Chatbots models (continued)

Pierre Lison

IN4080: Natural Language Processing (Fall 2023)

24.10.2023

Plan for today

- Obligatory assignment
- NLU-based models
- Generative models
- Speech recognition
- Summary

Plan for today

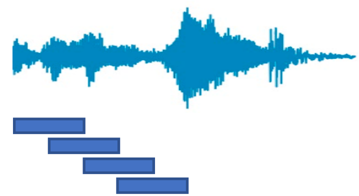
- **Obligatory assignment**
- NLU-based models
- Generative models
- Speech recognition
- Summary

Oblig 3

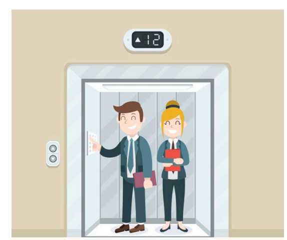
Three parts:

1. Chatbot based on movie and TV subtitles

2. Silence detector in audio files



3. (Simulated) talking elevator



Oblig 3

- Deadline: November 6
  - Concrete delivery: **Jupyter notebook**
  - Text explanations in the notebook as important as the code itself!
- Don’t hesitate to ask questions during the group sessions
  - we are here to help!

Plan for today

- Obligatory assignment
- **NLU-based models**
- Generative models
- Speech recognition
- Summary

Chatbot models: recap

- Rule-based models:

```
if (some pattern match X on user input)
then respond Y to user
```

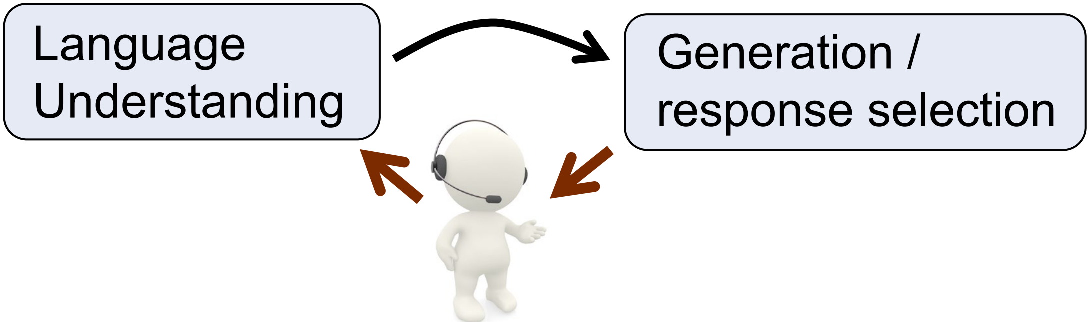
- IR models using cosine similarities between vectors

$$r = response\left(\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{||q|| \, ||t||}\right)$$

Where C is the set of utterances in dialogue corpus (in a vector representation)

and q is the user input (also in vector form)

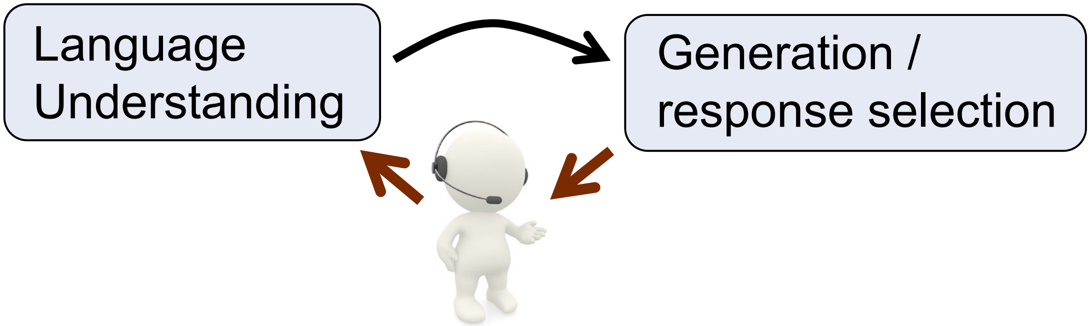
NLU-based chatbots



Can we build data-driven chatbots for task-specific interactions (not just chit-chat)?

- "Standard" case for commercial chatbots
- Typically: no available task-specific dialogue data

NLU-based chatbots

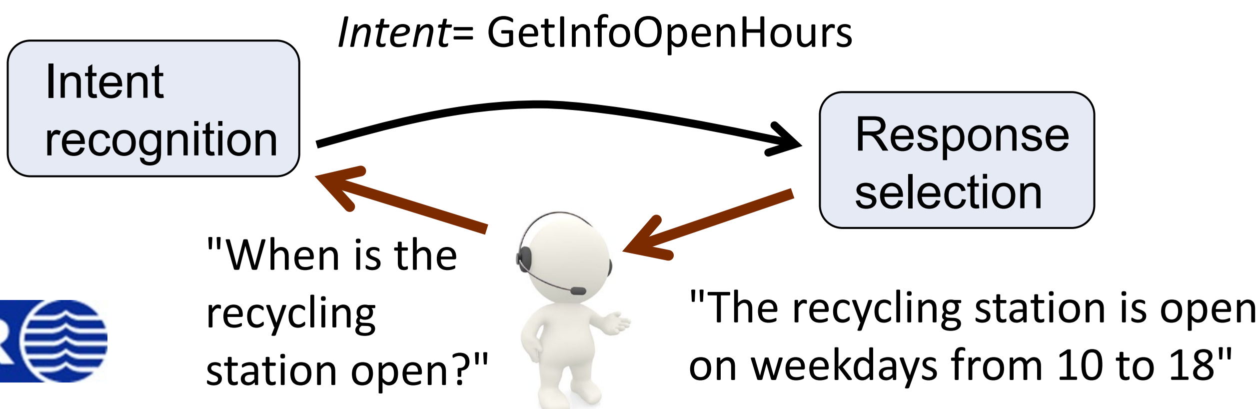


- Solution: NLU as a **classification task**
  - From a set of (predefined) possible **intents**
- Response selection generally handcrafted
  - Chatbot owners want to have control over what the chatbot actually says

Intent recognition

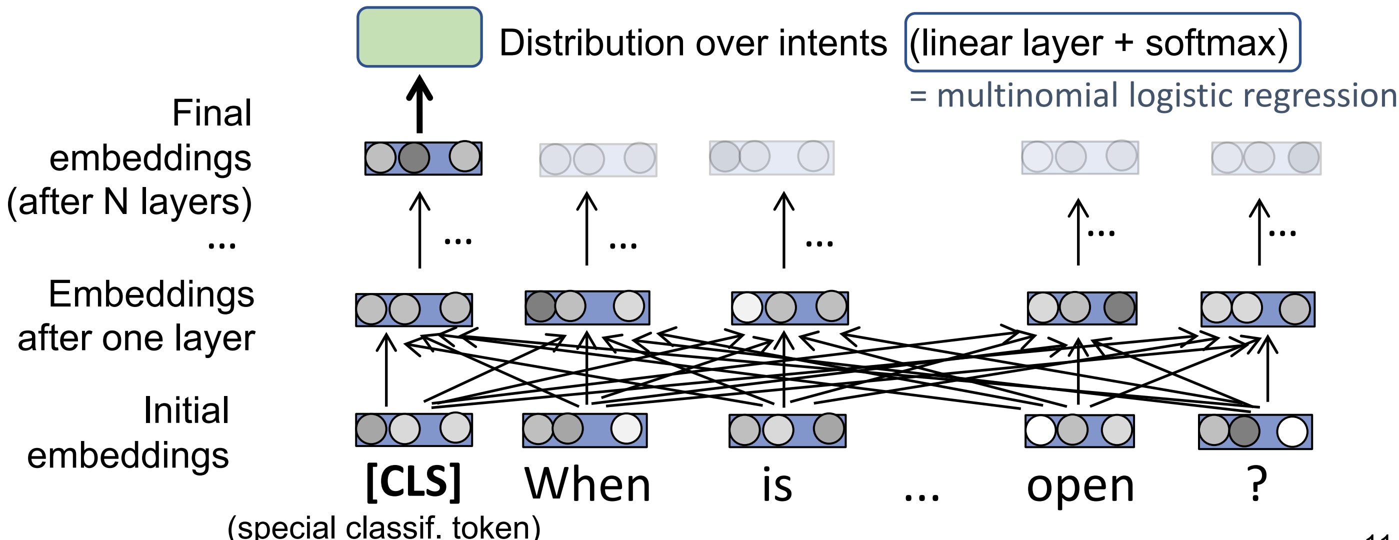
Goal: map user utterance to its most likely intent

- *Input*: sequence (of characters or tokens)
  + possibly preceding context
- *Output*: intent (what the user tries to accomplish)



Intent recognition

- Many possible machine learning models
  - Very often: LLM with classification head
- Example using BERT:



Intent recognition

- Need to collect *training data* to learn this classification model
  - *Data*: user utterances (+ context) manually annotated with their intent(s)
  - Often annotated by "chatbot trainers" in industry
- Standard approach these days:
  - Take a pre-trained neural language model (i.e. NorBERT for Norwegian)
  - *Fine-tune* it for this specific classification task

Slot filling

- In addition to intents, we also sometimes need to detect specific entities ("slots"), such as mentions of places or times

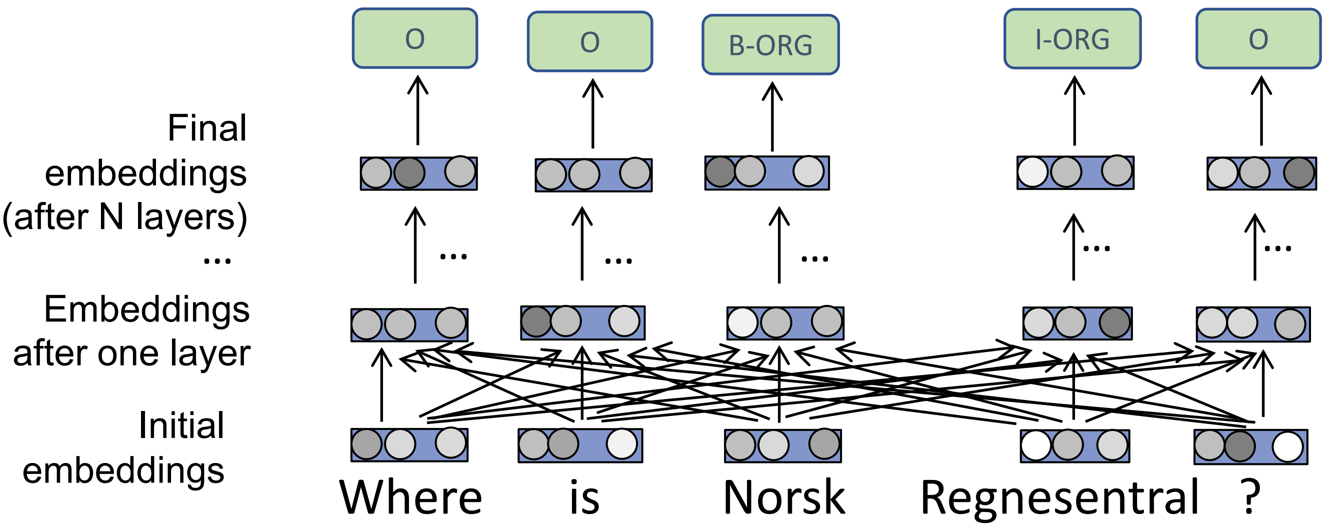
«Show me morning flights from Boston to San Francisco on Tuesday» →

DOMAIN:	AIR-TRAVEL
INTENT:	SHOW-FLIGHTS
ORIGIN-CITY:	Boston
ORIGIN-DATE:	Tuesday
ORIGIN-TIME:	morning
DEST-CITY:	San Francisco

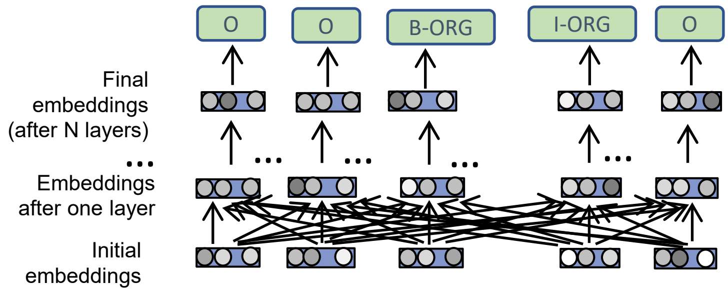
- Slots are domain-specific
  - And so are the ontologies listing all possible values for each slot

Slot filling

Can be framed as a *sequence labelling task* (as in NER), using e.g. **BIO** schemes



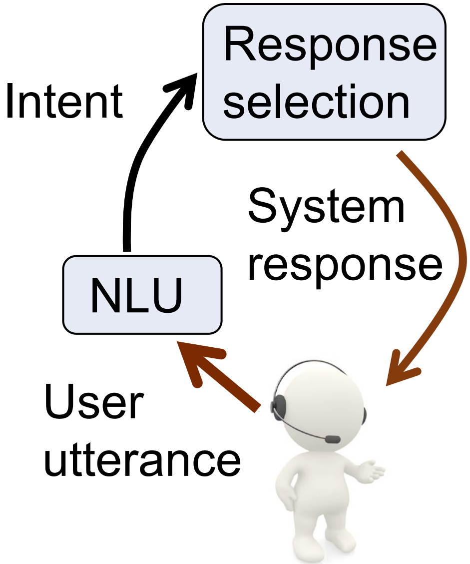
Slot filling



- Token-level classification task
    - Output classes: BIO-prefixed categories
- Slot-filling models also need to be trained / fine-tuned on annotated training data
- Possible to fine-tune intent classifier and slot filler on same model

Response selection

- Given an intent, how to create a response?
- In commercial systems, system responses are typically written by hand
  - Possibly in templated form, i.e. "{Place} is open from {Start-time} to {Close-time}"
- But data-driven generation methods also exists



[see e.g. Garbacea & Mei (2020), "*Neural Language Generation: Formulation, Methods, and Evaluation*"]

Plan for today

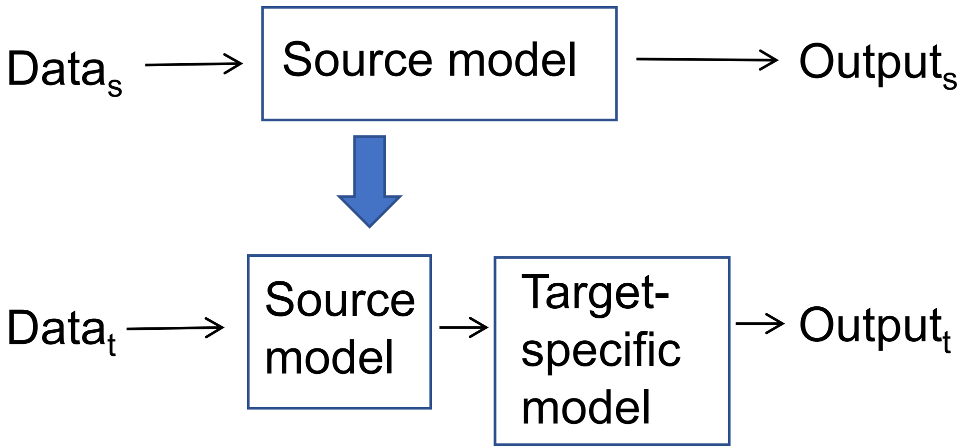
- Obligatory assignment
- **NLU-based models**
  - **Small amounts of data?**
- Generative models
- Speech recognition
- Summary

Small amounts of data?

1. Use *transfer learning* to exploit models trained on related domains

Source domain (with large amounts of training data)

Target domain (with small amounts of training data)



Fine-tuning of a pre-trained language model is a type of transfer learning

Small amounts of data?

1. Use *transfer learning* to exploit models trained on related domains

2. Use *data augmentation* to generate new labelled utterances from existing ones

"**When** is the recycling station open?" → GetInfoOpenHours

↓ Replace with synonyms

"**At what time** is the recycling station open?" → GetInfoOpenHours

Small amounts of data?

1. Use *transfer learning* to exploit models trained on related domains

2. Use *data augmentation* to generate more utterances from existing ones

3. *Label more data*, either manually or using weak supervision techniques

[see e.g. Mallinar et al (2019), "Bootstrapping conversational agents with weak supervision", IAAI.]

Small amounts of data?

1. Use *transfer learning* to exploit models trained on related domains

2. Use *data augmentation* to generate more utterances from existing ones

3. *Label more data*, either manually or using weak supervision techniques

4. Use *in-context learning* to provide examples *as part of the prompt*

Plan for today

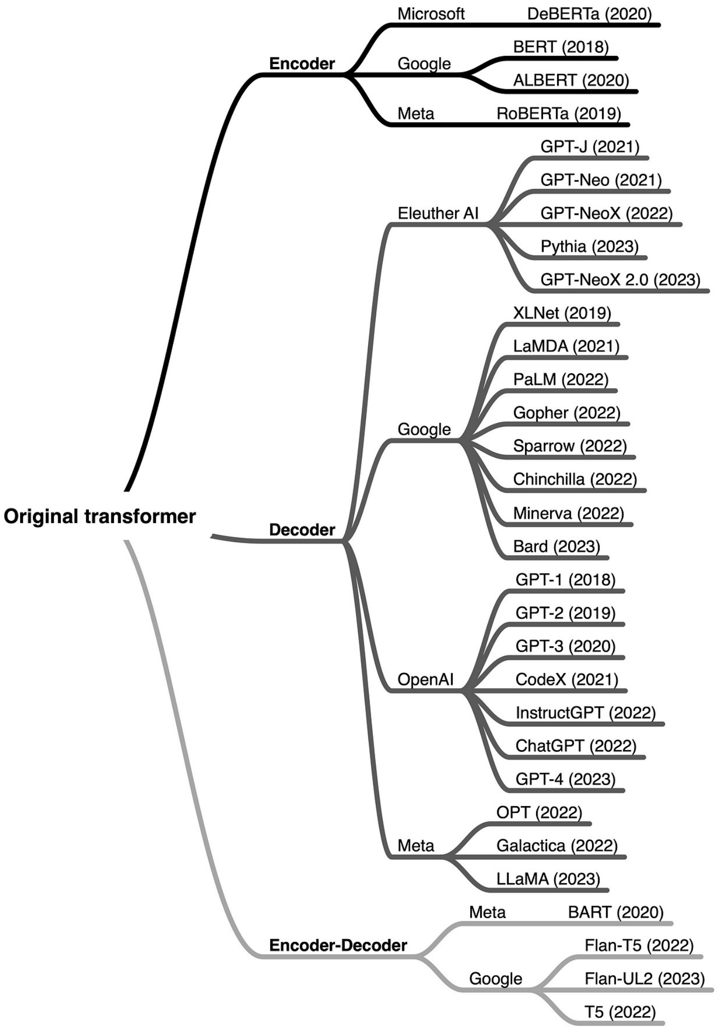
- Obligatory assignment
- NLU-based models
- **Generative models**
- Speech recognition
- Summary

Generative models

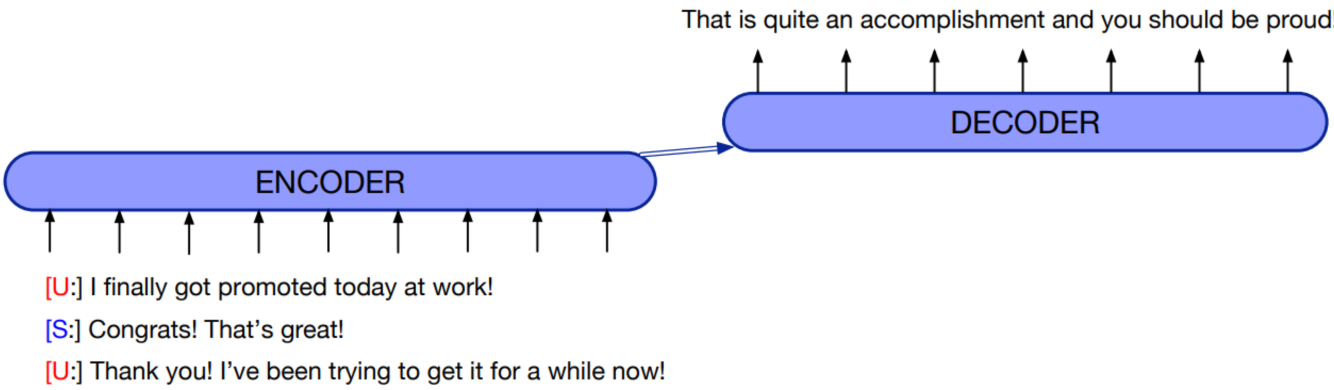
- Sequence-to-sequence models *generate* a response token-by-token
  - Akin to machine translation
  - Can generate new responses never observed in the corpus
- Two steps:
  - First «encode» the input with a neural model (=tokenise the input and extract the vectors for each token)
  - Then «decode» the output token-by-token (based on the input vectors and the output produced so far)

Generative models

- **Encoder-decoder models** (i.e. T5)
  - self-attention + cross-attention
  - Popular for tasks likes MT and summarization
- **Decoder-only models** (i.e. GPT models)
  - Has become the dominant approach



Generative models

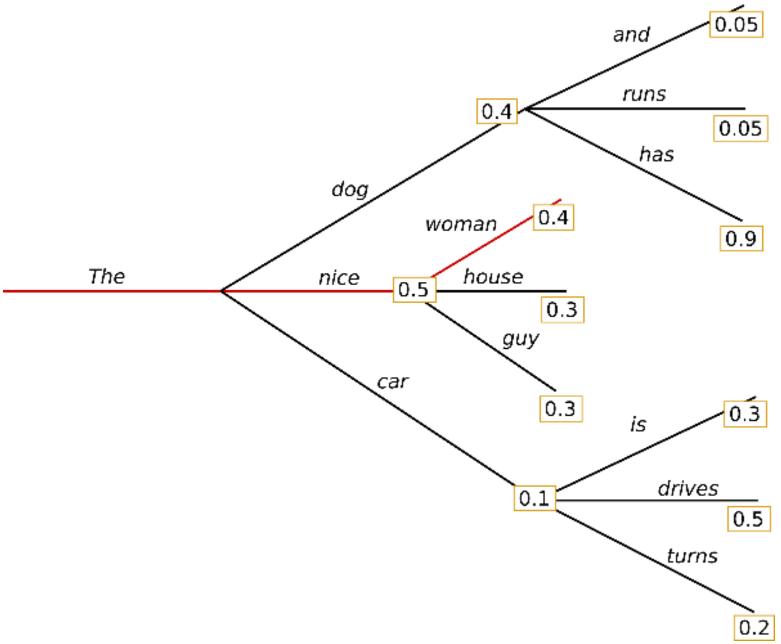


For decoder-only models, the encoder and decoder are the same (self-attention to all tokens from the start of the context window up to the current token)

[Example from Jurafsky and Martin, chap. 15]

Decoding

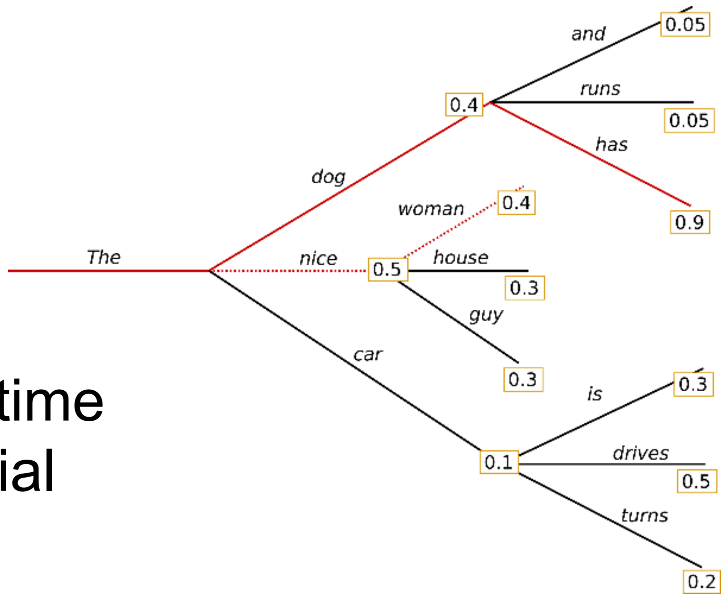
- Greedy



[Illustration borrowed from https://huggingface.co/blog/how-to-generate]

Decoding

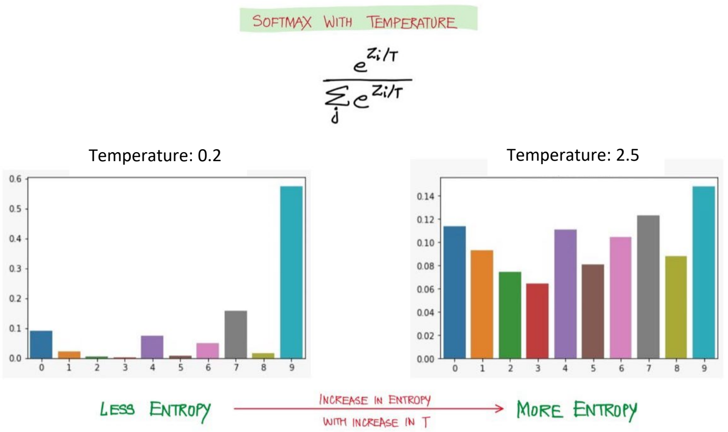
- Greedy
- Beam search
  - Keep at each time a set of K partial hypotheses
  - And expand these until <eos>



[Illustration borrowed from https://huggingface.co/blog/how-to-generate]

Decoding

- Greedy
- Beam search
- Sampling
  - Temperature controls the “creativity” of the response
  - Lower temperature = *sharper* distribution (increase likelihood of high probability words and decrease the likelihood of low probability words)



[Illustration from https://medium.com/mlearning-ai/softmax-temperature-5492e4007f71]

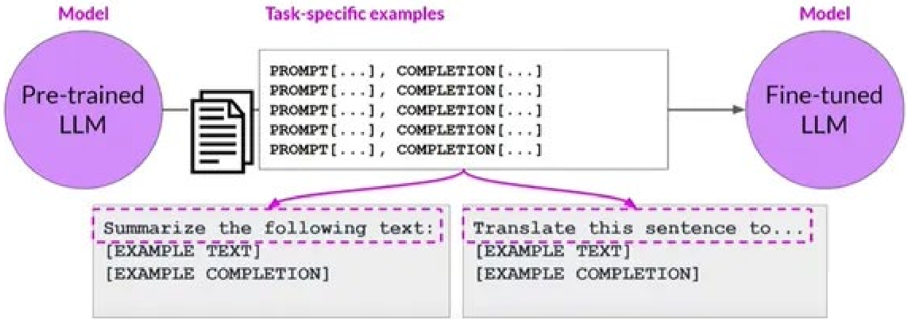
Decoding

- Greedy
- Beam search
- Sampling
- Top-K sampling
  - = select the K tokens with highest probability, redistribute the probability mass among them, and sample from that distribution

[Illustration from https://medium.com/mlearning-ai/softmax-temperature-5492e4007f71]

Instruction fine-tuning

- Systems like ChatGPT are not raw LLMs, they are specifically *fine-tuned* to follow instructions and/or engage in a dialogue with the user
- Many open-source LLMs have downloadable models that are instruction fine-tuned



Domain adaptation

Imagine you wish to build a generative chatbot for your domain. How do you proceed?

Easiest approach: **in-context learning**

You just add examples of <input, response> pairs as part of the prompt, and ask the model to answer like in the examples

Limitations:

→ Can only include a small number of examples (needs to fit in the context window)

→ Slow (needs to encode a longer context)

Ok for a prototype, but limited domain adaptation

Domain adaptation

Other techniques:

- Parameter-efficient fine-tuning (*PERT*)
  - LoRa: small number of learned parameters (millions) on top of the original frozen ones

    [Hu, E. J et al (2021). *LoRa: Low-rank adaptation of large language models*.]

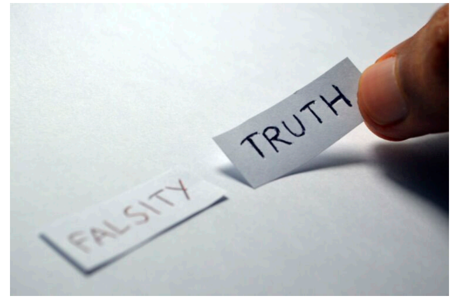
- *Prompt tuning*: search for the best possible prompt, keeping the model frozen
  - Can be a *soft* prompt (i.e. prefixed vectors instead of actual words)

[Liu et al (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, *55*(9), 1-35.]

Plan for today

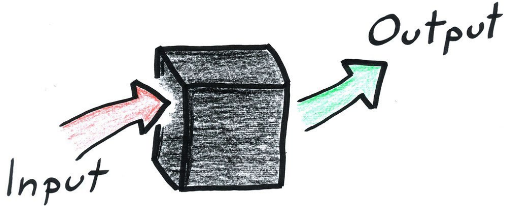
- Obligatory assignment
- NLU-based models
- **Generative models**
  - **Challenges and «hot topics»**
- Speech recognition
- Summary

Challenge 1: Factuality



- Large Language Models are optimized to produce *plausible* texts, not necessarily *correct* ones!
- Incorrect responses may come from the training data, which can contain errors / disinformation…
- … But language models may still hallucinate with a "perfect" training set!
- And often do so in an overly *confident tone*

Challenge 2: Control



- LLMs are “black-boxes”:
  we don’t really understand *why*
  they generate a given response
- We can “steer” the model in several ways:
  - *Prompting* with specific instructions
  - *Fine-tuning* on task-specific data
  - *Reinforcement learning* (reward good responses and punish bad ones)

➔ **But the model may still behave unpredictably**

- Side problem: How to delete information from a language model? (cf. GDPR’s *right to be forgotten*)

Multimodality

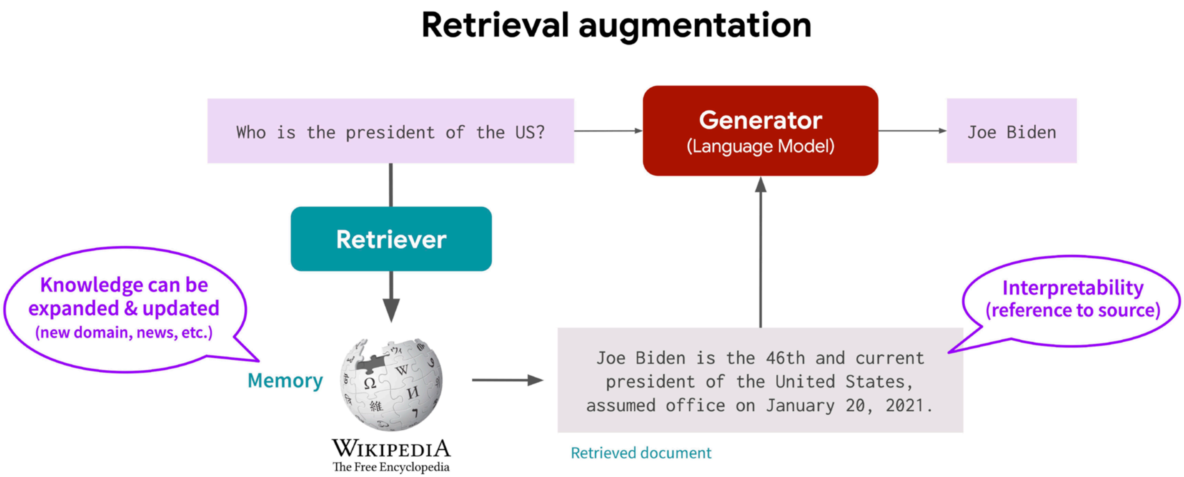
- *Multimodal* generative models are increasingly popular
  - Can be used for e.g. visual QA (ask questions based on an image)
- Development of «embodied» models
  - Grounding of linguistic inputs with real-world sensory inputs

Retrieval-augmented models

What if the answers need to rely on a knowledge base (corpus of documents, such as Wikipedia pages)?

- If the knowledge can fit into the context window, you can include it in the prompt
- Or use *retrieval-augmented models* which combines two neural models:
  - **Retriever**: selects relevant docs from the knowledge base
  - **Generator**: generates the answer, given the initial prompt *and the retrieved documents*

Retrieval-augmented models



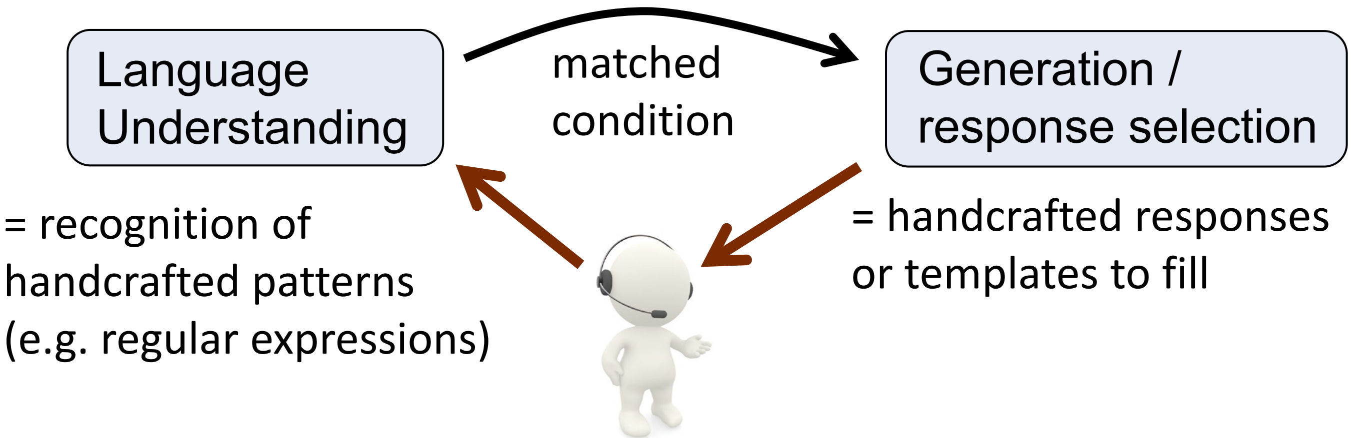
Benefits:

- Knowledge base can be easily inspected and updated (just add or remove documents)
- Can help reduce hallucinations

Summary

How to develop a chatbot:

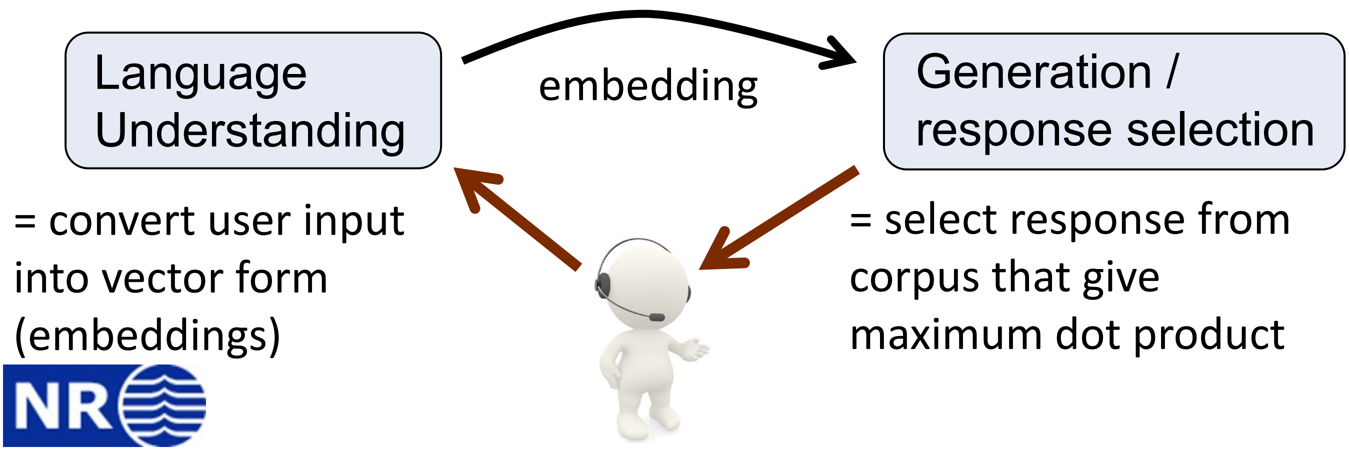
- **Rule-based approaches**



Summary

How to develop a chatbot:

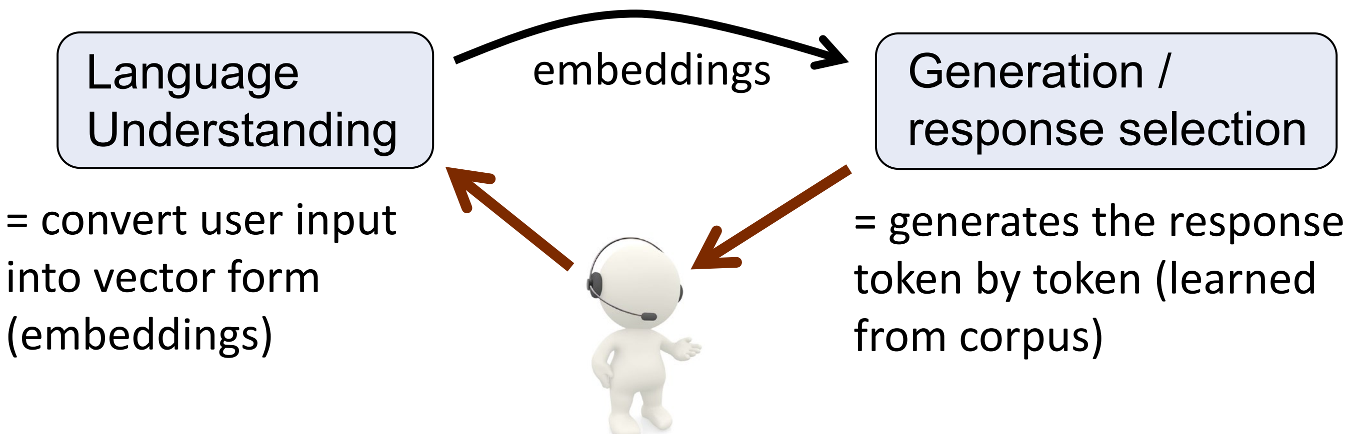
- Rule-based approaches
- **IR-based approaches**



Summary

How to develop a chatbot:

- Rule-based approaches
- IR-based approaches
- **Generative approaches**



Summary

How to develop a chatbot:

- Rule-based approaches
- IR-based approaches
- Generative approaches
- **NLU-based approaches**

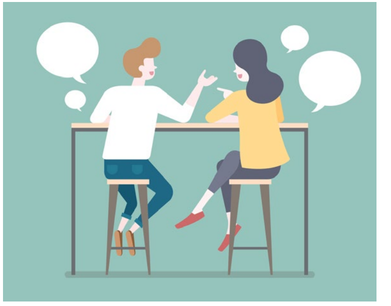
Often useful to rely on a combination of techniques – such as doing intent recognition using both rules and ML

Language Understanding → Intent + slots → Generation / response selection

=map utterance to an intent + slots

= handcrafted response or template to fill

Next week



- Next week, we'll talk about *dialogue management* – that is, how do we control the flow of the interaction over time?
  - Including how to optimise dialogue policies using reinforcement learning
- And we will also talk about how to *design* and *evaluate* dialogue systems