[MUSIC PLAYING]

BRIAN YU: Welcome back, everybody, to our final class in an Introduction

to Artificial Intelligence with Python.

Now, so far in this class, we've been taking problems

that we want to solve intelligently and framing them in ways that computers

are going to be able to make sense of.

We've been taking problems and framing them as search problems or constraint

satisfaction problems or optimization problems, for example.

In essence, we have been trying to communicate about problems in ways

that our computer is going to be able to understand.

Today, the goal is going to be to get computers

to understand the way you and I communicate naturally,

via our own natural languages.

Languages like English.

But natural language contains a lot of nuance and complexity

that's going to make it challenging for computers to be able to understand.

So we'll need to explore some new tools and some new techniques

to allow computers to make sense of natural language.

So what is it exactly that we're trying to get computers to do?

Well, they all fall under this general heading of natural language processing,

getting computers to work with natural language.

And these tasks include tasks like automatic summarization.

Given a long text, can we train the computer

to be able to come up with a shorter representation of it?

Information extraction.

Getting the computer to pull out relevant facts or details out

of some text.

Machine translation, like Google translate,

translating some text from one language into another language.

Question answering.

If you've ever asked a question to your phone

or had a conversation with an AI chatbot where

you provide some text to the computer, the computer

is able to understand that text and then generate some text in response.

Text classification, where we provide some text

to the computer and the computer assigns it a label, positive or negative,

inbox or spam, for example.

And there are several other kinds of tasks

that all fall under this heading of natural language processing.

But before we take a look at how the computer might

try to solve these kinds of tasks, it might be useful

for us to think about language in general.

What are the kinds of challenges that we might

need to deal with as we start to think about language

and getting a computer to be able to understand it?

So one part of language that we'll need to consider is the syntax of language.

Syntax is all about the structure of language.

Language is composed of individual words,

and those words are composed together in some kind of structured whole.

And if our computer is going to be able to understand language,

it's going to need to understand something about that structure.

So let's take a couple of examples.

Here, for instance, is a sentence.

"Just before nine o'clock Sherlock Holmes stepped briskly into the room."

That sentence is made up of words, and those words together

form a structured whole.

This is syntactically valid as a sentence.

But we could take some of those same words,

rearrange them, and come up with a sentence that

is not syntactically valid.

Here, for example, "Just before Sherlock Holmes

nine o'clock stepped briskly the room" is still composed of valid words,

but they're not in any kind of logical whole.

This is not a syntactically well-formed sentence.

Another interesting challenge, is that some sentences will have

multiple possible valid structures.

Here's a sentence, for example.

"I saw the man on the mountain with a telescope."

And here, this is a valid sentence, but it actually

has two different possible structures that

lend themselves to two different interpretations and two

different meanings.

Maybe I, the one doing the seeing and the one

with the telescope, or maybe the man on the mountain

is the one with the telescope.

And so natural language is ambiguous.

Sometimes the same sentence can be interpreted in multiple ways.

And that's something that we'll need to think about, as well.

And this lends itself to another problem within language

that we'll need to think about, which is semantics.

While syntax is all about the structure of language,

semantics is about the meaning of language.

It's not enough for a computer just to know

that a sentence is well-structured if it doesn't know what that sentence means.

And so semantics is going to concern itself with the meaning of words

and the meaning of sentences.

So if we go back to that same sentence as before, "Just before nine o'clock

Sherlock Holmes stepped briskly into the room."

I could come up with another sentence.

Say the sentence, "A few minutes before nine,

Sherlock Holmes walked quickly into the room."

And those are two different sentences, with some of the words the same

and some of the words different, but the two sentences

have essentially the same meaning.

And so ideally, whatever model we build, we'll

be able to understand that these two sentences

while different, mean something very similar.

Some syntactically well-formed sentences don't mean anything at all.

A famous example from linguist, Noam Chomsky, is the sentence,

"Colorless green ideas sleep furiously."

This is a syntactically, structurally well-formed sentence.

We've got adjectives modifying a noun, ideas, we've got a verb and an adverb

in the correct positions.

But when taken as a whole, the sentence doesn't really mean anything.

And so if our computers are going to be able to work with natural language

and perform tasks in natural language processing,

these are some concerns we'll need to think about.

We'll need to be thinking about syntax and we'll

need to be thinking about semantics.

So, how could we go about trying to teach a computer how to understand

the structure of natural language?

Well, one approach we might take is by starting by thinking

about the rules of natural language.

Our natural languages have rules.

In English, for example, nouns tend to come before verbs.

Nouns can be modified by adjectives, for example.

And so if only we could formalize those rules,

then we could give those rules to a computer,

and the computer would be able to make sense of them and understand them.

And so, let's try to do exactly that.

We're going to try to define a formal grammar, where

a formal grammar is some system of rules for generating sentences in a language.

This is going to be a rule-based approach to natural language

processing.

We're going to give the computer some rules that we know about language,

and have the computer use those rules to make

sense of the structure of language.

And there are a number of different types of formal grammars,

each one of them has slightly different use cases.

But today, we're going to focus specifically

on one kind of grammar known as a context-free grammar.

So how does the context-free grammar work?

Well, here is a sentence that we might want a computer to generate.

She saw the city.

And we're going to call each of these words a terminal symbol.

A terminal symbol, because once our computer has generated the word,

there's nothing else for it to generate.

Once it's generated the sentence, the computer is done.

We're going to associate each of these terminal symbols

with a nonterminal symbol that generates it.

So here we've got N, which stands for noun, like she or city.

We've got V as a nonterminal symbol, which stands for a verb.

And then we have D, which stands for determiner.

A determiner is a word like the or a or an in English, for example.

So each of these nonterminal symbols can generate the terminal symbols

that we ultimately care about generating.

But how do we know, or how does the computer

know which nonterminal symbols are associated with which terminal symbols?

Well, to do that, we need some kind of rule.

Here are some what we call rewriting rules, that

have a nonterminal symbol on the left-hand side of an arrow,

and on the right side is what that nonterminal symbol

can be replaced with.

So here, we're saying the nonterminal symbol N, again, which stands for noun,

could be replaced by any of these options separated by vertical bars.

N could be replaced by she or city or car or Harry.

D for determiner, could be replaced by the, a, or an, and so forth.

Each of these nonterminal symbols could be replaced by any of these words.

We can also have nonterminal symbols that are

replaced by other nonterminal symbols.

Here's an interesting rule.

NP arrow N bar D N. So what does that mean?

Well, NP stands for a noun phrase.

Sometimes when we have a noun phrase in a sentence,

it's not just a single word, it could be multiple words.

And so here, we're saying a noun phrase could be just a noun,

or it could be a determiner followed by a noun.

So we might have a noun phrase that's just a noun, like she.

That's a noun phrase.

Or we could have a noun phrase that's multiple words, something

like the city.

Also acts as a noun phrase, but in this case,

it's composed of two words, a determiner, the, and a noun, city.

We could do the same for verb phrases.

A verb phrase, or VP, might be just a verb,

or it might be a verb followed by a noun phrase.