Now stop me if you've heard this one before,
but there are a lot of large language models available today,
and they have their own capabilities and specialties.
What if I prefer to use one LLM to interpret some user queries in my business application,
but a whole other LLM to author a response to those queries?
Well, that scenario is exactly what the LangChain caters to.
Langchain as an open source orchestration framework
for the development of applications that use large language models.
And it comes in both Python and JavaScript libraries.
It's it's essentially a generic interface for nearly any LLM.
So you have a centralized development environment
to build your large language model applications
and then integrate them with stuff
like data sources and software workflows.
Now, when it was launched by Harrison Chase in October 2022,
LangChain enjoyed a meteoric rise,
and by June of the following year,
it was the single fastest growing open source project on GitHub.
And while the LangChain hype train
has slightly cooled a little bit, there's plenty of utility here.
So let's take a look at its components.
So what makes up long chain?
Well, LangChain streamlines the programing of LLM applications
through something called abstractions.
Now, what do I mean by that?
Well, your thermostat that allows you to control the temperature in your home
without needing to understand all the complex circuitry that this entails.
We just set the temperature. That's an abstraction.
So LangChain's abstractions represent common steps
and concepts necessary to work with language models,
and they can be chained together to create applications,
minimizing the amount of code required to execute complex NLP tasks.
So let's start with the LLM module.
Now, nearly any LLM can be used in LangChain.
You just need an API key.
The LLM class is designed to provide a standard interface for all models,
so pick an LLM of your choice.
Be that a closed source one like GPT-4 or an open source one like Llama 2,
or, this being LangChain, pick both.
Okay, what else we got?
We have prompts.
Now prompts are the instructions given to a large language model
and the prompt template class in LangChain
formalizes the composition of prompts
without the need to manually hard code context and queries.
A prompt template can contain instructions like,
"Do not use technical terms in your response".
That would be a good one.

Or it could be a set of examples to guide its responses.
That's called few-shot prompting.
Or it could specify an output format.
Now, chains, as the name implies,
are the core of the LangChain's workflows.
They combine LLMs with other components,
creating applications by executing a sequence of functions.
So let's say a application that needs to, first of all, retrieve data from a website,
then it needs to summarize the text it gets back,
and then finally it needs to use that summary to answer user submitted questions.
That's a sequential chain where the output of one function
acts as the input to the next, and
each function in the chain could use different prompts,
different parameters, and even different models.
Now, to achieve certain tasks,
LLMs might need to access specific external data sources
that are not included in the training data set of the LLM itself.
So things like internal documents or emails that sort of thing.
Now, LangChain collectively refers to this sort of documentation as indexes,
and there are a number of them.
So let's take a look at a few.
Now, one of them is called a document loader,
and the document loaders, they work with third party applications
for importing data sources from sources like file storage services.
So think Dropbox or Google Drive,
or web content, from like YouTube transcripts,
or collaboration tools like Airtable, or databases like Pandas and MongoDB.
There's also support for vector databases as well.
Now, unlike traditional structured databases,
vector databases represent data points by converting them into something called vector embeddings,
which are numerical representations in the form of vectors with a fixed number of dimensions.
And you can store a lot of information in this format
as it's a very efficient means of retrieval.
There are also something called text splitters,
which can be very useful as well
because they can split text up into small, semantically meaningful chunks
that can then be combined using the methods and parameters of your choosing.
Now, LLMs, by default, don't really have any long term memory of prior conversations,
and unless you happen to pass the chat history in as an input to your query.
But LangChain solves this problem
with simple utilities for adding in memory into your application,
and you have options for retaining like the entire conversations,
through to options to just retain a summarization that the conversation that we've had so far.
And then finally the last one will look at are agents.
Now, agents can use a given language model as a reasoning engine
to determine which actions to take.

And when building a chain for an agent,
you'll want to include inputs like a list of the available tools that you should use,
the user input like the prompts and the queries,
and then any other relevant previously executed steps.
So how can we put all of this to work for our applications?
Well, let's talk about a few LangChain use cases.
Now, obviously we have chatbots.
LangChain can be used to provide proper context for the specific use of a chatbot
and to integrate chat bots into existing communication channels and workflows with their own APIs.
We also have summarization.
Language models can be tasked with summarizing many types of text
from breaking down complex academic papers and transcripts,
to providing just a digest of incoming emails.
Processing lots of examples where this is used for question answering.
So using specific documents or specialized knowledge bases,
LLMs can retrieve the relevant information from the storage and then articulate
helpful answers using the information that would otherwise not have been in their training dataset.
And, yeah, this is a good one, data augmentation.
LLMs can be used to generate synthetic data for use in machine learning.
So, for example, a LLM can be trained to generate additional samples
that closely resemble the real data points in a training dataset.
And there are, of course, virtual agents, as we already started to discuss.
Integrating with the right workflows,
LangChain's agent modules can use an LLM to autonomously determine the next steps,
and then take the action that it needs to complete that step
using something called RPA, or robotic process automation.
LangChain is open source and free to use.
There are also related frameworks like LangServe for creating chains as REST APIs
and then LangSmith, which provides tools to monitor, evaluate and debug applications.
Essentially LangChain's tools and APIs
simplify the process of building applications
that make use of large language models.
If you have any questions, please drop us a line below
and if you want to see more videos like this in the future,
please like and subscribe.
Thanks for watching.