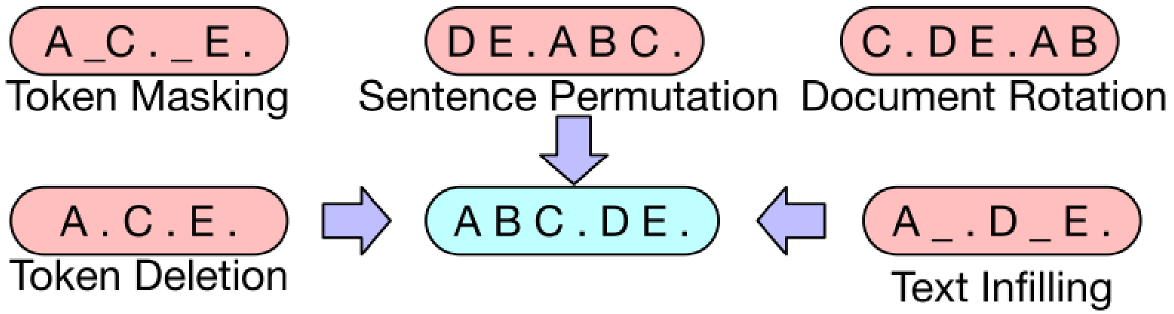
Pretrained Language Models

(Devlin et al., 2019) https://www.aclweb.org/anthology/N19-1423 https://github.com/google-research/bert

(Rogers et al., 2020) http://arxiv.org/abs/2002.12327

(Liu et al., 2019) http://arxiv.org/abs/1907.11692

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



(Lewis et al., 2020) http://arxiv.org/abs/1910.13461

(Raffel et al., 2019) http://arxiv.org/abs/1910.10683

- 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)

(Radford et al., 2019) https://openai.com/blog/better-language-models/

(Brown et al., 2020) http://arxiv.org/abs/2005.14165

(Xue et al., 2022) https://doi.org/10.1162/tacl_a_00461

https://github.com/huggingface/transformers

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

inform(food=Chinese, price=cheap)
request(address)

- 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 *uhm I'm looking for a cheap*
  - self-repairs (~6%!) *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**
  - Sentence embedding from NN-based language model + simple classifier (Logistic regression)
- **Slot value detection**
  - binary classification (“*is slot value X present?*”)
  - **slot tagging** – classify every token
    BIO/IOB scheme: slot beginning – inside slot – outside

I	*need*	*a*	*flight*	*from*	*Boston*	*to*	*New*	*York*	*tomorrow*
O	**O**	**O**	**O**	**O**	**B-dept**	**O**	**B-arr**	**I-arr**	**B-date**

- **Delexicalization**: replacing slot values by placeholders
  - named entity recognition
  - tagging, typically done by dictionaries

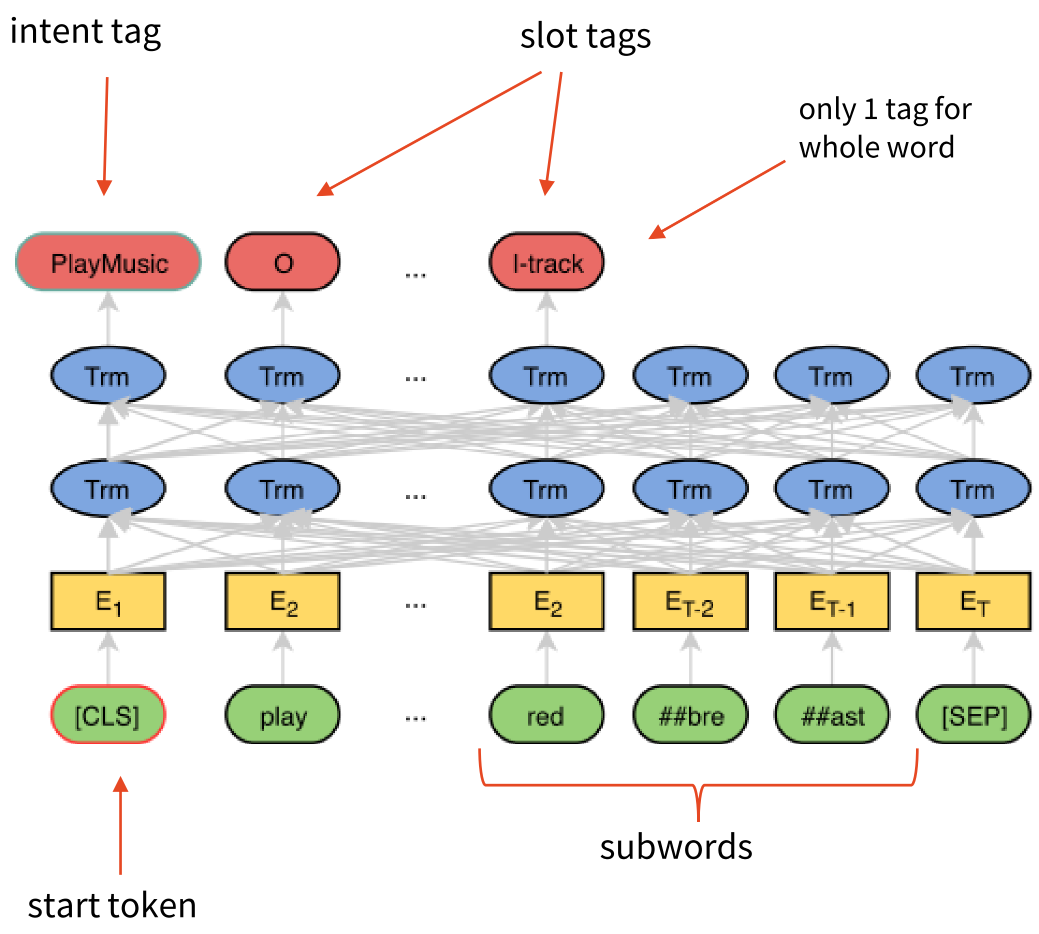
I’m looking for a Japanese restaurant in Notting Hill.
I’m looking for a <food> restaurant in <area>.

I need to leave after 12:00.
I need to leave after <time>.
(= not necessarily 1:1 with slots)

BERT-based NLU

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

- 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 ☺



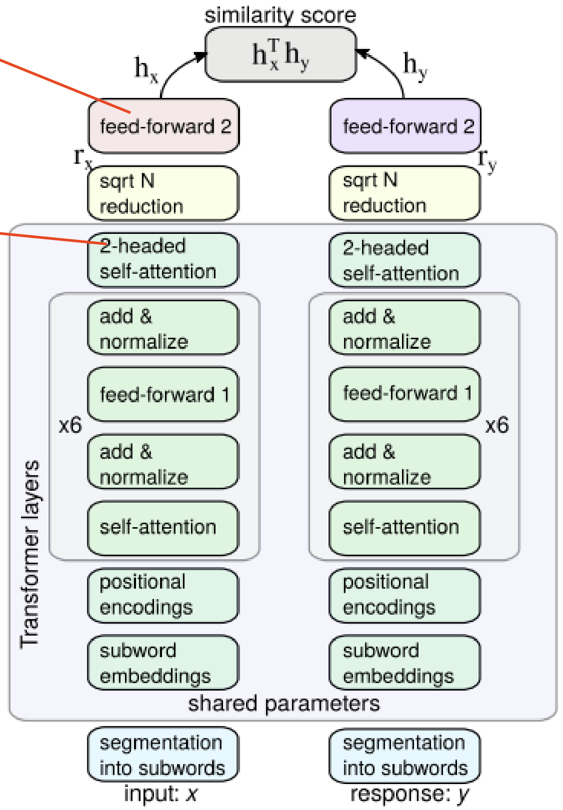
Dialogue Pretrained Models

(Henderson et al., 2020)
http://arxiv.org/abs/1911.03688

- 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**
  - **intent classification**
  - Transformer layers are fixed, not fine-tuned
  - works well for little training data (**few-shot**)

for intent classification

for slot tagging

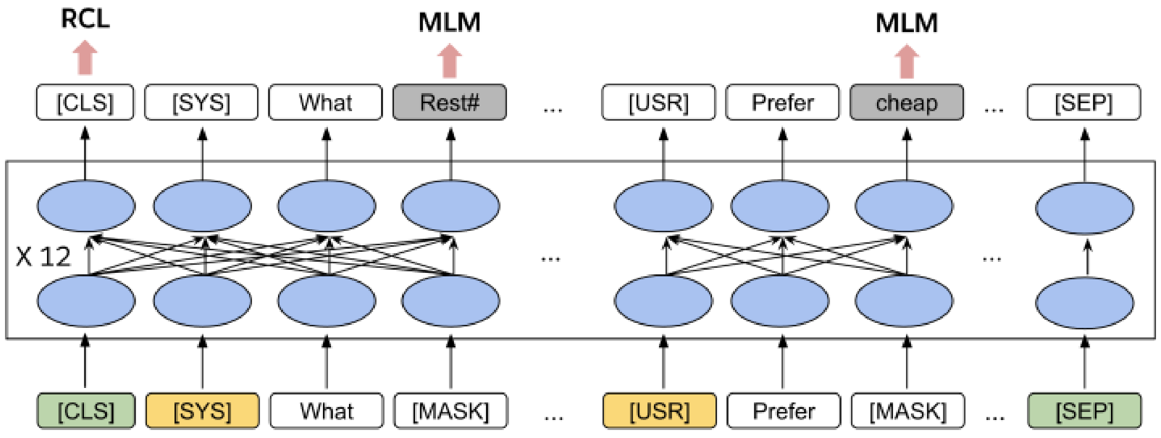


(Coope et al., 2020)
https://www.aclweb.org/anthology/2020.acl-main.11

(Casanueva et al., 2020)
https://www.aclweb.org/anthology/2020.nlp4convai-1.5

TOD-BERT

- 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

Basic State/Belief Trackers

a) Conditioned on previous state

- We always trust the NLU
- Often rule-based (but good if NLU is good)

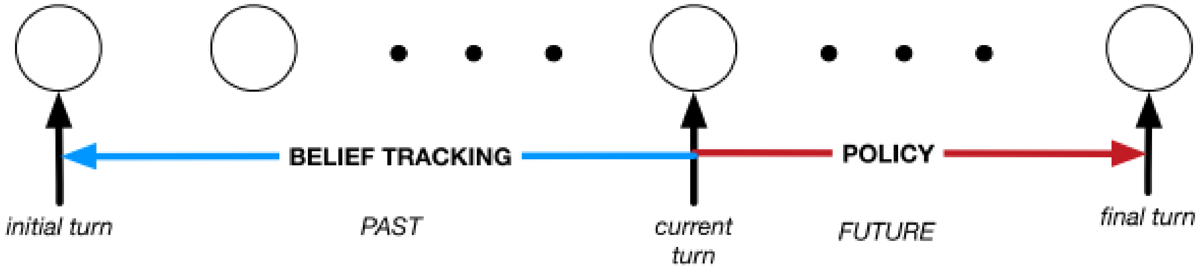
b) "NLU" over whole dialogue

- typically classification ("is slot value v present?")
  - option: limit to some candidates (from NLU/delexicalization), rank them
- may be better, but slower

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

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



(from Milica Gašić's slides)

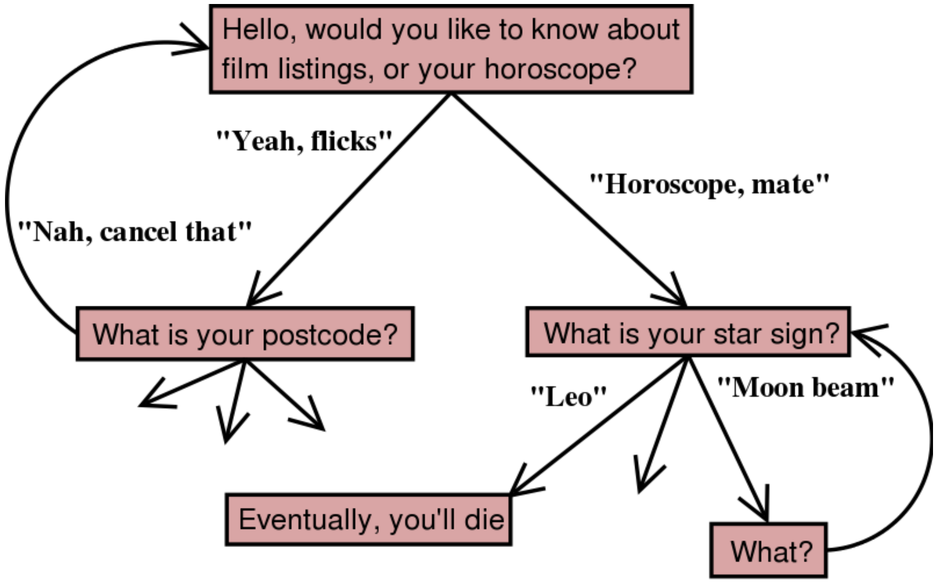
confirm(food=Chinese)

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

- Dialog manager policy should:
  - manage uncertainty from belief state ← *Did you say Indian or Italian?*
  - recognize & follow dialogue structure ← follow convention, don't be repetitive
  - plan actions ahead towards the goal ← e.g. ask for all information you require

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**



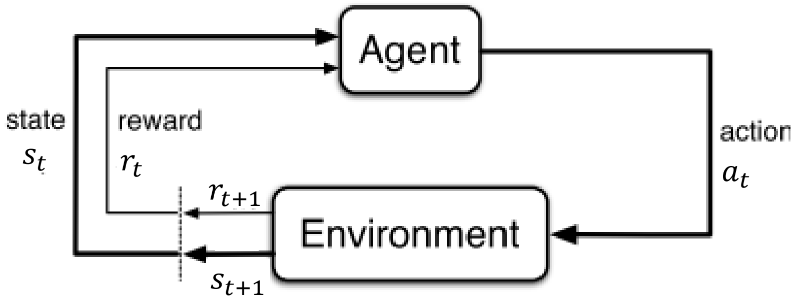

```
        elif fact['we_did_not_understand']:
            # NLG("Sorry, I did not understanc
            res_da = DialogueAct("notunderstoc
            res_da.extend(self.get_limited_cor
            dialogue_state["ludait"].reset()

        elif fact['user_wants_help']:
            # NLG("Pomoc.")
            res_da = DialogueAct("help()")
            dialogue_state["ludait"].reset()

        elif fact['user_thanked']:
            # NLG("Díky.")
            res_da = DialogueAct('inform(cordi
            dialogue_state["ludait"].reset()

        elif fact['user_wants_restart']:
```

Why Reinforcement Learning

- **Action selection ~ classification** → use supervised learning?
  - set of possible actions is known
  - belief state should provide all necessary features
- Yes, but…
  - Supervised ((sequence) classification) training is efficient with multiple good responses - RL can be
  - Supervised training cannot train with negative feedback - Noise contrastive estimation is not good enough.
  - RL is able to handle delayed feedback
  - supervised classification **doesn't plan ahead** - RL optimizes for the whole dialogue, not just the immediate action
- Interesting topic in general is how to work with API/DB calls (we can not take the derivative)
- The API calls could be used for dialogue state annotation

Reinforcement learning: Definition

- Agent in an environment, **state-action-reward**



- 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

return = accumulated long-term reward

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

$\gamma \in [0,1]$ = **discount factor** (immediate vs. future reward trade-off)

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

$\mathbb{E}[R_t | \pi, s_0]$ ← expected R_t if we start from state s_0 and follow policy π

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
    - **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

Natural Language Generation (NLG) / Response Generation

- 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

=

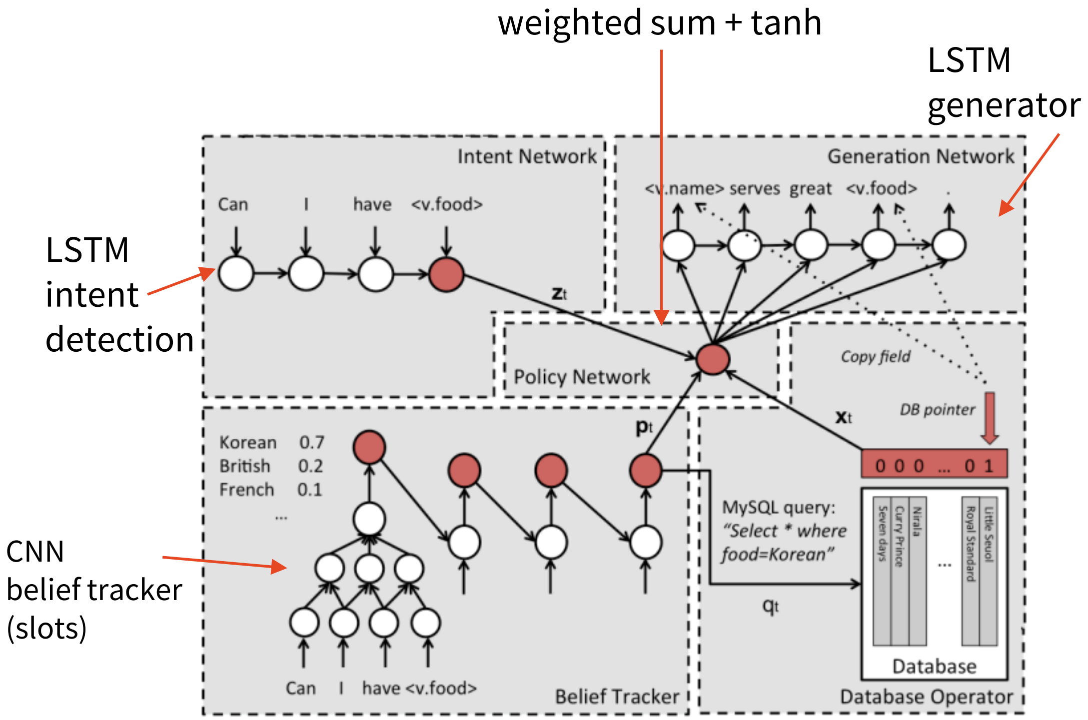
Golden Dragon is a cheaply priced restaurant serving Chinese food.

- Statistical approach: **seq2seq**/pretrained language models
  - input: system dialogue act, output: sentence (operation similar to →)

4. End-to-end models

End-to-End Systems

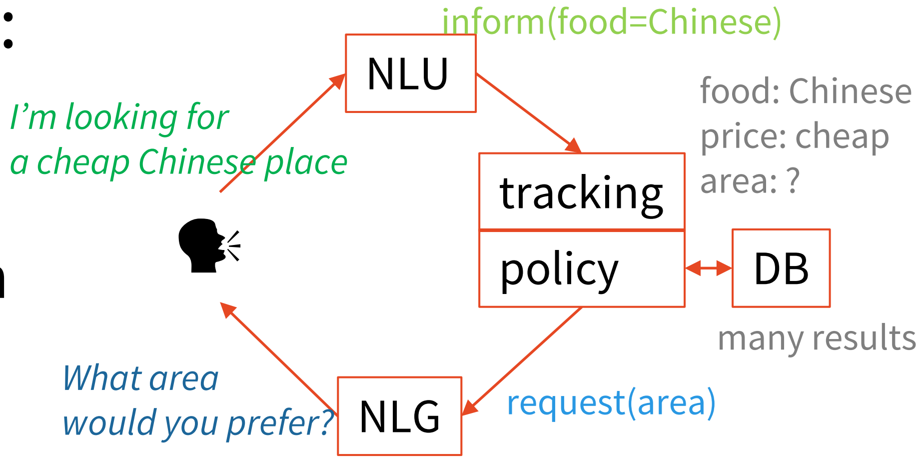
- experimental, research state-of-the-art
- the whole system (NLU/DM/NLG) is a single neural network
  - joint training ("end-to-end")
  - more elegant
  - potentially easily retrainable
- typically still needs annotation
  - same as individual modules
  - can be less predictable
- connecting the database is a problem
  - this step is done separately



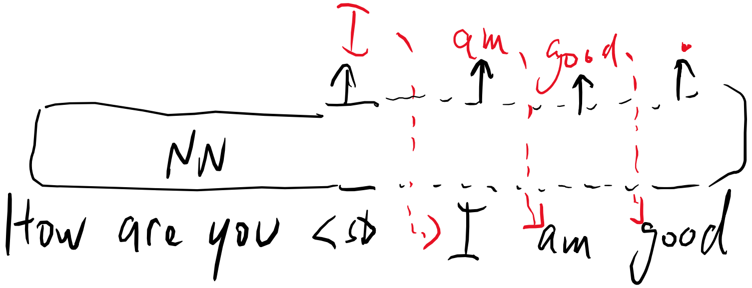
(Wen et al., 2017)
https://www.aclweb.org/anthology/E17-1042/

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



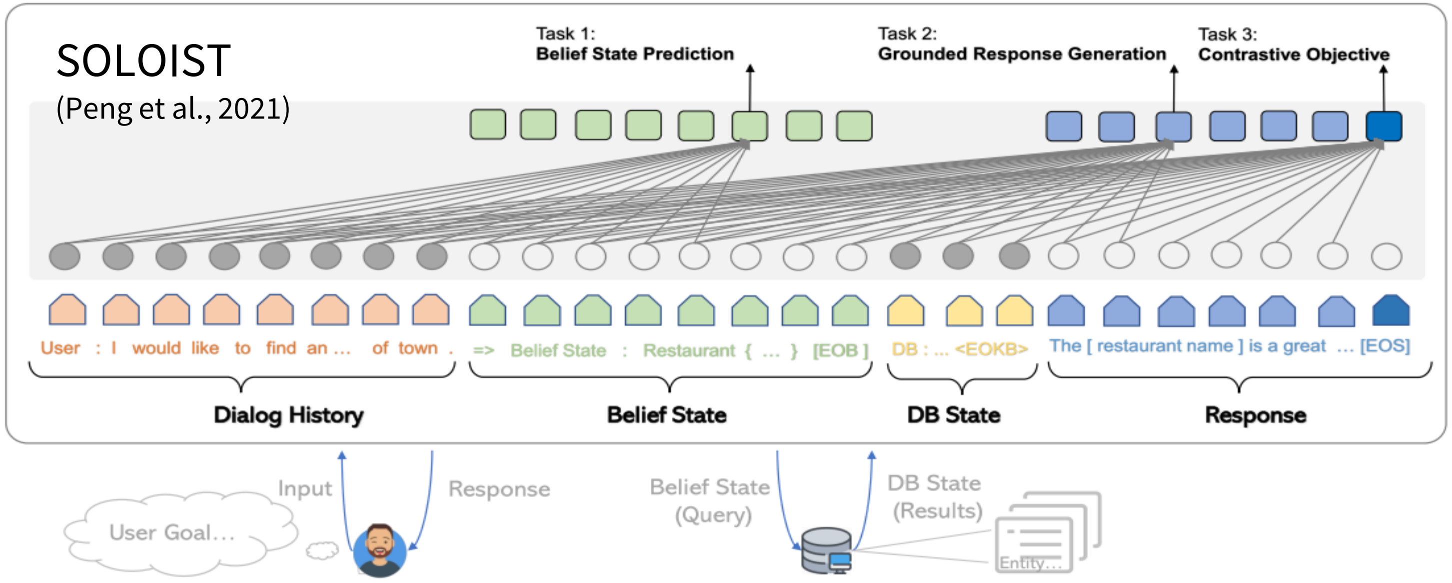
- End-to-end:
  - joint supervised optimization, RL still works
  - still needs dialog action level annotation
  - typically needs a lot of data
  - less control of outputs: hallucination, dull/repetitive



End-to-end Dialogue with GPT-2

(Peng et al., 2021) http://arxiv.org/abs/2005.05298
(Hosseini-Asl et al., 2020) http://arxiv.org/abs/2005.00796
(Ham et al., 2020) https://www.aclweb.org/anthology/2020.acl-main.54
(Yang et al., 2021) http://arxiv.org/abs/2012.03539

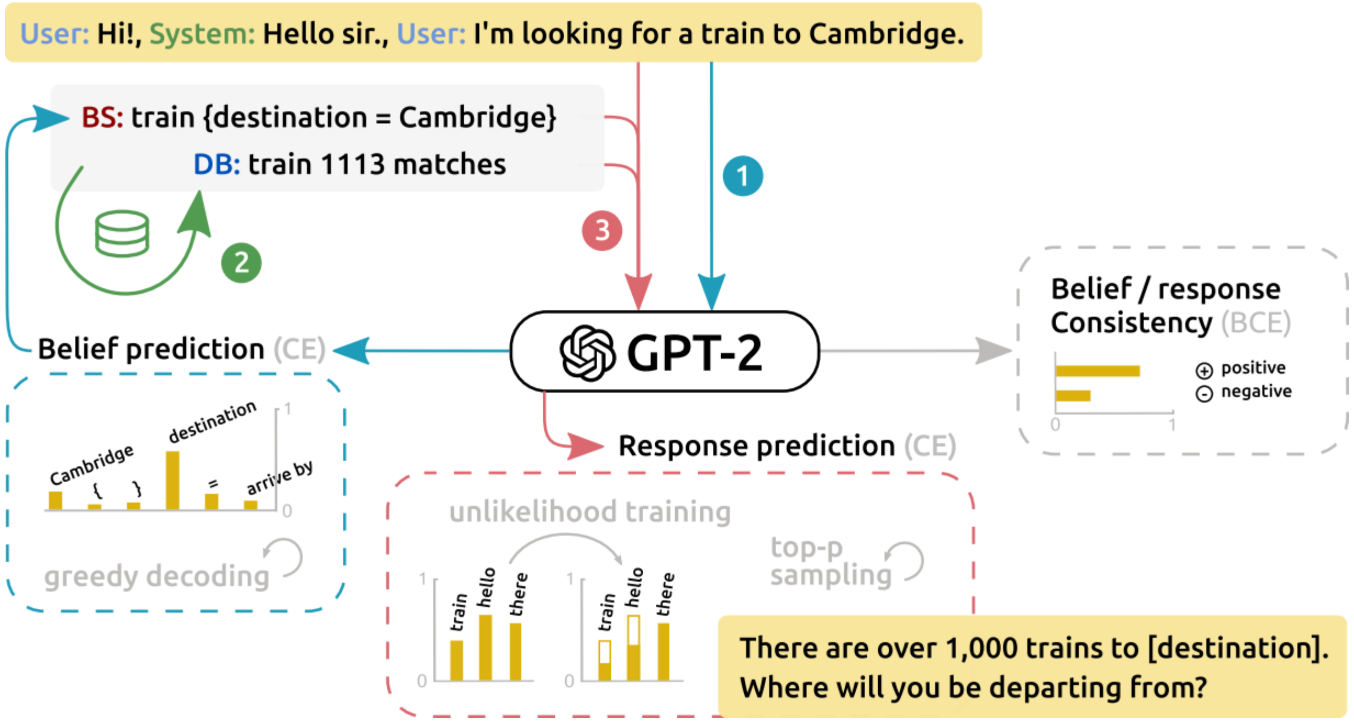
- Multiple recent dialog systems are based on GPT-2 (SOLOIST, UBAR, SimpleTOD, NeuralPipeline)
  - decoder-only pretrained language model from OpenAI
- Similar to Sequicity, everything recast as sequence generation
  - dialogue context, belief state, database outputs represented as sequences
  - GPT-2 **prompting**: force-decode some input (ignore softmaxes, feed your tokens)
- Multi-step operation:
  1) prompt with context & decode belief state
  2) query DB (external)
  3) prompt with DB output & decode response



AuGPT: Charles University approach

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

- Same idea as before, 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

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

- **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!**

context	state	**response**	**consistent?**
i want a cheap italian restaurant	*{ price range = cheap , food = Italian }*	ok which area ?	✓
i want a cheap Italian restaurant	*{ price range = cheap , food = Italian }*	thanks, goodbye !	✗ bad response
i want a cheap italian restaurant	*{ destination = Cambridge , leave at = 19:00 }*	ok which area ?	✗ bad state
i want a cheap italian restaurant	*{ area = north , food = Chinese }*	ok which area ?	✗ bad state (same domain)

new in AuGPT

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/)
  - 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

Questions?

If you have any question, come or reach us on Discord