# **Introduction and learning outcomes**

So welcome to topic nine in which we're

going to talk about dialog systems. So, to get us started, let's just briefly

revisit our course learning objectives. And of the five of those there are three

that are particularly relevant to this topic. So the number one which is we're going

to look at differences in rule based and statistical approaches. And that's very much

true of dialogue systems. We're also going to be using various

software tools and analyzing dialogues and drawing on various NLP libraries for

building our own chatbots. So we're looking at

objective number three, and we'll also briefly touch

on the topic of evaluation. So that also brings in

objective number five. So to look at our topic learning

objectives, designing dialogue systems is partly a technical task, but

it's also partly a linguistic task. It's also partly a task of

studying human behavior. So this a strong element of psychology and a strong element of human computer

interaction involved in this topic. So we're going to take a little bit

of a step back at the start and think about the properties

of human conversation. Because if we want chatbots or

other automated agents to emulate human conversation, we need to

understand how human compensation works. So that's it.

We'll start and then we'll move on to talk

about various architectures for building dialogue systems and

we'll take in the second half. We'll also take the opportunity to do a

little bit of practical work building our own chatbots. So there it is. Let's get started.

# **Properties of human conversation**

In this segment, we're

going to look in a little bit of more detail at human conversation and study

some of the properties that make human conversation

unique and interesting. We're going to start off by looking at a sample

dialogue just so as we really are all on the

same page as to what we mean by dialogue and

dialogue systems. Here's a fragment

of a conversation between two individuals and this conversation is all about trying to book

essentially a flight, trying to make travel

arrangements between a customer and a travel agent. You'll see actually even in just this fragment

of dialogue that, it's actually a very nuanced. We get lots of interesting

linguistic phenomena and we get lots of

interruptions and lots of other ways in which

the two individuals interact that go beyond just simply making statements

towards each other. If we actually look in a

little bit of closer detail, we see that human conversation

always is essentially a sequence of turns between

these two individuals. The question really is, what are the rules

of that exchange? How do we know when to stop talking in a conversation and how do we know when

to start talking? There's actually

some very subtle linguistic and social cues that govern those decisions. Part of it obviously

is detecting an interruption is appropriate

way when you should stop talking and also there's a more subtle one of knowing

when to start talking, when to detect the

other individual has completed their turn. Because sometimes a pause

is not strong enough queue there are

also other elements that indicate closure

of an idea and a handing over of the turn to the other individual

in that conversation. You can also see that thinking about that

little fragment of conversation that we saw that what's happening is that each utterance

plays a role in that exchange and we can analyze what those

actual roles are. There's various taxonomies

that have been proposed for analyzing these exchanges. There's one here that

we're looking at and it's called speech acts, which frames each utterance

as action performed by the speaker and there

are four types of these. There are constatives which essentially commit the

speaker to some proposition. If I was making a claim or confirming something

or denying something, that's a constative Then

there's directives, which are a little bit like

commands which are address to the other individual to encourage or persuade

them to do something. Asking for bidding or inviting. Then there are commissives

which commit the individual, the speaker, to some

future course of action. Promising or planning

or vowing that thing. Then very important part

of human conversation that especially in

polite conversation and something that differentiates

perhaps human conversation for more automated

types of exchange is acknowledgments where

we express a reaction to some action or we

apologize, greet, or thank. You may want to take a moment to pause and ponder and look

at that dialogue that we showed a moment ago and

perhaps have a think about which speech acts are we seeing at a stage

in this exchange. That introduces another

important quality of human conversation, which is that dialogue is not

just an exchange of turns. It's a collective

act working towards some common goal and it's

characterized by cooperation. The way in which that happens is by establishing common ground

between the participants. Have you ever had that feeling

where you're just not on the same wavelength or

to use an expression, you're not on the same

page with somebody. Very often it's because

you're talking across each other and you

haven't established that common ground between

the two individuals. This is called grounding and

it takes place via a number of activities that have

surfaced in that conversation. Essentially they're all about

acknowledging that not only that the hearer has

perceived utterance, but also understood

the implications of what the previous

speaker said. There's an analogous principle in intimate future interaction. If you've ever

pressed the button on a lift or an elevator, you'll see it lights up. That is in a way of grounding

and it's response to say that the system has

acknowledged and understood your intent

in your interaction. Certainly when you mouse over

a button in an interface, you may see it change color

again to indicate that it is interactive and that it has detected the presence

of the mouse movement. Again, similar

thing with a click sound when you press

a key, keyboard. Again, it's providing

feedback that your input has been not only

perceived but understood. This is a very

important phenomenon in human conversation and there's various ways that

grounding takes place. There's explicit grounding

where people say okay or all right or understood. Then there's more implicit

one which is where the following term will

echo or repeat back some fragments of

the previous turn indicating that they've understood and have absorbed that into their understanding of the common ground between

the two individuals. It's also important

to notice that some dialogues can have

a nested structure. Sometimes you get

sub-dialogues and this is particularly important

when you get correction. When you see two individuals

discussing something, they're working

towards that goal, but occasionally there

is a misunderstanding and then that misunderstanding is repaired through a

process of correction, which is a sub-dialogue, which then establishes the

correct interpretation, then moves on with whatever

the parent task was. Also clarification, which is

very important, of course, for automated systems because

speech recognition well even human recognition is

not 100 percent perfect. When an error takes place, but then needs to be a

clarification dialogue to repair that error

and put it right. That introduces another

important quality of human conversation, which is the initiative. Now, most of us have

been in a situation where we've either

witnessed an interview or being in an interview. Well depending on the interview, an interview tends

to be controlled by or can be controlled

by a script. If you imagine an interview

perhaps on TV where somebody is being questioned

in front of the camera, that dialogue is

generally controlled by the reporter or the

interviewer and that's what we call a

single initiative dialogue. But most dialogues between individuals are

mixed initiative. In other words the initiative goes backwards and forwards

between the two individuals. But that's a much more

difficult design challenge, as you'll see in

subsequent videos. It's much simpler to offer

a single initiative. You could have a user

initiative dialogue. For example, search engines, they sit there and they wait, you type in a question and

then respond with an answer, or you type into a search

query and they respond. That's the user initiative. They don't do anything until

they're spoken to you, a little bit like most

interactive systems that we have in the home, with some slight exceptions, they wait to be spoken to and when you speak to

them then they react. That's a user

initiative dialogue, but there's also system

initiative dialogues. Again, we've all had that

feeling when we call up to talk to the

bank or talk to some other utility

company or some of the systems which

we're subscribed and we find ourselves lost

in a cool system, often very frustrating, and we can't break

out of it because the initiative is wholly

led by the system. We can only ever react and respond to the prompts

that are given to us. Those prompts are fixed

unfortunately can't break out of it unless we find a way to

speak to a human agent. Finally, we should

also notice that dialogues rely on inferences. Its back to that

common ground idea and a shared mental model. This leads onto the idea of

what's called conversational implicature and some

work by a linguist, Paul Grice in the '70s, where he proposed various maxims that govern and shape

human conversation. For example, quantity that

in each utterance that the speaker should say enough but not too

much to communicate, to continue that dialogue and

quality that it should be truthful and banked by some evidence or

common ground relation that it should be relevant. We've always had that feeling

where somebody goes off, on a tangent, is off topic. They've broken that

relevance maxim and also manner that it should be

clear and avoid ambiguity. These are what we call

Grice's maxims and they help to govern

and shape and make effective

dialogues and they're all part of helping

us understand what makes human

conversation efficient. That's our brief look at the properties of

human dialogues. In the next few videos, we'll start to look a little bit more detail

about how we might automate this and what the architectures might look like for

automatic dialogue agents.

# **Properties of human conversation practice quiz**

### Question 1

Which of the following are types of speech act?

* Commissives
* Directives
* Sub-dialogues
* Constatives

### Question 2

Simple question answering systems exhibit which kind of dialogue?

* User-initiative
* System initiative
* Mixed initiative

# **Chatbots**

In this segment we're

going to continue looking at dialogue

systems and we're going to shift our

attention onto chatbots. Now, in case you're

wondering what a chatbot is, and I'm guessing

you're probably not because two or three years ago, everybody was talking about them and they were hyped

a little bit, but they do seem to have found a place in the home

and in the office. Most of us are familiar

with the conversations that we see now on the screen. Sometimes they're

productive, sometimes not so productive,

but you get the idea. There's my responses

on the right in blue and there's the

chatbots on the left there. You can see that this

conversation starts off okay, but rapidly doesn't

really go anywhere. But in a way, that's

what chatbots are for. They're not really necessarily designed for completing

specific tasks or goals, chatbots are more about

the conversation. The conversation

is the end goal. They're simple dialogue systems, that they're in some ways

for entertainment value, some of them have

a little bit more, what you might call utility, but in lot of cases, it's chat for chat's sake. But we shouldn't

underestimate that, that can have therapeutic value. That can also, as I said, be an introduction

to other more tasks or into utilitarian exchanges. There's been various

examples that have been popularized, you may have heard of

Facebook's BlenderBot or Microsoft's Xiaoice, I think it's pronounced. There's all sorts of

chatbots that get deployed on websites and Slack, and Facebook Messenger and

all sorts of other channels. You're probably

familiar with the user experience of a chatbot, but you might not be

quite so familiar with what is actually happening under the hood and how

they're constructed and what kind of architecture

is involved. To start off, to start

to look at that topic, we should first of all

point out that there are primarily two

classes of architecture, like a lot of things in NLP. There's a rule-based

approach and then there is a more data-driven or

corpus-based approach. In the rule-based

approach, we'll be looking at something

called ELIZA, which was the

first-ever chatbot. We'll be looking also at various corpus-based approaches, which as you might imagine, are all about mining

large datasets and using, in our case, information

retrieval techniques, with possibly encoder-decoder

based approaches to facilitate that dialogue. I mentioned ELIZA. This actually has

a long history. We talked about it very

briefly in topic 1 of NLP. It was put together by

Joseph Weizenbaum in 1966, so a long time ago. It was really the first chatbot. You can see a transcript of

an exchange here with ELIZA. ELIZA was what we call a

Rogerian psychotherapist, which basically means

that they reflect back to the patient

or participant, their own thoughts and utterances in a way that allows

them to find, I suppose, solutions or a therapeutic

value in that conversation, as opposed to

dispensing what you might call clinical advice. You can see an exchange here. ELIZA, "Is something

troubling you?". YOU, "Men are all alike". Actually if you look

at this exchange, it's actually quite

coherent, quite human. It really is quite remarkable

that this exchange took place as long ago as 1966. How does it work under the hood? Well, its rules. You can see rules here such as, this is essentially a RegEx, a regular expression, where the star indicates

some matching string, we're matching something,

you something, me, that's the

pattern, and then you get what's called

production rule, which transforms it into what

makes you think I 3 you, where 3 is one of the things that matched in the

previous pattern. For example, if you say

something like you love me, a transform can be

applied, which says, what makes you think I

and then the wildcard is instantiated to the string that matched in the pattern,

which is the word love. So it says, "What makes

you think I love you?" These transformations, although

they're fairly trivial, they actually are remarkably effective in creating

that dialogue. There's more to it than

that. The rules are actually linked to keywords, so things like I and my, and everybody, and family, and all sorts of keywords which you can actually create

your own if you want. But there's a set that

you get by default in out of the box, ELIZA. There's rules attached

to these keywords and they're ranked by specificity. You may recall we

talked earlier about Grice's maxims when we

talked about relevance. In a way, this is

an instantiation of one of Grice's maxims, trying to make sure

that the response that ELIZA gives

is as relevant as possible or at least

as specific as possible to the users options. For example, if

there's an input, I know everybody laughed at me. Two keywords might match that

word I and word everybody, but everybody's ranked

higher than I because it provides a more specific and

interesting transformation. In this case, ELIZA responds with who in particular

are you thinking of instead of some response to space around the keyword I. Of course, occasionally

nothing matches, in which case ELIZA gives a non-committal

response like please go on or tell me more

or something like that. It also has another

trick up its sleeve, which is to use a memory

of previous utterances. The keyword my triggers, the storage of an utterance

with a transformation. When ELIZA is stuck for words

because nothing matches, instead of giving a non-committal

responses what it can look out as its

memory or stack of previous transformations

applied to utterances that have

been made earlier in the conversation attached

to the word my so it can bring the conversation

back to a previous topic. Again, that's a

very effective way of maintaining a

coherent dialogue. This approach is still used

today in many chatbots. The complimentary approach is to use a corpus based approach, which is, as we discussed

more data-driven. Here it's the classic

machine learning paradigm. We would train on

a large corpus of human-human conversations

and we could acquire these from call-center transcripts or even movie dialogues

or other sources. We can pre-train on larger

datasets, like for example, use Twitter exchanges

as pseudo dialogue. Sometimes when

chatbots are deployed, the live usage data can also be used retrospectively

for training as well. Of course if you do that, then you need to

be careful that, that live usage data doesn't pollute what would

otherwise have been a better standard

dataset or training data, confidence metrics are then

applied in order to filter out the good data from

the not-so-good data. Then the responses

themselves are generated through usually

one of two approaches. There's retrieval as an

information retrieval type approaches or

generation methods, and we'll look at those two. Response by retrieval,

we're framing it that the task as an

information retrieval problems. The user's turn we think

of it as a query Q. The task of a system is to retrieve the highest

matching response R from a corpus C responses. We can think of this

in classic IR terms, information retrieval

terms using TF, IDF, weighting, cosine measure, and all the things

that we talked about earlier in the course. Mathematically we can think

of it as we're trying maximize the dot-product

over the normalized vectors. We can also do this using

neural IR approaches. It's essentially

the same paradigm, but now we use a

bi-encoder one for the query and one for the

response and as before, we take the dot product

of the vectors. In this framing we normally consider all of the

previous conversation. That's responds by

retrieval there is also a response by generation. In this approach we essentially think of it as an

encoder-decoder task. A little bit like

machine translation. Well, we're trying to transduce from the user's turn

to system turn. We generate each

token by conditioning on the entire query and

the response so far, as indicated in that

little formula there with the probability of the word given the query and

the R responsive so far. In this approach

we would normally, as opposed to the previous one, the retrieval approach

where we'd normally think of the queries as

the last utterance here we would normally

think of the query based on the entire

conversation so far. One of the downsides of

this approach generation, is that it tends

to be a little bit predictable and a little bit conservative in the

responses that are given. Often there's elements of

diversity are built-in to the algorithms so that there's a little bit more variation in the responses that come back. We can also extend this by thinking about

generation responses by retrieving and refining knowledge which may come

from external sources. We've still got the essentially the information

retrieval paradigm, but now we're complementing

the responses from the corpus of responses with external knowledge sources

such as Wikipedia. Using query expansion

techniques, which we talked about earlier

in order to make them more effective matching

process and retrieval. Also hybrid architectures

are possible. You can combine the rule-based and the corpus based approaches. You might, for

example, Bootstrap using templates, and regexes, and then extend using

various classifiers, prompts based on sentiment or other techniques in order to find an appropriate response. There it is. That's a

brief summary of chatbots and in the next topic we'll talk about task-based

dialogue systems.

# **Chatbots practice quiz**

### Question 1

Which of the following are chatbot architectures?

* Task-oriented
* Corpus-based
* Rule-based
* Hybrid

### Question 2

Which of the following methods are used by corpus-based chatbots?

* Response by retrieving and refining knowledge
* Response by keyword matching
* Response by retrieval
* Response by generation

# **Task-based dialogue systems**

So in this segment, we're going to

continue our discussion of dialogue systems and we're going to turn our

attention to task based dialogue systems. So whereas in the previous video we

talked about chatbots, where the goal was essentially the chart itself and

the entertainment value that provided. Task based dialogue systems are much more

utilitarian that they have a function or a purpose, and they help the user

achieve some sort of goal. And the concept here really

goes back quite a long way to a very significant early

dialogue system called gus. Which was designed to help use make

travel arrangements to book a flight in particular, and the example dialogue that

we saw earlier was from that system. And GUS has been very

influential in the design and architecture of task

based dialogue systems. And some of the original ideas from that

system going back many decades are still very relevant and very useful today and

to those key ideas. The notion of frames which is some gus

by actually quite a long way in AI. It's essentially a way of a structured

knowledge representation that in this case is used to represent user intense AI what

the users goal is in that conversation. And the other concept is something

called slots which are essentially a way of managing attribute value

pairs that reside in the frames. So then the purpose or

the the process of task based dialog management becomes one

essentially of filling those slots. And it does start by asking

questions of the user and populating the slots with

the attribute values and then performing any relevant

actions that may be appropriate. And obviously for booking a flight, you

can imagine that there is some transaction that will take place at

the end of that exchange. The way tone of the benefits they provide

is that rather than the exchange being, somewhat formulaic with what city you're

going to where are you going when you want to go? It allows the user to articulate many

slots in one utterance and the system is flexible enough to accommodate those which

will need some more natural dialogue. And bear in mind also that some

tasks may require multiple frames. Obviously if they're the system's very

tightly constraints for specific domains. Booking flights and

maybe one frame might be sufficient. But in many cases, the user might want

also to ask more generic questions about destinations and

flight preferences and so on. In which case other frames would be

involved to handle those things. See some examples here on

the footer of this slide. We've got examples of slots and the entity

type that goes with those slots and then the sort of question that the system might

trigger in order to populate that slot. So that means that natural language

understanding essentially based around three key tasks, so

there's three parts of this process. The first which is domain classification. As I mentioned, some system

dialogue systems are very simple. They're constrained to one domain but

more often than not the the automated assistance that we

have on our smartphone and in the home. Are actually quite versatile and

quite capable of holding conversations or interacting on more than one topic. So the first task is to

identify that topic or domain. And the second one is to determine the

intent, whether it's to book or cancel or confirm reservation or

something more complex. And then to acquire that necessary

information assign entities to the slots. So for example, if we articulate

the statement wait me tomorrow at 7, the dialog system would need to

identify the domain was alarm clock. Intent was to set the alarm, and

it will populate the slot for the time with whatever today's date and

time is. And then the matching essentially

takes place with matching with the utterance with the intent. The termination is done using

rules using semantic grammars, a little bit like grammars that we looked

at earlier and regular expressions and other hand built production rules. Which as many things in natural language

processing can lead to very high precision. Very accurate systems but

they're expensive. Creating rules requires

a lot of knowledge. A lot of technique and

it's difficult to scale, and there can also be low recall as well. So that leads us on to what we would

call the dialogue state architecture, slightly more sophisticated approach

to what we saw with graphs earlier. And this introduces something

called dialogue acts, which we'll talk about in a moment and

also dialogue state architecture use machine learning,

in addition to rule based formalisms. So you can see this little graphic that

we have over here on the right which indicates kind of six stages in

the dialogue state architecture. You have, obviously the automatic

speech recognition, which then leads to a spoken language

understanding component which is what we talked about briefly

on the previous slide. And really what's interest here is

the three blocks at the bottom, which we're going to introduce, as part

of that extra sophistication, dialogue, state architecture. That the state tracker, the policy

in the natural language generation. And briefly the state tracker does exactly

that it keeps track of the most recent act and all the slots and

fillers of all the exchanges up to now. The dialogue policy is what defines what

the system should do or say next and when to answer users questions

versus when to clarify or suggest. And then natural language generation,

then creates the response that is given back to the user

perhaps based on templates. And also the context that's

coming out of the frames and then he filled slots so far. So I mentioned dialogue acts on the

previous slide, he can think of dialogue acts as specifying essentially

the function of a turn or an utterance. And I tend to think of these a little bit

like moves in a chess, game of chess where there are certain rules as to

when you can play a certain move. And then certain implications of

when you've made that move what the implications are for the state of the

game and dialogues play a similar role. A little bit like the speech acts

that we talked about earlier but also a notion of grounding the common

representation we talked about earlier. They're obviously very task specific. The task tag set for dialogue items for

booking a restaurant would be very different to

the tank set for setting an alarm. So that develops on a case by case basis. And then essentially the slot filling

takes place through a machine learning paradigm where we have semantic parsing,

supervised semantic parsing. Where we will train a parser that

can map from the input words to fill the slots to identify the domain and

the intent. So for example we might use a contextual

embedding model such as BERT and then take the representation

output from that. Put it through a feed forward layer and a softmax which would then give us

the BIO tags beginning important outside tags using the formulas

that we talked about earlier. And also we could combine

that with a domain and intent classification as the desired

output for the final and the sentence token to give us all

three elements that we that we need. In practice, often, these systems are bootstrapped using

simple handwritten GUS-style rules. So for example, we might start from

a hand annotated set of rules and a test data set. And then new utterances are labeled using

those rules to produce new tuples or training data. And then we can go ahead and

train our classifier on those tuples and continue to evaluate them in like a bit of

a regression approach on the test set and use that to build out our training set. So that leads us on to talking

about the dialogue state tracker, which we mentioned briefly earlier,

which essentially does exactly that. It keeps a track of where

we are in the dialogue. What the current state of the frame is,

what slots have been filled, what slots remaining to be filled. What the users most recent

dialogue act was and essentially this works through

a process of interpretation via. Again, machine learning paradigm

supervised classification, which generally is trained

using hand labelled data. And the dialogue policy, which you mentioned earlier is essentially

deciding what action to take next. And again,

that's using machine learning approaches. But typically based on the entire

dialogue state where we're looking for the speech act. That is the maximum probability

based on all the previous speech acts from the system a and

the user you shown here in this formalism. But we can simplify that by

considering only the current frame and the last exchange. So essentially the formalism boils

down to again finding the most likely speech out based

on the current frame and just the previous articulation

from the user and from the system. So finally we will look at natural

language generation and there's two parts. This is the content planning which

is determining what to say and the sentence realization which is

giving it the surface structure or instantiating it in other words

determining exactly how to say it. Now we talked about the dialogue policy

earlier which essentially does the most of the content planning element. So that is decides on what is

the appropriate dialogue act and also determine some of the attributes and

slots and fillers. So that leaves the sentence

realization part and that's trained on again

a representation and sentence pairs. So hand label data in a corpus,

of course training data like this, we're talking about very generative,

diverse dialogues. Training data is very hard to come by all

the different combinations of, restaurant, for example, restaurants, locations. Or cuisines or cities or flights or airlines in the case of cars very

hard to get that training data. So what we do is typically a process

of what's called delexicalization to generalize, so

we'll substitute instances for the class. So here we've got an example of

a particular restaurant a particular city and we'll delexicalized that to just

restaurant name in whatever location serve whatever cuisine. And then at runtime we use an encoder and decoder to map from the frames to generate

those delexicalized sentences and that's where the the input frame then

comes in with the slots and fillers. Combined with the content planner to

relexicalize to instantiate those categories for, in this case restaurant

location cuisine with the actual items that are there present in the in

the slots, the fillers for the slots. So there it is there's a brief overview of

task based dialogue systems very useful and all around us all the time. So it's quite revealing to have

some insight into how these things actually work.

# **Task-based dialogue systems practice quiz**

### Question 1

What do dialogue acts represent?

* Specific slot values
* A structured representation for user intents
* Attribute-value pairs in the frames
* The interactive function of a turn or utterance

### Question 2

Which of the following are components of the dialogue state architecture?

* Natural language generation
* Dialogue policy
* Delexicalisation
* Dialogue state tracker

# **Design and evaluation**

In this segment, we're going to talk about

some of the issues involved in designing and evaluating

dialogue systems. In contrast with some

other topics within NLP, designing dialogue systems is a very much an

interactive process. Essentially, you're building

something that's going to converse with human beings. In that respect, you're

building what would be called an

interactive product. In that context, it's

appropriate to draw on principles from

another discipline, that is human-computer

interaction, in order to guide and

shape that design process. Essentially, what that

means is we can adopt what's called the

user-centered design process. There's a brief summary of

what that involves here. You can see essentially

it means that we study uses and tasks that

we develop a deepened, very solid evidence-based

understanding of what the user

is trying to do, who they are, what

their capabilities are, and what they're trying to

do and in what context. Then we iteratively build

simulations and prototypes. I design solutions to meet those requirements

and then we test the design on users,

getting actual feedback. Now, note actually that designing dialogue

systems very often uses a technique in

[inaudible] which is actually quite rare in most

team [inaudible] contexts. But because designing

dialogue systems can be quite expensive or

traditionally was quite expensive, it's certainly a lot

more commoditized now. Once often happened in the past. As a solution to that, rather than build an expensive

system and then find it needs to radically

change and then build another expensive system, researchers often

use what's called Wizard of Oz techniques. Wizard of Oz techniques,

the idea is it borrows from the storybook where

the Wizard of Oz was essentially this

all powerful character that projected great abilities. But actually behind

the screens was just a very ordinary person with very ordinary capabilities. The idea is that we can research complex

dialogue systems without actually having to build them by putting

a human researcher in the loop that emulates

these complex behaviors and then elicits the feedback that

we need in order to guide the development of those

dialogue systems to meet those needs without actually having to build them in advance, which of course, can be very expensive

and time-consuming. That's essentially how Wizard

of Oz methods work and that simulates the interactions that we need in order to inform

the design process. We mentioned that it's a

highly iterative process. It turns out there's

actually a standard of human-centered

design process for interacting systems,

ISO 9241-210. You can see here

that essentially, we identify the need for the center design

specified context of use. You can read this graphic

in a clockwise manner. We specify the requirements by doing the user research that we talked about

on the previous slide. Produce design

solutions, prototypes, and then we evaluate

their designs where we test them

directly with users. There's all sorts of different techniques for

actually doing that. But note that the whole

process is iterative. The evaluation of design, very rarely does anybody get any design right the first time. The feedback is then used to refine and better specify

the context of use, which in turn leads to a better understanding

of the requirements and a refinement of the design

solution and so on, up to some point at which is

deemed as meeting whatever acceptance criteria

are appropriate for production or launch. There's other ways

of looking at this. User-centered design process. Here's Design Council's

Double Diamond way of framing of the

design process. You can see here the

stage is discover, define, develop, and deliver. A lot of people think

design starts with this point in the

middle, it doesn't. It starts over here on

the left where actually the first phase is what we

might call divergent thinking, trying to understand the problem more deeply through a process

of research or discovery. Then there's convergent thinking where we're trying to define the specific problem

that we're trying to solve and develop the

insights that we need. Then from the specific problem, then there's design

exploration or ideation where we're developing

alternative solutions. Finally, then we narrow

down the scope of those solutions to one

particular prototype or set of prototypes

which we then deliver. That's another way of thinking about the design process which is very applicable

to dialogue systems. That's a brief overview

of the design process. Let's turn our attention

to evaluation. A lot of evaluation in NLP is done using

automated techniques, certainly for

machine translation. You may have heard of metrics

such as BLEU or ROUGE. They're essentially based on patterns of co-occurrence and n-grams in the benchmark

data and in the test data. They often work as a

reasonably good proxy for human performance

or human evaluation. It turns out that with chatbots, because they're so

generative because chat is so diverse and human

conversation is so diverse that these methods really

are a good proxy, and that instead

evaluation has to be a more manual undertaking. Typically there are

two approaches; We have what's called participant evaluation and

we have observer evaluation. With participant

evaluation typically, the person taking

part in that dialogue is the one that makes

the evaluation. For example, we might

have six terms in the dialogue and then

a questionnaire is administered where the user is invited to school

that dialogue alone. Eight dimensions such as avoiding repetition,

interestingness, making sense,

fluency, listening, inquisitiveness, humanness,

and engagingness. Then there's

observer evaluation, which is where the people

or the individuals making that evaluation

aren't necessarily the ones involved

in that dialogue. You might have third-party

annotators evaluating a complete conversation and comparing that with

some benchmark systems. This is a kind of AB test or AB bake-off between two

different systems and comparing them, which is the better

one on various scales, typically engagingness, interestingness, humanness,

and knowledgeable. That's how to evaluate chatbots. What about task-based

dialogue systems? Well, task-based

dialogue systems in some ways is simpler, that adds up to help the user

complete a specific task. The simplest approach is

to measure a task success. Did they complete that task within what space and

time and in what effort? Or alternatively, because

it's not always possible to administer measure task success, another approach is to

measure user satisfaction, that's via a questionnaire

or to look at something a little bit more quantitative, such as looking at

the slot error rates. We could retrospectively

look at how the slots were filled after a

particular dialogue and mark them according

to some gold standard. Or we could work out

the slot precision or recall or F-measure. Same idea of looking at the extent to which the slots have been populated accurately. Just finally, we'll touch

on some ethical issues. Like a lot of things in

natural language processing, if you use machine

learning approaches, you've got to be aware of the fact that your

training data will almost inevitably contain

biases of some sort. The machine Learning systems

can not only replicate, but also amplify that bias. There's also gender stereotypes that are rife in

this training data. There are many examples in the literature of

researchers that have discovered and

shown the existence of negative gender stereotyping

and other types of stereotyping as well that become propagated through machine

learning approaches. Most of us are familiar

with the example of Microsoft's Tay chatbot

launched a couple of years ago. Again, trained on the various sources of

publicly available data, but picked up all sorts

of negative connotations or inappropriate attitudes and beliefs from that training data amplified and propagated those. Of course, if we have

dialogue agents in the home, they may be listening

for the wake of words. They're going to catch

or the fragments of dialogue that may be private

between those individuals. Of course, users may

accidentally disclose information as part of that speech recognition

process, passwords, and so on which certainly

should not be uploaded elsewhere and not be

used as training data. There it is. That's

a brief look at the design and

evaluation issues. In the next part of the topic, we'll start to turn our

attention to some of the practical

implications of how we might go about building

our own chatbots.

# **Design and evaluation practice quiz**

### Question 1

Which of the following are steps in a typical user-centred design process?

* Evaluate designs
* Specify context of use
* Produce design solutions
* Annotate training data

### Question 2

Which of the following are common approaches for chatbot evaluation?

* Task success evaluation
* Observer evaluation
* Automated evaluation
* Participant evaluation

# **Activity: Exploring speech acts**

Examine the following conversation, and label each turn with one of the following speech acts:

* Constatives: commit the speaker to some proposition, e.g. claim, confirm, deny
* Directives: get the addressee to do something, e.g. ask, forbid, invite
* Commissives: commit to some future course of action, e.g. promise, plan, vow
* Acknowledgements: express reaction to some action, e.g. apologize, greet, thank

C1: . . . I need to travel in May.

A2: And, what day in May did you want to travel?

C3: OK uh I need to be there for a meeting that’s from the 12th to the 15th.

A4: And you’re flying into what city?

C5: Seattle.

A6: And what time would you like to leave Pittsburgh?

C7: Uh hmm I don’t think there’s many options for non-stop.

A8: Right. There’s three non-stops today.

C9: What are they?

A10: The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.

C11: OK I’ll take the 5ish flight on the night before on the 11th.

A12: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.

C13: OK.

A14: And you said returning on May 15th?

C15: Uh, yeah, at the end of the day.

A16: OK. There’s #two non-stops . . . #

C17: #Act. . . actually #, what day of the week is the 15th?

A18: It’s a Friday.

C19: Uh hmm. I would consider staying there an extra day til Sunday.

A20: OK. . . OK. On Sunday I have . . .

# **Understanding speech acts**

In the previous exercise you examined a conversation and labelled each turn with speech acts. Now consider the following questions: How easy was the task? Which speech acts are more frequent? Were there any ambiguous instances? Are there any labels where you disagree with others? How language-specific are they? What other speech acts could you imagine that aren’t accounted for by this set? Once you’ve posted your comments in the forum, take a look at those of other learners and comment on the differences.

# **Basic chatbot using nltk**

In this segment, we're going to put

into practice some of the things we talked about in the first half of this topic. As opposed to looking

at the theory and the concepts principles in

the architectural chapels, we're actually going

to have a little bit of a hands-on here. In our Jupyter Notebook, we're going to experiment

with a chatbot and see how it's put together. Here we are in our

Jupyter Notebook, and we're going to look at a couple of chatbots

actually built using NLTK. This is fairly simple stuff. As you'll see, it's

basically based around the idea of

rule-based approaches. A little bit similar

to what we studied in the first half of this topic. Let's have a look how

these things work. To get us started, we

just simply import NLTK. Then we import

something called chat, which is the module for

creating the chatbots, and also something

called reflections, which will have a closer

look at in a moment. Now, the way NLTK works, it's actually similar to various other Java

architectures, but it revolves around

this idea of pairs, where essentially

it's the rules. Each pair has a

regular expression that matches the user input, and then a set of

possible responses in another list which

are chosen at random. You can see here, the pairs is just a big list and then we've got

another list within it, and each list has a

pattern and a response. Notice also that you can refer to entities

that the user enters. Here, for example, we've got I like dot star in brackets, and that's the element

captured in the brackets, is what we call a

captured group. If you have multiple

capture groups, you can refer to them by

their index position. Here we're saying, "Why

do you like percent one?" and that refers to the first

captured group in the input. If somebody says, "I like football," we can respond by

"Why do you like football?" It's increasing the coherence of the conversation by

mentioning something that your counterparty has

already mentioned to you. You can see here,

just very simple. We've just got half a dozen, maybe seven different pairs. Not, that sophisticated, but we can cover basic

conversation with that. We instantiate a chatbot using the pairs and

the reflections. I'm going to show you the

reflections just a moment. We can coli converse method, which comes up a little

input field and we can say, "Hello, how are you?" We can say, "I'm fine. How about you?"

That didn't work. Well, what we actually we'd say? I think what it's expecting

is this, "And you." There we go, "Doing

well, thank you." We can say, "I like pizza." There you go. That's an

example of the captured group. It's recognized the

string after "I like", and it's repeating

that back to me. We could say, "It's tasty." We get a response like that. When we've had enough, what if we just put

something random, it just says, "I'm sorry,

I don't understand." What you can see is,

there's a catchall. These pairs are ordered and

the catchall is the last one. So if none of these

other things match, then it basically just

says, "I'm sorry, I don't understand" in

a very simple chatbot. As when we're done, we can say, "Quit" and that is that. Obviously, very simple, but you can get up and

running and create your own chatbot in minutes, basically, not hours, minutes. Reflections is the part where

we do the transformations. You don't have to use this. But you can override it. There's a default set, but

you can override if you want, or not use it as you see fit. This is basically the

bit, remember analyzer, where it was able to echo

back to the user and transform their inputs from the first-person to the

second person singular. "I am" becomes "you are", "I was" becomes "you were", and so on all the way

down, "me" becomes "you", so that when you record

something back to somebody, obviously you're talking

in the second person, whereas when they were talking, they're talking

about themselves, so that would be

in first-person. That's another way of making the dialogue more coherent and more believable.

There we have it. That's the simple

NLTK chat class. We can instantiate analyzer

chat as we saw earlier, we could say, I don't

know. "Very interesting." You can chat away. I don't know, let's

just try something that might trigger

one of these words. There you see now there is a reflection happening

because that, "I'm unhappy", and it says, "Did you come

to me because," and it's transformed the "I am"

to "you are unhappy." But if I say something

meaningless, it'll probably just

come up. There you go. Now there's another

interesting one. It's trying to keep the

conversation going. I'm going to really fool it anyway. That's

in it for that one. That's Eliza, there's also

I Eisha which I think is some sorts of teen chatbots. We can give that a

whirl, "I am cool." Let's see what it

says. "No I am not." I'll say yes. Well, you get the

idea. That's that one. Then I'll just try one more. Let's try the route chats. "If you must" [inaudible]. "Let's talk about footy." "Help". Yeah. You get the idea. I think I just typed

that in the wrong place. There you go.

That's another one. It doesn't want to let me quit. No, maybe it has let me quit. There we go. There's two others. Suntsu chat, which I think

he's some Chinese martial, deep insights into

warfare, and the zen chat, which I think is all

about enlightenment and I'm that kind of thing. Feel free to have a play. You can obviously

use one of these. They're all open source,

freely available. You can use one of these as the base for you to build

something yourself. There it is. Chatbots in NLTK, hours of entertainment

doing itself, but probably more productive to take one of these as baseline to build something of

your own choosing.

# **Installing Rasa**

In the previous segment, we looked at NLTK and we saw how we can

attribute to know, but we could build

a basic chatbot using keyword

matching techniques. That was a platform that we could build on to

extend our own, create our own chatbot. We're going to have a look at another chatbot framework here. We're going to look at

something called Rasa, which is a more

recent development. Probably the most

sophisticated platform for building chatbots to utilize this deep learning and various other more contemporary

approaches to NLP. It's the thing where

you probably want to do outside of a Jupiter Notebook. We can see here, basically

a Windows command prompt. I will share with you that it was a little bit tricky

to install on Windows, but I did succeed in the end. I've also installed

it on a Mac and that was a slightly less,

more seamless process. Not entirely straightforward,

but it's worth pursuing. Certainly, when you

got Rasa installer, it's a very flexible

and powerful package. Seeing here in this window, recently I use pip install

to install Rasa within a virtual environment and

what we're going to do now is we're going

to initialize Rasa. That will do some

interesting things. I haven't actually

done this yet. What you're seeing here is live, I've tried this on the Mac. So I know what to

expect on a Mac, but I've not done it on

the Windows machine, let's see what happens

when we try to initialize Rasa on Windows. I don't think the

answer is no a lot. Let's give it a little

bit more patience and see what happens. We're getting started. If you install this yourself, you'll see a transcript

similar to this. You can get started quickly. Initial project will be created. I've already created a

directory, put all this in, so we will allow it to install

in the current directory, and we will train

an initial model. You'll see various bits of output flushing of the

screen as this continues. It's probably going to

run the somewhat more slowly on this

machine and on my Mac it's got half as much memory and the process probably

not as powerful either. Your mileage will

vary depending on the specification of

your own machine. If you want to try this

as a local install, you might find you

got experienced similar to what

we're seeing here, which is not a lot happening, but if I run it on

a MacBook Pro with 16 gigabytes memory and think it was the

default processor. What we're about to see now took of the order

of about a minute, maybe two at the most. It's sending the thing that you can do live solely on the Mac. As I said, I'm on a

Windows machine now, which is a much

lower specification, which I think is why this is

taking a little bit of time. Here it's moved onto

training the model. You see lots of activity there and in a moment we'll

have a little look under the hood that the various configuration

files that specify how the bot is configured and how the

training process works. We'll look at that in a moment or possibly in the next video. See here it's now

very onsite training and going through

various epochs. The training is completed. I think that's fair to

say that that was so much slower than on Mac machine, but mileage may vary. In a moment when it's

completed this training, it should ask us if we want

to interact with the bot. It looks as if the core model

training has completed. Some point we should

segway into the shell, which allows us to

interact with the bot. Here we go, join speeds to train the system on

the command line. Who would say no to a

question like that. It will load the model that

it's just trained and this might take a moment because I'd imagine that model

is quite vague. Again, just to reiterate, I run this on a 16 gigabyte Mac and everything that

you've seen up to now, the whole process of

the order of minutes, two minutes at the most, really not that long, but obviously this machine

is much lower specification. Here we go. You can't see, but I've got a message

popped up whether the Windows Defender firewall

blocks some features, but I'm going to allow access. Hopefully, that will

solve that problem. It's waiting for some input now. So what I can do is I can type hello and hopefully it will come back with something

intelligent. Here we go. Say good, and you? [inaudible] number

shows expecting. [inaudible] here's

something to cheer you up. I am not sad. Let's see what it

thinks of that. Same response. Let's say where do you live? It definitely doesn't really

want to talk to me today. So let's just stop that. But you get you'll

see in a moment, we'll look under the hood

of this and we'll look at the various files but for now we'll just stop the

bot at this point. I think and that put

shuts down the server. If you ever want to

run the bots again, all you need to do is type rasa shell and it doesn't need to go through a

lot training process again, it will just fire up the

[inaudible] could try that now just to

show you rasa shell, and you'll see the

same bot is loaded. In the next couple of videos, we'll look at the various

configuration files that govern the behavior of the bot and how to refine and extend them. But I think that's it for now. That's how to get Rasa up and running on your own machine.

# **Configuring Rasa**

So in the previous video we had a quick

look at how to install razr and in this video we're going to have a little

look at how it's structured and try and understand the roles of the various

configuration files that razr relies on. So here we are in our Jupyter notebook and you can see there's five particular files

that are worth having a quick look at. First one is what's called

a config dot yaml file and this basically specifies how your

agent is going to be trained. So you can see here that there's

the model information and there's also some policy

information as well. Both of these are commented out and

we're not going to cover them, the content in too much detail today. But these are the defaults that

you would get out of the box and that you can use to create your

own specific configuration. So the next file to look at is domain

dot yaml and this is the one that lists all the possible intent that

your bot might want to represent. And you can see here there's the seven

ones that we encountered when we looked at the previous video and

we actually tried to bottom out. So those are the intent and obviously the

intent, we discussed these in the first half of this topic from

a conceptual point of view but you can see how they're

instantiated here within razr. And you can also remember when

we looked at the NLTK bot, how we sort of had the idea of intents,

we had pairs where you had certain types of input

mapping onto certain types of output. What was essentially a one to one

relationship with reg x matching and then some sort of output. There's a similar sort of thing in razr

but it's a much more flexible approach. So we have the intents

which are the inputs and then we have the responses as well. So you can see there's

a variety of responses here, half a dozen of those and you'll see

how these are connected up in a moment. But just if I should say that the main

yaml file, you've got the intents and the responses now how those

intents recognized that's the role of the nlu dot yaml files. So you've got here,

you've got the intent greet and you've got all the examples

that it looks for. So this is kind of the input rules

that specify how the matching is done on the input. So you can see intent goodbye and you got

the examples in terms of deny and so on. So those are the intents. And then we've got what

are called rules and these are the closest analogue

I guess to the pares in an LTK. These are the hard coded ones that

says for this input do this output. So for example here say goodbye

anytime user says goodbye. So you've got the intent goodbye and

the action, utter goodbye and that goes back to domain dot yaml file

where we've got the intent goodbye. And then the action was to utter

goodbye which is down here so that gives the text output. So that's really the equivalent

of the pares in LTK. But rather is so much more sophisticated

in a number of ways it removes that sort of direct one to one mapping with

a more indirect many too many mapping of lots of inputs could produce

lots of alternative outputs. But also it has the notion of rather

than just relying on whatever the last input was which is sort of

what the rules dot yaml specifies. It has the concept of [INAUDIBLE] or

coherent conversations or paths through that conversation which are

encapsulated in the stories dot yaml file. And here we can see we've got

three alternative stories, conventional to articulate in the first

story, what we call a happy path. Which is the sort of clear and unambiguous

route from a start of conversation to an end of conversation, achieving some

sort of goal in the simplest way possible. That's what we call the happy paths,

that one normally comes first. But first but you've also got various

other alternatives which indicate how the dialogue could flow. So that's obviously a much more

sophisticated approach than the simple input output pattern matching pairs. Which don't have much history other than

the memory function which we talked about that was built into Eliza. So that's just a quick introduction

to the key files within razr, and these are the ones that

you'll be modifying and editing if you want to build

your own chat, bot using razr.

# **Basic chatbot using Rasa**

In the previous video, we had a look at how to

install and configure Rasa. In this video, we're going

to have a brief look at how to use it to build or rather to extend the default

chatbot that you get and how to just do some very elementary

things using the rules. Now, I should point out actually that, in the previous video, obviously, we looked at those

five configuration files. I'll just draw your attention

to a couple of things. The config.yml file

is interesting. We're not going to

change it today, but I'll just draw

your attention to the fact that up to now with the work we've done with NLTK has been really rule-based. In other words, for

a specific input, then the chatbot comes

with a specific output, which is fine for

certain use cases, but, of course, it

doesn't generalize. Users can come up with all sorts of creative and

unpredictable inputs, and rule-based systems

can be very brittle. That's where the power of

all these other policies come in with Rasa. It has obviously the

ability to handle rules, but also it has other policies. Memoization refers to how

the stories are interpreted. If you recall, we looked

at the stories earlier, which provided a happy path

through the dialogue i.e the way you would

expect users to behave with your system ideally, and then various

other paths through the system that

you've considered in advance and ways of

managing that dialogue. These are much more flexible. They are parts of Rasa's

deep learning frameworks, so they can generalize to unpredicted inputs in

ways that rules can't. Further to that, there's

also the TED policy, transformer embedding

dialogue policy, which takes that concept of generalization further

still with the ability to the next action prediction and entity recognition

at the same time. In other words, it can generalize

much more gracefully to unexpected and

unseen user inputs. But we're just going to

focus on the simplest of those use cases at the

moment, just the rules. If you recall, we looked at

the domain.yml file earlier, and it had certain intents and responses already in there. We've just added one, this one for hours, which we'll define

in just a moment. We've also added a

response, utter hours. You'll see how

those are connected through the nlu.yml files. It recognizes the intent hours by these examples

that you see here. Also, you can have

regexes as well to articulate how an hour's

intent might be expressed. Then, as I mentioned, we can add a rule

which basically says, for this pattern,

do this action. For an intent hours,

we utter hours. If we recognize the

intent that we have here, then we utter hours, which gives this text here. That basically is

how we can extend the rules-based framework to add functionality

to our chatbot. Here we've got it running. I've already trained it, and I've run Rasa shell

to get it running. If we now type, "What are your hours?" Hopefully, we will get a sensible response

that we just saw. There you go, straight out of the rule that we put

in and the intent. If I just put hours,

maybe that should hopefully match the same in

turn. Yeah, there you go. That's much from the regex. We could do all those sorts of other things that we did before. It tries to cheer us up. We could say I'm great

or something like that. I will stop. We'll

see if we can exit. There we are. Let me

just stop it now. There we are. That's

the Rasa chatbot. As I said, all we've done really is just to extend the rules. I would encourage you to

also explore stories as a much more flexible way of

managing a coherent dialogue. Of course, the other policies, the TED policy and so on, to deal with a much more sophisticated

conversational behavior.

# **Basic chatbots quiz**

### Question 1

How are dialogue rules represented in nltk?

* As stories
* As reflections
* As pairs
* As policies

### Question 2

How are dialogue rules represented in Rasa?

* As policies
* As stories
* In the rules.yml file
* In the config.yml file

### Question 3

Why should dialogue rules not be over used in Rasa?

* They take a long time to train
* They are best suited to small specific conversation patterns
* They don’t generalise to novel inputs
* They rely on stories to make sense

# **Activity: Building your own chatbot**

In this exercise you will build your own chatbot. If you don’t want to install and configure Rasa, use the nltk example as your starting point. Choose a simple task for your assistant to do, like ordering a pizza or booking an appointment. Or you could simply make your chatbot knowledgeable about your favourite subject. If you are using the nltk example, you may wish to:

* Create additional input/output pairs
* Capture input variables using parentheses and field indexes (e.g. %1)
* Order the rules to support precedence and diversity
* Implement a simple memory stack as described in the lecture

# **Understanding chatbot platforms**

In the previous exercise you built your own chatbot. Now consider the following questions: How easy was it to get started? What limitations did you encounter? Which platform offers the most sophistication, and why? Which one would you recommend for building task-based dialogue systems, and why? Once you’ve posted your comments in the forum, take a look at those of other learners and comment on the differences.

# **Dialogue systems summary**

So here we are, then we've finished our journey

through the world of dialogue systems. Let's just briefly summarize

the things that we've learned. So, we said our learning objectives this

topic were to understand the properties of human conversation, and

that's how we started. We spent some time analyzing

dialogue system architectures, looking at different

types of dialogue system. And we spent some time in

the second half of this topic, creating some simple chatbots. So if you recall right at the start,

we talked about human conversation, we talked about that it was

a sequence of turns and there were rules governing

that human conversation. And we talked about speech acts. We talked about grounding, how we might establish common

ground between two participants. And we talked about dialogues

being mixed initiative as well. And all these things are properties of

human conversation that we need to know about in order to design

effective automated conversations. And then we talked about how, essentially, there are two types of dialogue systems

chatbots and task-oriented systems. And the chatbots are more

about extending conversations, where the chat is essentially

the whole purpose and a lot of these are used precisely

because of that entertainment value. And we looked at all the way

back to our lives on one of the very first chatbots and

we saw how that was rule-based and that more contemporary types of

chatbot used corpus-based approaches. Then we turned our attention to

task-based dialogue systems. We talked about GUS, again a very

earliest but influential example of that. We talked about frames as

a representation of user intents, and slots as a way of capturing

the attribute values for the entities and the things that

were mentioned in that exchange. And we talked about how, essentially, the process was to ask questions to fill

those slots and perform random actions. And then we talked also about

designing dialogue systems. And we saw, essentially because dialogue

systems are so interactive, we should borrow principles from a neighboring

discipline of human computer interaction. And we talked about user centered-design

processes, studying users and tasks, building simulations,

and iterative testing. So that's it. We finally talked about evaluation. We talked about the fact that it has to

be done manually, then we talked about the two approaches of participant

evaluation, observer evaluation. And we talked about how for chatbots,

we talked about have dialogues, task-based dialogue systems we could

measure things like task success, or user satisfaction,

or the slot error rate. So that brings our little journey through

dialogue systems to a conclusion. It's a very rapidly expanding and important subtopic within

natural language processing. Lots of interesting research were going

on, so I hope you found it useful.