blank chain what is it why should you use it and how does it work let's have a look
Lang chain is an open source framework that allows developers working with AI to combine large language models like gbt4 with external sources of computation and data the framework is currently offered as a python or a JavaScript package typescript to be specific in this video we're going to start unpacking the python framework and we're going to see why the popularity of the framework is exploding right now especially after the introduction of gpt4 in March 2023 to understand what

need Lang chain fills let's have a look at a practical example so by now we all know that chat typically or tpt4 has an impressive general knowledge we can ask it about almost anything and we'll get a pretty good answer
suppose you want to know something specifically from your own data your own document it could be a book a PDF file a database with proprietary information link chain allows you to connect a large language model like dbt4 to your own sources of data and we're not talking about pasting a snippet of a text document into the chativity prompt we're talking about referencing an entire database filled with your own data and not only that once you get the information you need you can have Lang chain help you take the action you want to take for instance send an email with some specific information
and the way you do that is by taking the document you want your language model to reference and then you slice it up into smaller chunks and you store those chunks in a Victor database the chunks are stored as embeddings meaning they are vector representations of the text this allows you to build language model

applications that follow a general pipeline a user asks an initial question this question is then sent to the language model and a vector representation of that question is used to do a similarity search in the vector database this allows us to fetch the relevant chunks of information from the vector database and feed that to the language model as well
now the language model has both the initial question and the relevant information from the vector database and is therefore capable of providing an answer or take an action
a link chain helps build applications that follow a pipeline like this and these applications are both data aware we can reference our own data in a vector store and they are authentic they can take actions and not only provide answers to questions
and these two capabilities open up for an infinite number of practical use cases anything involving personal assistance will be huge you can have a large language model book flights transfer money pay taxes now imagine the implications for studying and learning new things you can have a large language model reference an entire syllabus and help you learn the material as fast as possible coding data analysis data science is all going to be affected by this
one of the applications that I'm most excited about is the ability to connect large language models to existing company data such as customer data marketing data and so on
I think we're going to see an exponential progress in data analytics and data science our ability to connect the large language models to Advanced apis such as metas API or Google's API is really gonna gonna make things take off
so the main value proposition of Lang

chain can be divided into three main Concepts
we have the llm wrappers that allows us to connect to large language models like gbt4 or the ones from hugging face
prompt templates allows us to avoid having to hard code text which is the input to the llms
then we have indexes that allows us to extract relevant information for the llms the chains allows us to combine multiple components together to solve a specific task and build an entire llm application
and finally we have the agents that allow the llm to interact with external apis
there's a lot to unpack in Lang chain and new stuff is being added every day but on a high level this is what the framework looks like we have models or wrappers around models we have problems we have chains we have the embeddings and Vector stores which are the indexes and then we have the agents so what I'm going to do now is I'm going to start unpacking each of these elements by writing code and in this video I'm going to keep it high level just to get an overview of the framework and a feel for the different elements first thing we're

going to do is we're going to pip install three libraries we're going to need python.in to manage the environment file with the passwords we're going to install link chain and we're going to install the Pinecone client Pinecone is going to be the vector store we're going to be using in this video in the environment file we need the open AI API key we need the pine cone environment and we need the pine cone API key foreign once you have signed up for a Pinecone account it's free the API keys and the environment name is easy to find same thing is true for openai just go to

platform.orgmaili.com account slash API keys
let's get started so when you have the keys in an environment file all you have to do is use node.n and find that in to get the keys and now we're ready to go so we're going to start off with the

llms or the wrappers around the llms then I'm going to import the open AI Rubber and I'm going to instantiate the text DaVinci 003 completion model and ask it to explain what a large language model is and this is very similar to when you call the open AI API directly
next we're going to move over to the chat model so gbt 3.5 and gbt4 are chat models
and in order to interact with the chat model through link chain we're going to import a schema consisting of three parts an AI message a human message and a system message
and then we're going to import chat open AI the system message is what you use to configure the system when you use a model and the human message is the user message
thank you
to use the chat model you combine the system message and the human message in a list and then you use that as an input to the chat model
here I'm using GPT 3.5 turbo you could have used gpt4 I'm not using that because the open AI service is a little bit Limited at the moment
so this works no problem let's move to the next concept which is prompt

templates so prompts are what we are going to send to our language model but most of the time these problems are not going to be static they're going to be dynamic they're going to be used in an application and to do that link chain has something called prompt templates and what that allows us to do is to take

a piece of text and inject a user input
into that text and we can then format
The Prompt with the user input and feed
that to the language model
so this is the most basic example but it
allows us to dynamically change the
prompt with the user input
the third concept we want to Overlook at
is the concept of a chain

a chain takes a language model and a
prompt template and combines them into
an interface that takes an input from
the user and outputs an answer from the
language model sort of like a composite
function where the inner function is the
prompt template and the outer function
is the language model
we can also build sequential chains
where we have one chain returning an
output and then a second chain taking
the output from the first chain as an
input
so here we have the first chain that
takes a machine learning concept and
gives us a brief explanation of that
concept the second chain then takes the
description of the first concept and
explains it to me like I'm five years
old
then we simply combine the two chains
the first chain called chain and then
the second chain called chain two into
an overall chain
and run that chain
and we see that the overall chain
returns both the first description of
the concept and the explain it to me
like I'm 5 explanation of the concept
all right let's move on to embeddings

and Vector stores but before we do that
let me just change the explainer to me
like I'm five prompt so that we get a
few more words
I'm gonna go with 500 Words
all right so this is a slightly longer
explanation for a five-year-old

now what I'm going to do is I'm going to check this text and I'm going to split it into chunks because we want to store it in a vector store in Pinecone and Lang chain has a text bitter tool for that so I'm going to import recursive character text splitter and then I'm going to spit the text into chunks
like we talked about in the beginning of the video
we can extract the plain text of the individual elements of the list with page content
and what we want to do now is we want to turn this into an embedding which is just a vector representation of this text and we can use open ai's embedding model Ada
with all my eyes model we can call embed query on the raw text that we just extracted from the chunks of the document and then we get the vector representation of that text or the embedding
now we're going to check the chunks of the explanation document and we're going to store the vector representations in pine cone
so we'll import the pine cone python client and we'll import pine cone from Lang chain Vector stores and we initiate the pine cone client with the key and the environment that we have in the environment file
then we take the variable texts which consists of all the chunks of data we want to store we take the embeddings model and we take an index name and we load those chunks on the embeddings to Pine Cone and once we have the vector stored in Pinecone we can ask questions about the data stored what is magical about an auto encoder and then we can do a similarity search in Pinecone to get the answer or to extract all the relevant chunks
if we head over to Pine Cone we can see

that the index is here we can click on
it and inspect it
check the index info we have a total of
13 vectors in the vector store

all right so the last thing we're going
to do is we're going to have a brief
look at the concept of an agent
now if you head over to open AI chat GPT
plugins page you can see that they're
showcasing a python code interpreter
now we can actually do something similar
in langtune
so here I'm importing the create python
agent as well as the python Rebel tool
and the python webble from nankchain
then we instantiate a python agent
executor
using an open AI language model
and this allows us to having the
language model run python code
so here I want to find the roots of a
quadratic function and we see that the
agent executor is using numpy roots to
find the roots of this quadratic
function
alright so this video was meant to give
you a brief introduction to the Core
Concepts of langchain if you want to
follow along for a deep dive into the
concepts hit subscribe thanks for
watching