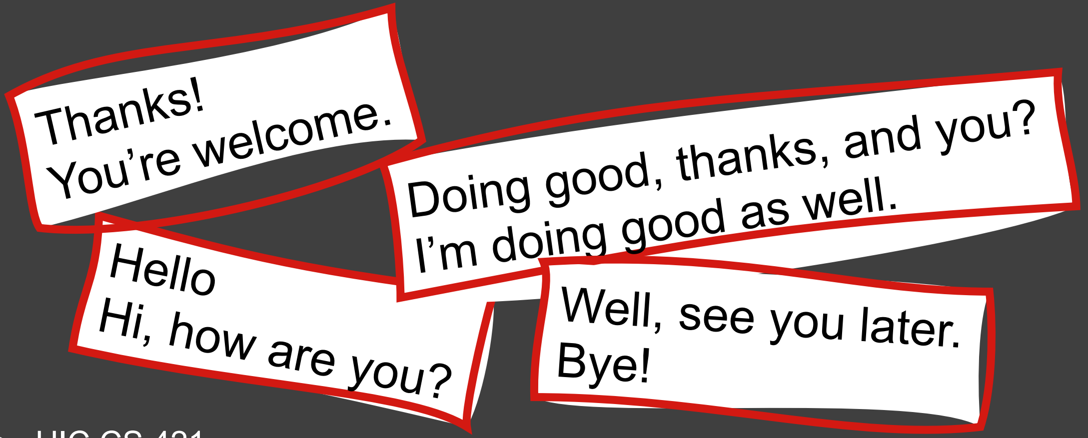
Corpus-Based Chatbots

Natalie Parde
UIC CS 421

Corpus-based Chatbots

- No manually created rules
- Instead, learn mappings from inputs to outputs based on large human-human conversation corpora
- Very data-intensive!
  - May require hundreds of millions, or even billions, of words

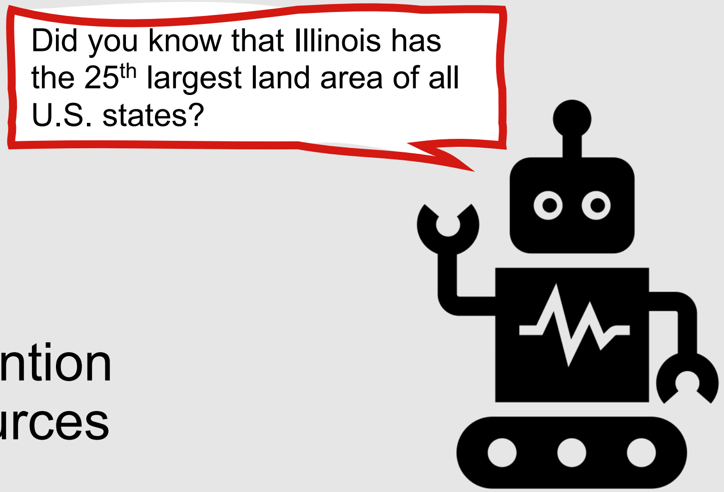


What kind of corpora are used to train corpus-based chatbots?

- Large spoken conversational corpora
  - Switchboard corpus of American English telephone conversations: https://catalog.ldc.upenn.edu/LDC97S62
- Movie dialogue
- Text from microblogging sites (e.g., Twitter)
- Collections of crowdsourced conversations
  - Topical-Chat: https://github.com/alexa/alexa-prize-topical-chat-dataset

Possible responses can also be extracted from non-dialogue corpora.

- Possible sources:
  - News
  - Online knowledge repositories (e.g., Wikipedia)
- This allows the chatbot to tell stories or mention facts acquired from non-conversational sources



As humans interact with a chatbot, their own utterances can be used as additional training data as well.

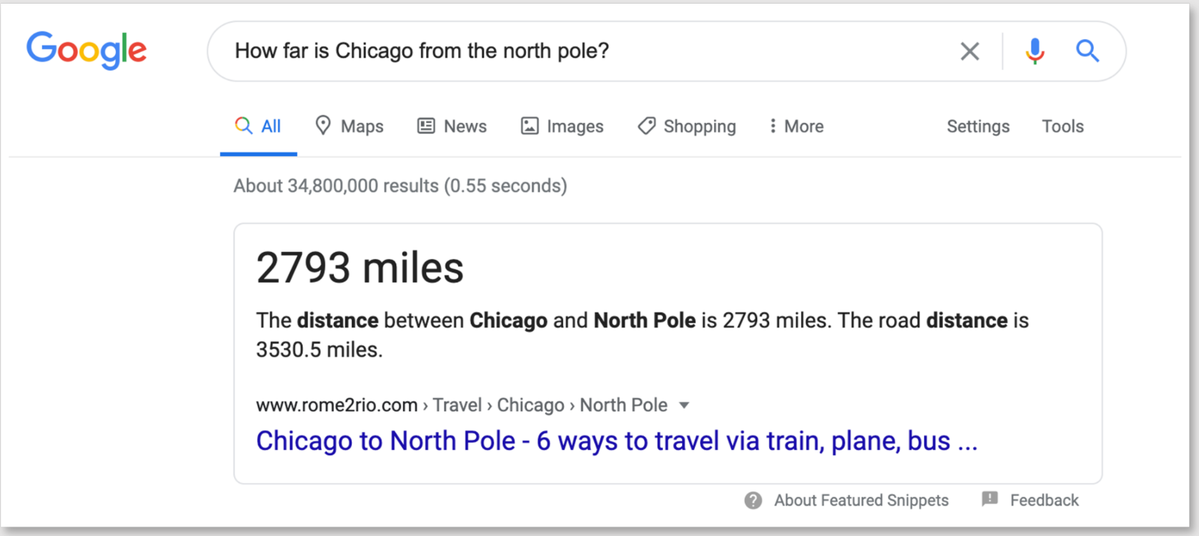
- This allows a chatbot's quality to gradually improve over time
- Some privacy concerns can emerge when using this strategy (it's crucial to remove **personally identifiable information**!)

Corpus-based Chatbots

- Two main architectures:
  - **Information retrieval**
  - **Machine learned sequence transduction**
- Most corpus-based chatbots do (surprisingly!) very little modeling of conversational context
- The focus?
  - Generate a single response turn that is appropriate given the user's immediately previous utterance(s)

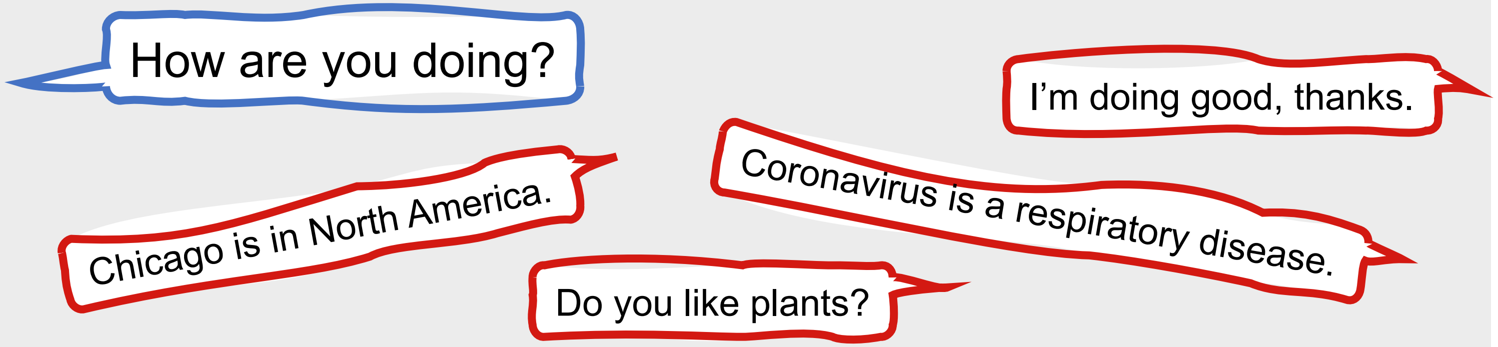
Corpus-based Chatbots

- Minimal contextual awareness → many corpus-based chatbots may be viewed more as **response generation** systems
- This makes them similar to **question answering systems**:
  - Focus on single responses
  - Ignore larger conversational goals



Information Retrieval-based Chatbots

- Respond to a user's turn by **repeating some appropriate turn from a corpus** of natural human conversational text
- Any information retrieval algorithm can be used to **choose the appropriate response**
- Two simple methods:
  - Return the response to the most similar turn
  - Return the most similar turn



How can we return the response to the most similar turn?

- Look for a turn that resembles the user's turn, and return the human response to that turn
- More formally, given:
  - A user query, *q*
  - A conversational corpus, *C*
- Find the turn *t* in *C* that is most similar to *q* and return the human response to *t*
  - $r = \text{response}(\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{\|q\| \|t\|})$

How can we return the most similar turn?

- Or, directly match the user's query, *q*, with turns from *C*, since a good response will often share words or semantic patterns with the prior turn
- More formally:
  - $r = \underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{\|q\| \|t\|}$

Various techniques can be used to improve performance with IR-based chatbots.

- Possible additional features:
  - **Entire conversation** with the user so far
    - Particularly useful when dealing with short user queries, e.g., "yes"
  - **User-specific** information
  - **Sentiment**
  - Information from **external knowledge sources**

$$\boxed{\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{\|q\|\|t\|}}$$

Encoder-Decoder Chatbots

- **Machine learned sequence transduction:** System learns from a corpus to **transduce a question to an answer**
  - Machine learning version of ELIZA
- Intuition borrowed from **phrase-based machine translation**
  - Learn to convert one phrase of text into another
- Key difference?
  - In phrase-based machine translation, words or phrases in the source and target sentences tend to align well with one another
  - In response generation, a **user's input might share no words or phrases with a coherent, relevant response**

How does a chatbot learn to perform this transduction?

Encoder-decoder models

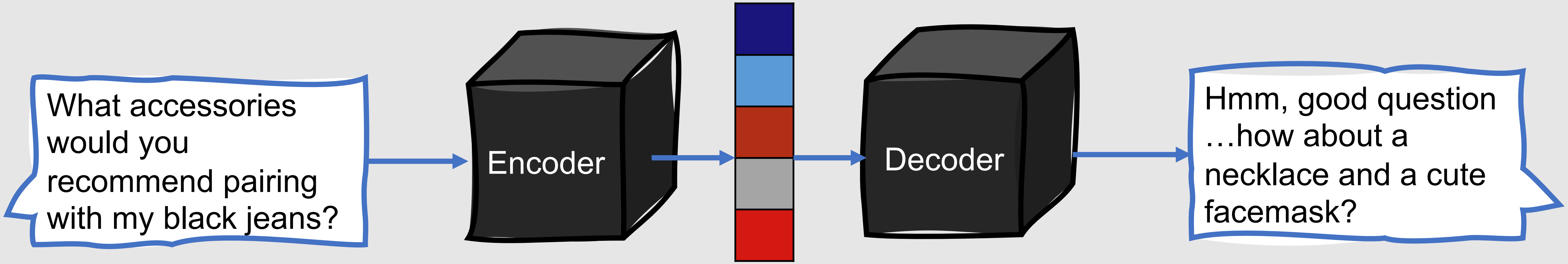
- Accept sequential information as input, and return different sequential information as output

Also recently used in:

- Machine translation
- Question answering
- Summarization

How do encoder-decoder models work?

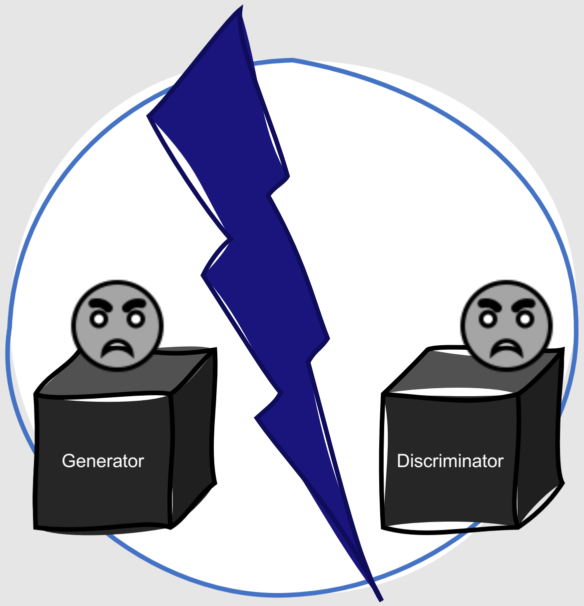
- In NLP applications, encoders and decoders are often some type of **recurrent neural network**
- Encoders take sequential input and generate an **encoded representation** of it
  - This representation is undecipherable to casual observers!
- Decoders take this representation as input and generate a sequential (interpretable) output



Encoder-Decoder Chatbots

- Basic encoder-decoder models tend to produce repetitive (and therefore boring) responses that don't encourage further conversation
  - "I'm okay"
  - "I don't know"
- To avoid this, it is important to incentivize response diversity
  - Mutual information objective function
  - Beam search

Encoder-Decoder Chatbots



- Other challenges?
  - Inability to model prior context
    - Can be solved by using a **hierarchical model** that summarizes information over multiple turns
  - Often poor multi-turn coherence
    - Can be addressed to some extent using **reinforcement learning** or **adversarial networks** to learn to choose responses that make the overall conversation more natural