

Speech processing, dialogue management, system design & evaluation

Pierre Lison

IN4080: Natural Language
Processing (Fall 2023)

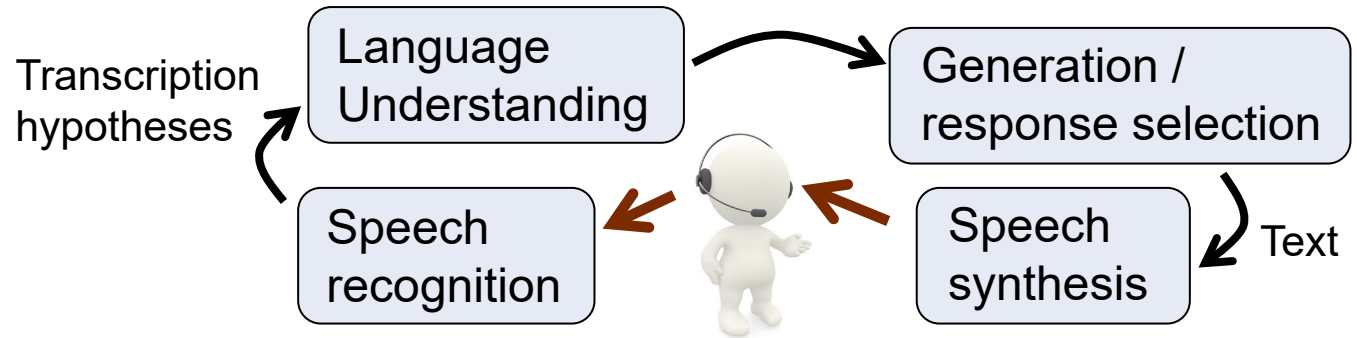
31.10.2023



Plan for today

- ▶ Speech processing
- ▶ Dialogue management
 - Handcrafted approaches
 - Data-driven approaches
- ▶ Design of dialogue systems
 - Architectures
 - Evaluation

Spoken dialogue systems



Spoken interfaces add a layer of complexity

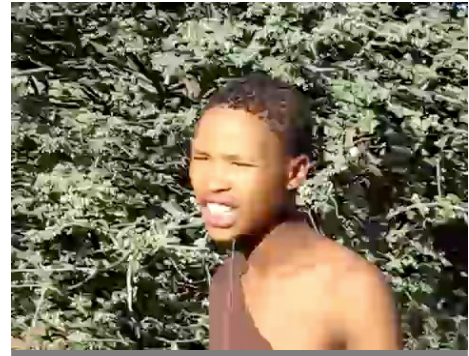
- ▶ Need to handle uncertainties, ASR errors etc.
- ▶ Speech communicates more than just words (intonation, emotions in voice, etc.)
- ▶ Need to handle turn-taking

Speech production

- ▶ Sounds are *variations in air pressure*
- ▶ How are they produced?
 - An **air supply**: the *lungs* (we usually speak by breathing out)
 - A **sound source** setting the air in motion (e.g. vibrating) in ways relevant to speech production: the *larynx*, in which the *vocal folds* are located
 - A set of 3 **filters** modulating the sound: the *pharynx*, the *oral tract* (teeth, tongue, palate, lips, etc.) & the *nasal tract*

Speech production

Visualisation of the vocal tract via *magnetic resonance imaging* [MRI]:

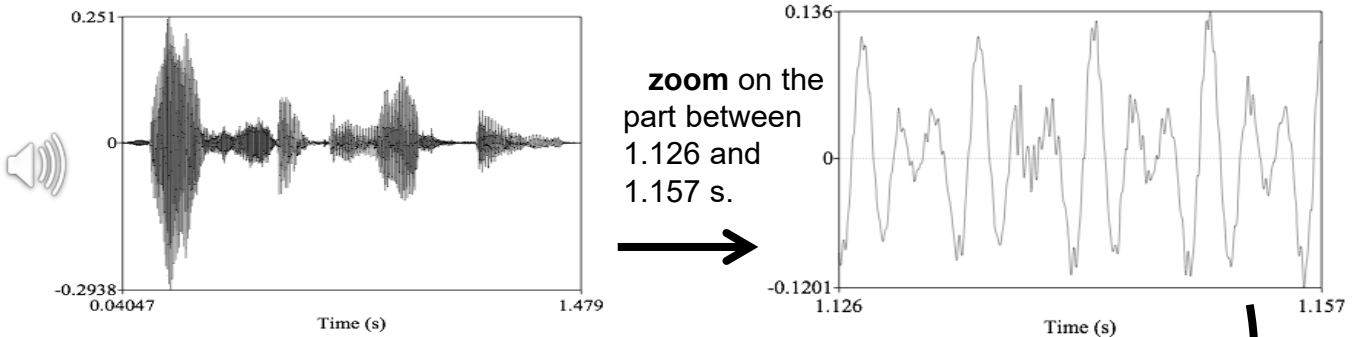


NB: A few languages also rely on sounds not produced by vibration of vocal folds, such as *click languages* (e.g. Khoisan family in south-east Africa):

Speech perception

A (speech) sound is *a variation of air pressure*

- This variation originates from the speaker's speech organs
- We can plot a *wave* showing the changes in air pressure over time (zero value being the normal air pressure)



About 4 cycles in the waveform, which means a frequency of about $4/0.03 \approx 129$ Hz

Important measures

1. The **fundamental frequency F_0** : lowest frequency of the sound wave, corresponding to the speed of vibration of the vocal folds (between 85-180 Hz for male voices and 165-255 Hz for female voices)
2. The **intensity**: the signal power normalised to the human auditory threshold, measured in **dB** (decibels):

$$\text{Intensity} = 10 \log_{10} \frac{\text{Power}}{P_0} = 10 \log_{10} \frac{1}{NP_0} \sum_{i=1}^N y(t_i)^2$$

Total energy of signal

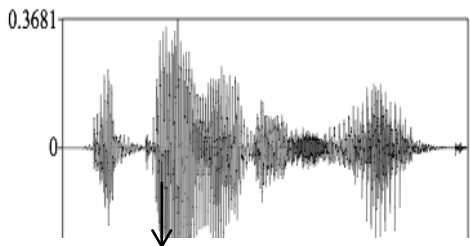
for a sample of N time points t_1, \dots, t_N

P_0 is the human auditory threshold, = 2×10^{-5} Pa

Note: dB scale is logarithmic, not linear!

The speech recognition task

Input: Audio data



Sequence **O** of acoustic observations (i.e. every 20 ms)



Output: Transcription

"The ball is red"



Goal: Map speech signal **O** into sequence of linguistic symbols \widehat{W} (words or characters):

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|O)$$

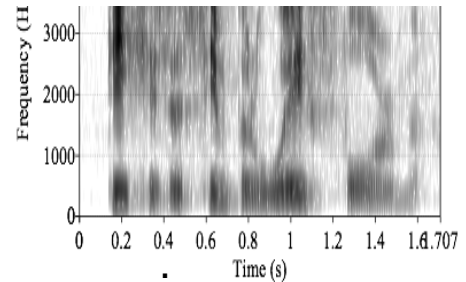
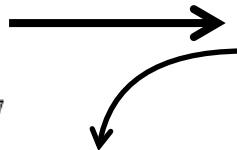
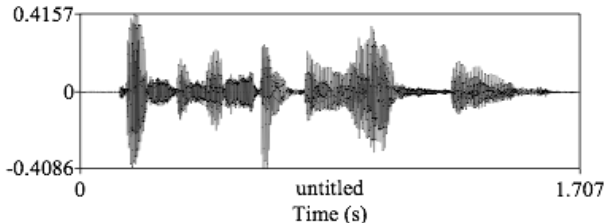
Why is ASR difficult?

- ▶ *Many sources of variation:* speaker voice (and style), accents, ambient noise, etc.



Preprocessing

- ▶ Most speech sounds cannot be distinguished from the raw waveform
- ▶ Better: convert the signal to a representation of the signal's *component frequencies*
 - Based on Fourier's transform



spectrogram showing which frequencies are most active at a given time

Neural ASR

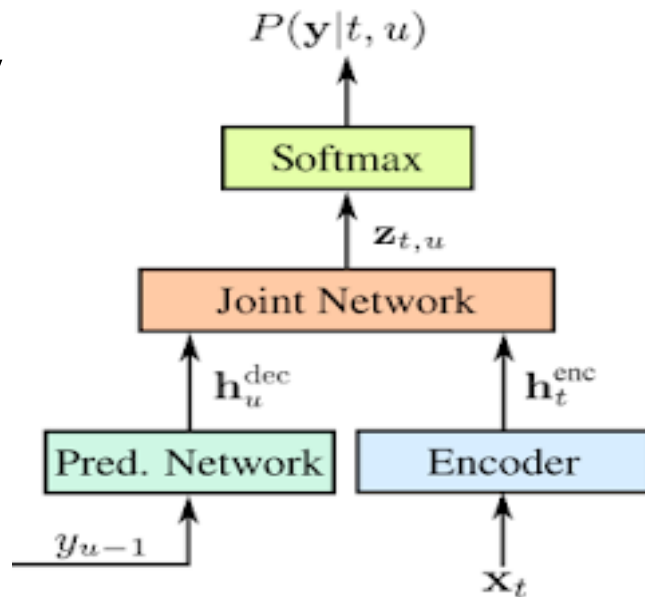
- ▶ The best performing ASR are *deep, end-to-end neural architectures*
 - Less dependent on external resources (such as pronunciation dictionaries)
 - Move from carefully handcrafted acoustic features to *learned* representations
- ▶ Too time demanding to review here
 - But they rely on the same building blocks as other NNs: convolutions, recurrence, (self-)attention, etc.

Neural ASR

<https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html>

An example of a relatively simple neural model:
Google's on-device ASR

- ▶ *Encoder* maps audio signal \mathbf{x}_t to hidden representations (with stacked LSTMs)
- ▶ *Prediction Network* is a language model
- ▶ Model then merges the two hidden representations and predicts outputs character-by-character



ASR evaluation

- ▶ Standard metric: **Word Error Rate**
 - Measures how much the utterance hypothesis h differs from the «gold standard» transcription t^*
- ▶ = Minimum edit distance between h and t^* , counting the number of word substitutions, insertions and deletions:

$$\text{Word Error Rate} = 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Number of words in transcription}}$$



ASR evaluation

Gold standard Transcription	yes can you now rotate this triangle
ASR hypothesis	yes can you not rotate this triangle there

$$\text{WER} = 100 \times \frac{1 \text{ Sub} + 1 \text{ Ins}}{7}$$
$$= 28.6\%$$

Gold standard Transcription	there is five and
ASR hypothesis	the size and

$$\text{WER} = 100 \times \frac{2 \text{ Sub} + 1 \text{ Del}}{4}$$
$$= 75\%$$



Disfluencies

- ▶ Speakers construct their utterances «as they go», incrementally
 - Production leaves a *trace* in the speech stream
- ▶ Presence of multiple disfluencies
 - Pauses, fillers («øh», «um», «liksom»)
 - Repetitions («the the ball»)
 - Corrections («the ball err mug»)
 - Repairs («the bu/ ball»)

Some disfluencies



så gikk jeg e flytta vi til Nesøya da begynte jeg på barneskolen der

og så har jeg gått på Landøya ungdomsskole # som ligger ## rett over broa nesten # rett med Holmen



jeg gikk på Bryn e skole som lå rett ved der vi bodde den gangen e barneskole

videre på Hauger ungdomsskole



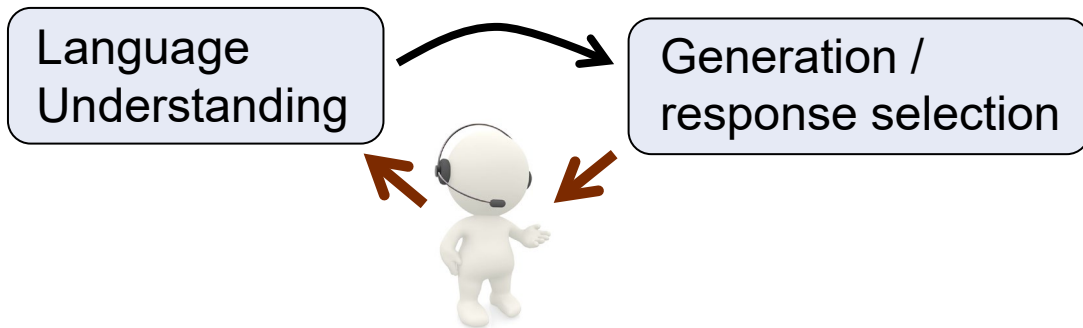
da hadde alle hele på skolen skulle liksom # spise julegrøt og det va- det var bare en mandel

og da var jeg som fikk den da ble skikkelig sånn " wow # jeg har fått den " ble så glad

Plan for today

- ▶ **Dialogue management**
 - Handcrafted approaches
 - Data-driven approaches
- ▶ Design of dialogue systems
 - Architectures
 - Evaluation

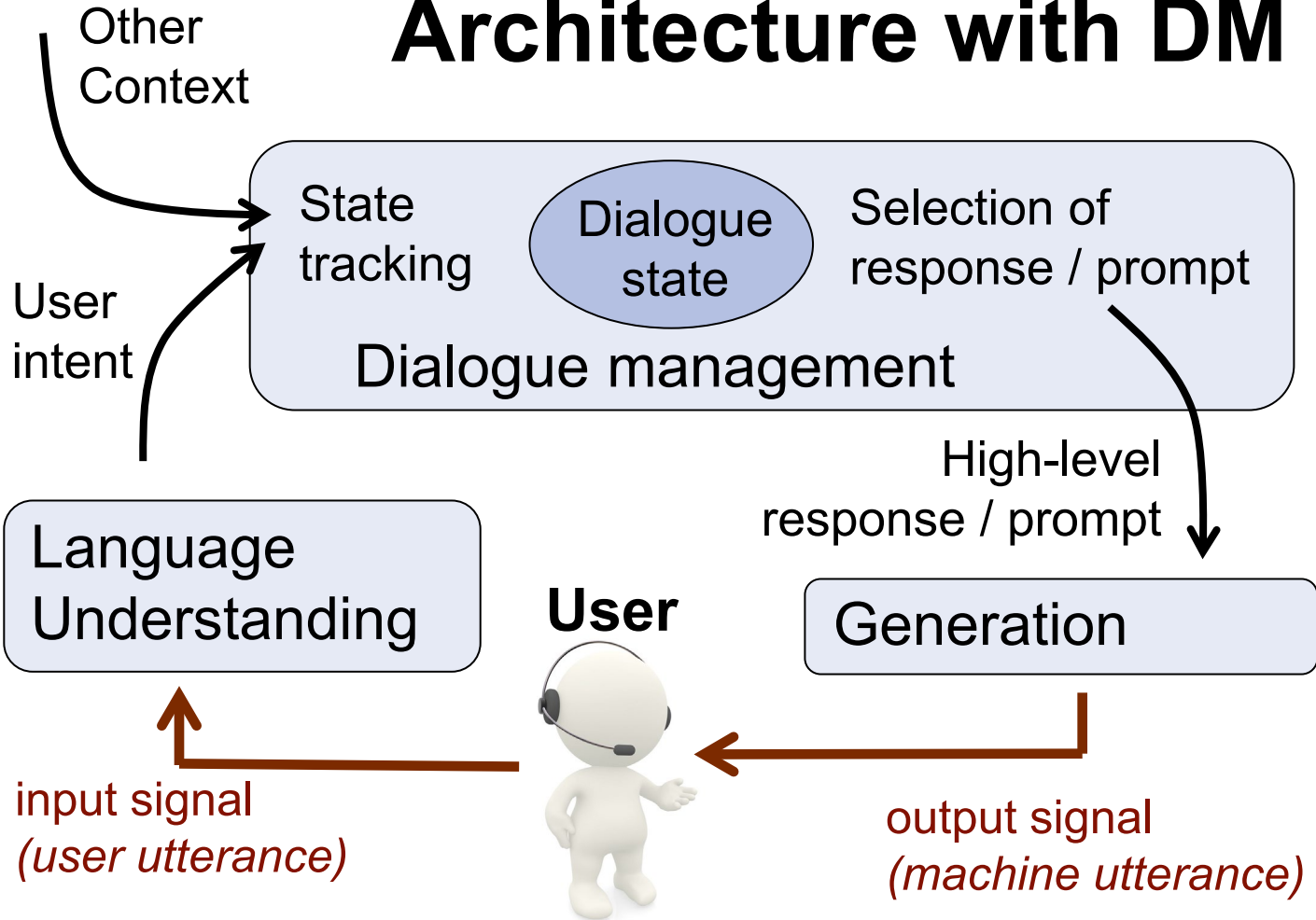
Basic architecture



This pipeline is often used for chatbots

- **Main limitation:** no management of the dialogue itself (beyond current utterance)
- Most appropriate for short interactions

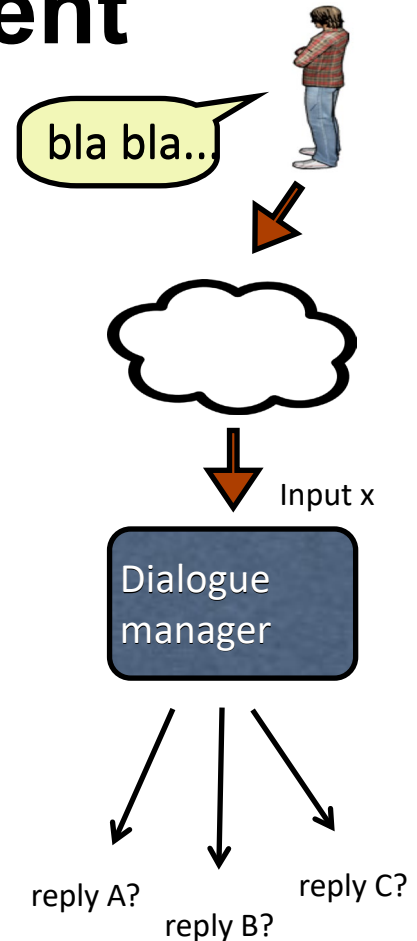
Architecture with DM



Dialogue management

... is about **decision-making**:

- **What** and **when** should the system decide to say or do something
- decision-making under uncertainty, since the communication channel is “noisy” (errors, ambiguities, etc.)
- Actions can be both linguistic and non-linguistic (booking a flight ticket, picking up an object, etc.)
- The same holds for observations (visual input, external events, etc.)

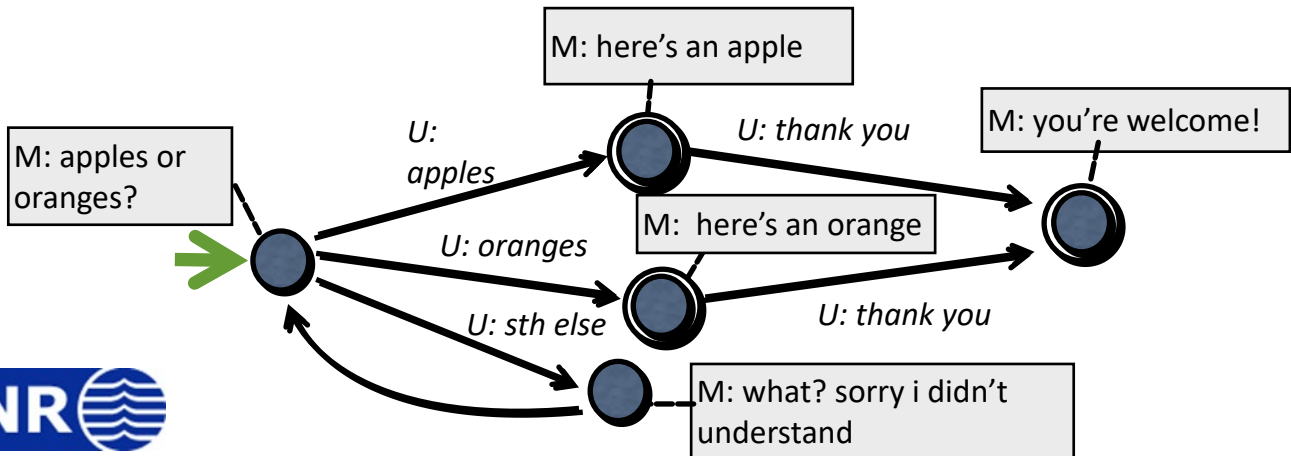


Finite-state automata

= encode dialogue strategies as **finite-state automata**

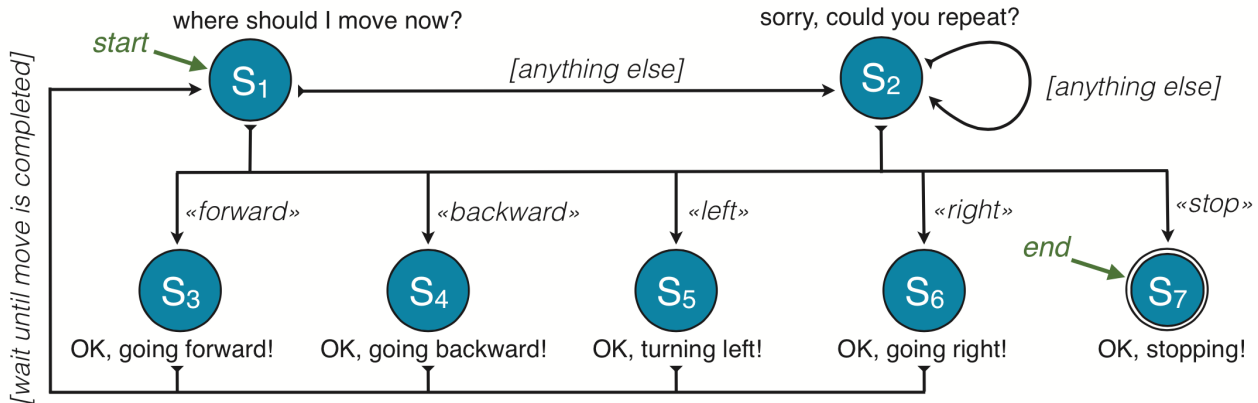
Also called **flowcharts**
(somewhat more loosely)

- the nodes represent *machine actions*
- and the edges possible (mutually exclusive) *user responses*



Finite-state automata

- ▶ Transitions can relate to other signals than user inputs (for instance, external events)
- ▶ And can also express complex conditions (pattern matching on the user input, confidence thresholds, etc.)



Finite-state automata

Advantages	Limitations
<ul style="list-style-type: none">• Easy to design• Fast, efficient• Does not require dialogue data• <i>Predictable</i> system behaviour (both for the user and for the system designer)	<ul style="list-style-type: none">• Only allows for <i>scripted</i> interactions - not "true" conversation• No principled account of uncertainties• Difficult to scale to complex domains with many variables and alternative inputs

Frame-based managers

- ▶ The interaction flow can be made slightly more flexible in *frame-based systems*
- ▶ The state is represented as a **frame** with **slots** to be filled by the user's answers

Slot	Question
ORIGIN CITY	«From what city are you leaving?»
DESTINATION CITY	«Where are you going?»
DEPARTURE TIME	«When would you like to leave?»
ARRIVAL TIME	«When do you want to arrive?»

Frame-based managers

- ▶ The user will sometimes provide additional information to the system's questions

System: What is your departure?

User: I want to leave from Oslo before 9:00 AM»

- ▶ The system should fill the appropriate slots with all available information
- ▶ Given the current state, the dialogue manager selects an unfilled slot and ask the user for its value ... *and we repeat until all slots are filled*

Interaction style

- ▶ Rigid, repetitive structure of the interaction
- ▶ Irritating confirmations & acknowledgements
- ▶ No user or context adaptivity



“Saturday night live” sketch comedy, 2005

Plan for today

- ▶ Dialogue management
 - Handcrafted approaches
 - **Data-driven approaches**
- ▶ Design of dialogue systems
 - Architectures
 - Evaluation

Data-driven techniques

The approaches presented so far suffer from several limitations:

- Difficult to predict the user behaviour in advance
- They ignore all the *uncertainties* appearing through the dialogue (ASR errors, ambiguities, etc.)
- Unable to *learn* or adapt to the users or the environment (leading to rigid/repetitive behaviour)
- Limited to one goal... but real interactions are trade-offs between *various competing objectives*

Data-driven techniques

- ▶ *Solution*: perform **automatic optimisation** of the «dialogue policies» from experience:
 - Often based on *reinforcement learning* techniques
 - "Experience": interactions with real or simulated users
- ▶ **General procedure**:
 - Dialogue manager starts with «dumb» dialogue policy
 - It interacts with users and receives a **feedback**
 - It can then correct his policy based on this feedback
 - Repeat process until policy is fully optimised

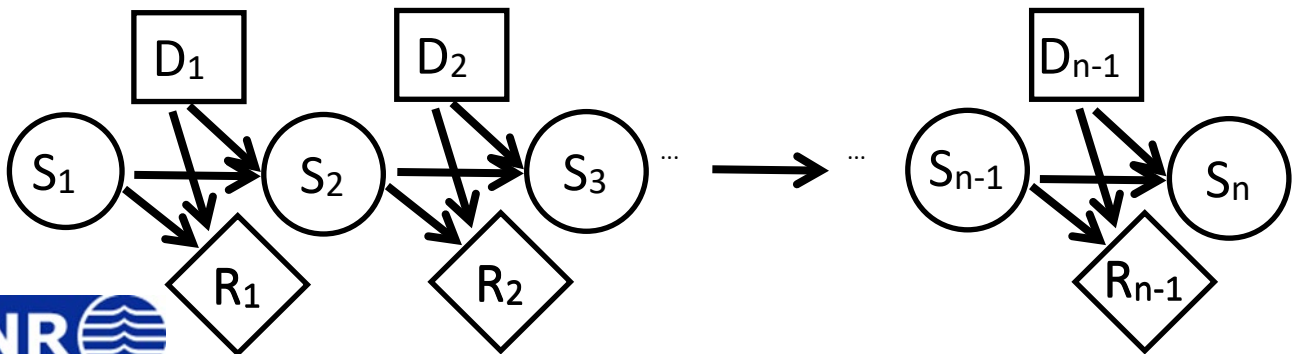
Data-driven techniques

- ▶ Most tasks must encode trade-offs between various, competing objectives
 - A flight booking system must book the right ticket
 - But it must do so with the fewest number of requests
- ▶ Typically encoded via **rewards** (utilities) associated to particular state/action pairs

State	Action	Reward
User wants to book ticket x	Booking x	+10
User wants to book ticket x	Booking $y \neq x$	-30
User wants to book ticket x	Clarification request	-1

Markov Decision Processes

- ▶ We can define these ideas more precisely using a formalism called **Markov Decision Processes** (MDPs)
- ▶ Markov Decision Processes are an extension of Markov Chains where the agent *selects an action* at each state
 - This action will then modify the state space
 - And will yield a particular reward for the agent

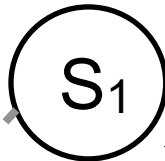


Graphical notation

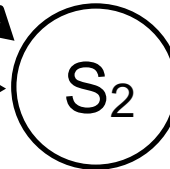
(decision
variable)



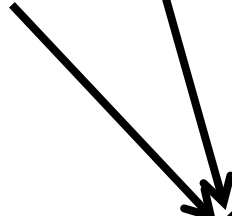
$P(S_2|S_1,d)$ determines the *probability* of reaching S_2 when executing action D in state S_1



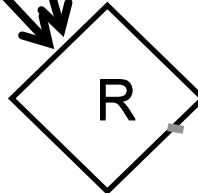
(random
variable)



(random
variable)



(utility
variable)



$R(S_1,D)$ determines the *utility* of executing action D while in state S_1

$P(S_1)$ determines the probability of being in state S_1

Markov Decision Processes

A MDP is as a tuple $\langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{R} \rangle$, where:

- ▶ \mathbf{S} is the *state space* (possible states in the domain)
- ▶ \mathbf{A} is the *action space* (possible actions for the agent)
- ▶ \mathbf{T} is the *transition function*, defined as $T(s, a, s') = P(s'|s, a)$. It is the probability of arriving to state \mathbf{s}' after executing action \mathbf{a} in state \mathbf{s} .
- ▶ \mathbf{R} is the *reward function*, defined as $R : S \times A \rightarrow R$. It is a real number encoding the utility for the agent to perform action \mathbf{a} while in state \mathbf{s} .

Expected cumulative reward

- ▶ In an MDP, the agent seeks to maximise its *expected cumulative reward* $Q(s,a)$



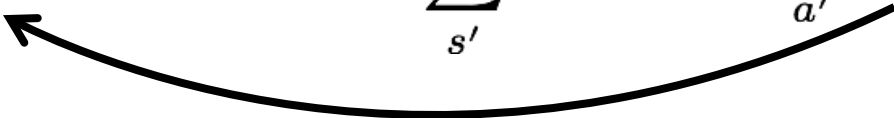
The agent must try to predict future inputs/rewards

The rewards accumulate over time

- ▶ How much worth is a reward expected at time $(t+i)$ compared to one received right now?
 - We use a *discount factor* γ to capture this balance
 - Related to *delayed gratification* in psychology

Bellman equation

The *Bellman equation* tells us that we can write the expected cumulative reward Q in a recursive fashion:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$


Notice that we are estimating the Q-values based on... our estimation of the Q-values (can be used to iteratively refine these estimates until convergence)

MDP policy

- ▶ Given an MDP, a (dialogue) policy tells us which action to execute in each state
- ▶ A dialogue policy is a *mapping* $\pi: S \rightarrow A$ from states to actions
- ▶ An *optimal* dialogue policy π^* is a policy that always outputs the action yielding the maximum expected cumulative reward:

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Reinforcement learning

- ▶ **Reinforcement learning** can help us learn these Q values through interaction
- ▶ They work by iteratively refining their estimate of the Q values
 - The agent acts in the environment and observes both states and rewards
 - This operation is repeated until convergence
- ▶ In dialogue systems: policy learning can be done either in simulation or with real users



[R. Sutton & A. Barto (2018): «*Reinforcement Learning: An Introduction*»]
([complete book available online!](#))

Plan for today

- ▶ Dialogue management
 - Handcrafted approaches
 - Data-driven approaches
- ▶ **Design of dialogue systems**
 - Architectures
 - Evaluation

Pipeline architectures

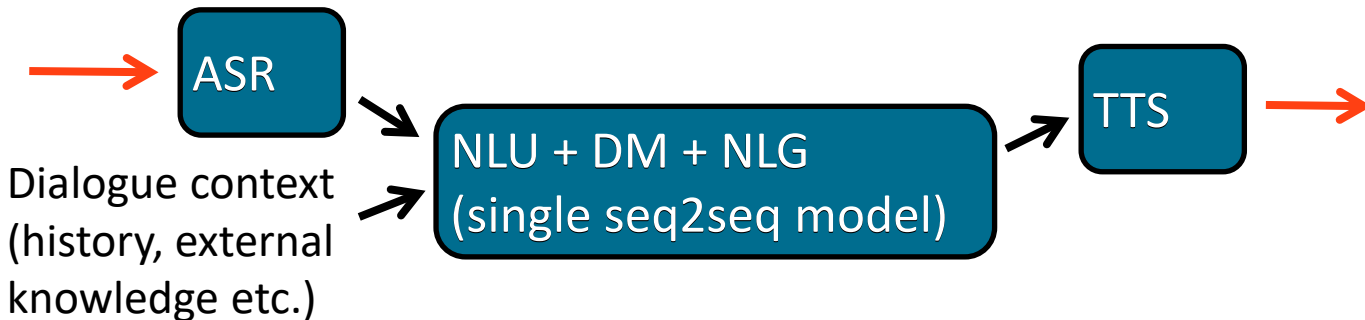
- ▶ Components connected in processing chain
- ▶ Each component is a black box getting inputs from its predecessor and generating an output



Limitations:

- No feedback between components
- Rigid information flow
- Poor turn-taking behaviour (system does not react until the full pipeline has been traversed)

End-to-end architectures



Single neural model mapping user inputs to responses

- **Pro**: no need for separate modules or annotated data
- **Con**: less modular setup, need dialogue corpus

Dialogue management often done implicitly or by modifying the prompt

Incrementality

Humans process and produce language **incrementally**:

- ▶ When listening, we don't wait for an utterance to be fully pronounced to process it!
- ▶ We gradually refine our understanding as we go, phoneme by phoneme
- ▶ We also continuously provide feedback signals



www.savagechickens.com

Human-human dialogues are full of *interruptions*, *speech overlaps*, *backchannels*, and *co-completion* of utterances

Incrementality

- ▶ But most dialogue systems operate in «batch mode»
 - NLU expects full utterance as input
 - TTS waits for complete system response to start synthesis
- ▶ Leads to «ping-pong» turn-taking behaviour:
 - Alternating turns between user & system, one speaker at a time



Can dialogue systems be made to work *incrementally*, on partial units of content?

How to collect data?

- ▶ "Chicken-and-egg" problem:
 - Need data to train data-driven models
 - But to collect data, we need a system that can interact with users

- ▶ One solution is to use *Wizard-of-Oz* studies:
 - Replace the system with a human operator (without the users being aware of it)



Evaluation



- ▶ Some dialogue processing tasks have standard evaluation metrics:
 - ASR: *Word Error Rate*
 - NLU: [*precision, recall, F-score*] for intent recognition and slot-filling
 - TTS: evaluation by human listeners on sound intelligibility and quality
- ▶ But how do we evaluate the end-to-end the conversational behaviour of the system?

Evaluation

One way to evaluate is via **user satisfaction ratings**

The ratings can be obtained from surveys that users are asked to fill after interacting with the system:

<i>TTS Performance</i>	Was the system easy to understand ?
<i>ASR Performance</i>	Did the system understand what you said?
<i>Task Ease</i>	Was it easy to find the message/flight/train you wanted?
<i>Interaction Pace</i>	Was the pace of interaction with the system appropriate?
<i>User Expertise</i>	Did you know what you could say at each point?
<i>System Response</i>	How often was the system sluggish and slow to reply to you?
<i>Expected Behavior</i>	Did the system work the way you expected it to?
<i>Future Use</i>	Do you think you'd use the system in the future?



[M. Walker et al. (2001), «Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems», *Proceedings of ACL*]

Evaluation

- ▶ However, user evaluation surveys are expensive and time-consuming
 - Not feasible to conduct after each system change!
 - Can we automate the evaluation process?
- ▶ Solution: rely on metrics that can be extracted from interaction logs, and are known to *correlate with user satisfaction*
 - Improving these observable metrics should therefore increase user satisfaction



[M. Walker et al. (1997), "PARADISE: A general framework for evaluating spoken dialogue agents", Proceedings of ACL]

Evaluation

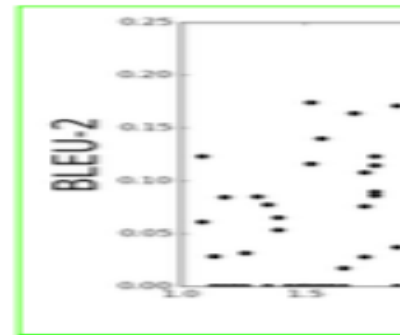
Criteria	Description	Possible metrics
<i>Task completion success</i>	How often did the system complete its task successfully?	- κ agreement on slots - completion ratio
<i>Efficiency costs</i>	How efficient was the system in executing its task?	- nb of turns (from user, system, or both) - total elapsed time
<i>Quality costs</i>	How good was the system interaction?	- nb of ASR rejection prompts - nb of user barge-ins - nb of error messages



NB: this list of metrics is of course not exhaustive!

Evaluation

- ▶ Can't we use metrics like BLEU to compare system outputs with human responses?
 - **No:** very weak correlation between BLEU scores and human judgments!
- ▶ But alternative metrics exist, like ADEM



[Lowe et al. (2017). Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In *ACL*.]

[Liu et al (2016). How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *EMNLP*.]

Plan for today

- ▶ Dialogue management
 - Handcrafted approaches
 - Data-driven approaches
- ▶ Design of dialogue systems
 - Architectures
 - Evaluation
- ▶ **Summary**

Summary

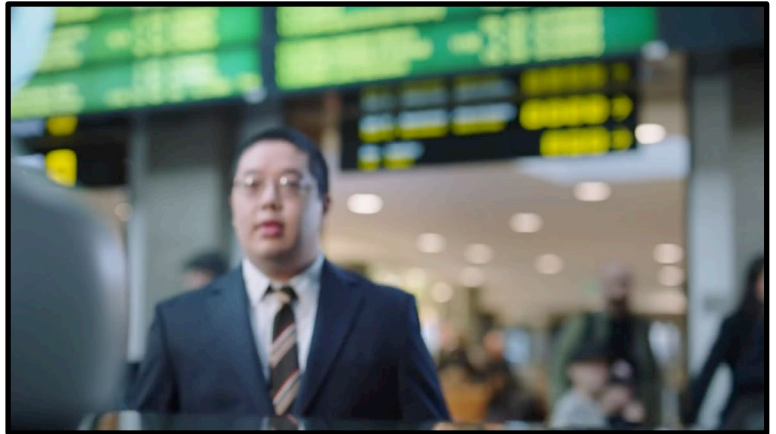
What to say *next* ?



- ▶ Dialogue management = **decide**
what to do/say at a given time, based on:
 - System goals (and trade-offs)
 - Current (uncertain) dialogue state
- ▶ Various approaches:
 - Easiest (but quite rigid): *finite-state* approaches
 - *Frame-based* systems (slightly) more flexible
 - Statistical/neural approaches *optimise* dialogue policies from (real/simulated) interactions
- ▶ Evaluation via objective and subjective metrics

What we haven't covered

- ▶ Natural language generation (NLG)
- ▶ Speech synthesis
- ▶ Multimodal & situated systems



Furhat robot (initially developed at KTH, Stockholm), see www.furhatrobotics.com