



Natural Language Processing: Crash Course Computer Science #36

Crash Course: Computer Science

<https://youtube.com/watch?v=fOvTtapxa9c>

<https://nerdfighteria.info/v/fOvTtapxa9c>

Hi, I'm Carrie Anne and welcome to Crash Course Computer Science.

Last episode, we talked about computer vision, giving computers the ability to see and understand visual information. Today we're going to talk about how to give computers the ability to understand language.

You might argue they've always had this capability. Back in episodes 9 and 12, we talked about machine language instructions as well as higher-level programming languages. While these certainly meet the definition of a language, they also tend to have small vocabularies and follow highly structured conventions.

Code will only compile and run if it's 100% free of spelling and syntactic errors. Of course, this is quite different from human languages, what are called natural languages; containing large, diverse vocabularies, words with several different meanings, speakers with different accents, and all sorts of interesting word play.

People also make linguistic faux pas when writing and speaking, like slurring words together, leaving out key details so things are ambiguous, and mispronouncing things.

But for the most part, humans can roll right through these challenges. The skillful use of language is a major part of what makes us human. And for this reason, the desire for computers to understand and speak our language has been around since they were first conceived.

This led to the creation of natural language processing or NLP, an interdisciplinary field combining computer science and linguistics.

===== Intro (1:16-1:23) =====

There's an essentially infinite number of ways to arrange words in a sentence. We can't give computers a dictionary of all possible sentences to help them understand what humans are blabbing on about. So an early and fundamental NLP problem was deconstructing sentences into bite-size pieces which could be more easily processed.

In school, you learned about nine fundamental types of English words: nouns, pronouns, articles, verbs, adjectives, adverbs, prepositions, conjunctions, and interjections.

These are all called parts of speech. There are all sorts of subcategories too, like singular versus plural nouns and superlative versus comparative adverbs. But we're not going to get into that.

Knowing a word's type is definitely useful, but unfortunately, there are a lot of words that have multiple meanings. Like rose and leaves, which can be used as nouns or verbs.

Additional dictionary alone isn't enough to resolve this ambiguity. So computers also need to know some grammar. For this, phrase structure rules were developed, which encapsulate the grammar of a language.

For example, in English there's a rule that says a sentence can be comprised of a noun phrase followed by a verb phrase. Noun phrases can be an article like "the" followed by a noun. Or they can be an adjective followed by a noun.

And you can make rules like this for an entire language. Then, using these rules, it's fairly easy to construct what's called a Parse Tree, which not only tags every word with a likely part of speech,

but also reveals how the sentence is constructed.

We now know, for example, that the noun focus of this sentence is "the Mongols" and we know it's about them doing the action of rising from something. In this case, leaves. The smaller chunks of data allow computers to more easily access, process, and respond to information.

Equivalent processes are happening every time you do a voice search. Like, "Where's the nearest pizza?" The computer can recognize this is a "where" question, knows that you want the noun "pizza", and the dimension you care about is "nearest".

The same process applies to "What is the biggest giraffe?" or "Who sang Thriller?" By treating language almost like Lego, computers can be quite adept at natural language tasks.

They can answer questions and also process commands like "set an alarm for 2:20" or "play T-Swizzle on Spotify". But as you've probably experienced, they fail when you start getting too fancy. And they can no longer parse the sentence correctly or capture your intent.

Hey Siri. Methinks the Mongols doth roam too much. What think ye on this most gentle midsummer's day?

Siri: I am not sure I got that...

I should also note that phrase structure rules and similar methods that codify language can be used by computers to generate natural language text.

This works particularly well when data is stored in a web of semantic information, where entities are linked to one another in meaningful relationships, providing all the ingredients you need to craft informational sentences.

"Thriller was released in 1983 and sung by Michael Jackson."

Google's version of this is called Knowledge Graph. At the end of 2016, it contained roughly 70 billion facts about, and relations between, different entities.

These two processes, parsing and generating text, are fundamental components of natural language chatbots - computer programs that chat with you.

Early chatbots were primarily rule-based, where experts would encode hundreds of rules mapping what a user might say to how a program should reply. Obviously, this was unwieldy to maintain and limited the possible sophistication.

A famous early example was Eliza, created in the mid 1960s at MIT. This was a chatbot that took on the role of a therapist and used basic syntactic rules to identify content in written exchanges, which it would turn around and ask the user about.

Sometimes it felt very much like human-human communication, but other times would make simple and even comical mistakes.

Chatbots and more advanced dialog systems have come a long way in the last 50 years and can be quite convincing today. Modern approaches are based on machine learning, where gigabytes of real human to human chats are used to train chatbots.

Today, the technology is finding use in customer service applications, where there's already heaps of conversations to learn from.



Natural Language Processing: Crash Course Computer Science #36

Crash Course: Computer Science

<https://youtube.com/watch?v=fOvTtapxa9c>

<https://nerdfighteria.info/v/fOvTtapxa9c>

People have also been getting chatbots to talk with one another, and in a Facebook experiment, chatbots even started to evolve their own language. This experiment got a bunch of scary-sounding press, but it was just the computers crafting a simplified protocol to negotiate with one another.

It wasn't evil, it was efficient.

But what about if something is spoken? How does a computer get words from the sound? That's the domain of speech recognition, which has been the focus of research for many decades.

Bell Labs debuted the first speech recognition system in 1952, nicknamed AUDREY, the automatic digit-recognizer. It could recognize all 10 numerical digits if you said them slowly enough. Five... Nine... Seven...?

The project didn't go anywhere because it was much faster to enter telephone Numbers with a finger. Ten years later, at the 1962 World's Fair, IBM demonstrated a shoebox-sized machine capable of recognizing 16 words.

To boost research in the area, DARPA kicked off an ambitious 5-year funding initiative in 1971, which led to the development of HARPY at Carnegie-Mellon University. HARPY was the first system to recognize over 1000 words.

But on computers of the era, transcription was often 10 or more times slower than the rate of natural speech. Fortunately, thanks to huge advancements in computing performance in the 80s and 90s, continuous real-time speech recognition became practical.

There was simultaneous innovation in the algorithms for processing natural language, moving from hand-crafted rules, to machine learning techniques that could learn automatically from existing datasets of human language.

Today, the speech recognition systems with the best accuracy are using deep neural networks, which we touched on in episode 34. To get a sense of how these techniques work, let's look at some speech. Specifically, the acoustic signal.

Let's start by looking at vowel sounds like "aaaa" and "eeee". These are the waveforms of those two sounds, as captured by a computer's microphone.

As we discussed in episode 21 on files and file formats, this signal is the magnitude of displacement of a diaphragm inside of a microphone as sound waves cause it to oscillate. In this view of sound data, the horizontal axis is time and the vertical axis is the magnitude of displacement or amplitude.

Although we can see there are differences between the waveforms, it's not super obvious what you would point to and say, "Aha! This is definitely an 'eeee' sound!"

To really make this pop out, we need to view the data in a totally different way: a spectrogram! In this view of the data, we still have time along the horizontal axis.

But now instead of amplitude on the vertical axis, we plot the amplitude of the different frequencies that make up each sound. The brighter the color, the louder that frequency component.

This frequency conversion from waveform to frequencies is done with a very cool algorithm called a fast Fourier transform. If you've ever stared at a stereo system's EQ visualizer, it's pretty much the same thing. A spectrogram is plotting that information over time.

You might have noticed that the signals have a sort of ribbed pattern to them. That's all the resonances of my vocal tract. To make different sounds, I squeeze my vocal chords, mouth, and tongue into different shapes, which amplifies or dampens different resonances.

We can see this in the signal, with areas that are brighter and areas that are darker. If we work our way out from the bottom, labeling where we see peaks in the spectrum, what are called formants, we can see the two sounds have quite different arrangements.

And this is true for all vowel sounds. It's exactly this type of information that lets computers recognize spoken vowels. And indeed, whole words.

Let's see a more complicated example like when I say, "She was happy." We can see our "eeee" sound here and "aaaa" sound here. We can also see a bunch of other distinctive sounds like the "she" sound in she, the "w" "aaa" "s" sound in was and so on.

These sound pieces that make up words are called phonemes. Speech recognition software knows what all these phonemes look like. In English, there are roughly 44, so it mostly boils down to fancy pattern matching.

Then you have to separate words from one another, figure out when sentences begin and end and ultimately you end up with speech converted into text, allowing for techniques like we discussed at the beginning of the episode.

Because people say words in slightly different ways due to things like accents and mispronunciations, transcription accuracy is greatly improved when combined with a language model, which can take statistics about sequences of words.

For example, "she was" is most likely to be followed by an adjective like "happy". It's uncommon for "she was" to be followed immediately by a noun. So if the speech recognizer was unsure between "happy" and "harpy", it would pick happy, since the language model would report that as a more likely choice.

Finally, we need to talk about speech synthesis. That is giving computers the ability to output speech. This is very much like speech recognition, but in reverse.

We can take a sentence of text and break it down into its phonetic components and then play those sounds back to back out of the computer's speaker.

You can hear this changing of phonemes very clearly with older speech synthesis technologies like this 1937 hand-operated machine from Bell Labs.

"Say she saw me. With no expression. She saw me... And I said an answer to these questions. Who saw you? She saw me... Who did she see? She saw me... Did she see you or hear you? She saw me..."

By the 1980s, this had improved a lot. But that discontinuous and awkward blending of phonemes still created that signature robotic sound.

"Thriller was released in 1983 and sang by Michael Jackson."

Today, synthesized computer voices like Siri, Cortana, and Alexa have gotten much better. But they're still not quite human. But we're so, so close and it's likely to be a solved problem pretty soon.

Especially because we're now seeing an explosion of voice user



Natural Language Processing: Crash Course Computer Science #36

Crash Course: Computer Science

<https://youtube.com/watch?v=fOvTtapxa9c>

<https://nerdfighteria.info/v/fOvTtapxa9c>

interfaces on our phones, in our cars and homes, and maybe soon, plugged right into our ears.

This ubiquity is creating a positive feedback loop with people using voice interaction more often. Which, in turn, is giving companies like Google, Amazon, and Microsoft more data to train their systems on. Which is enabling to better accuracy, which is leading people to use voice more, which is enabling even better accuracy, and the loop continues.

Many predict that speech technologies will become as common a form of interaction as screens, keyboards, track pads, and other physical input/output devices that we use today. That's particularly good news for robots, that don't want to walk around with keyboards in order to communicate with humans.

But we'll talk more about them next week. See you then!

===== Credits (11:17) =====

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows like Eons, PhysicsGirl, and It's Okay to Be Smart.

This episode was filmed at the Chad and Stacey Emigholtz studio in Indianapolis. And it was made with the help of all these nice people and our wonderful graphics team is Thought Cafe.

Thanks for the random access memories! I will see you next time.