Extracting Knowledge Graphs From Text With GPT4o

Thu Vu
321K subscribers

Subscribe

5.9K | Share | Ask | Download | ...

133,106 views May 28, 2025 #ai #python #graph
Join me to Master Python for AI Projects https://python-course-earlybird.frame...
Github repo for this project https://github.com/thu-vu92/knowledge...

TIMESTAMPS

0:00 - Intro
0:44 - Project - building knowledge graph from text
1:20 - What is a knowledge graph?
2:18 - Differences to traditional databases/ spreadsheets
3:25 - Applications of KGs
6:54 - Why was it challenging to create KGs?
8:01 - Quick Experiment: Neo4j LLM Knowledge Graph Builder
10:37 - Project tutorial - Creating KGs with OpenAI GPT4o in Python
22:46 - Conclusions

https://github.com/thu-vu92/knowledge-graph-llms/

thu-vu92 / knowledge-graph-llms

Type / to search

Code | Issues 1 | Pull requests | Actions | Projects | Security | Insights

knowledge-graph-llms Public

Watch 10 | Fork 138 | Star 533

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

thu-vu92 Update README.md 2306166 · 3 months ago | 4 Commits

LICENSE	Create LICENSE	3 months ago
README.md	Update README.md	3 months ago
app.py	Add files	3 months ago
generate_knowledge_graph.py	Add files	3 months ago
knowledge_graph.html	Add files	3 months ago
knowledge_graph.ipynb	Add files	3 months ago
requirements.txt	Add files	3 months ago

About

In this project, I explored how to extract knowledge graphs from text using LLMs, such as OpenAI GPT4o.

Readme
MIT license
Activity
533 stars
10 watching
138 forks
Report repository

Releases

No releases published

Packages

No packages published

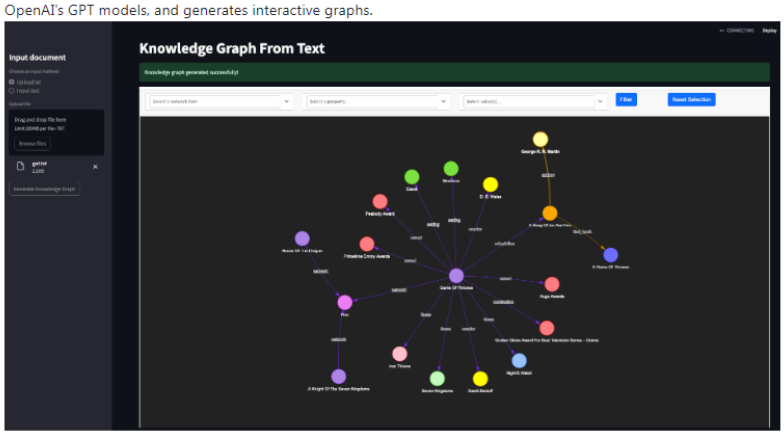
Languages

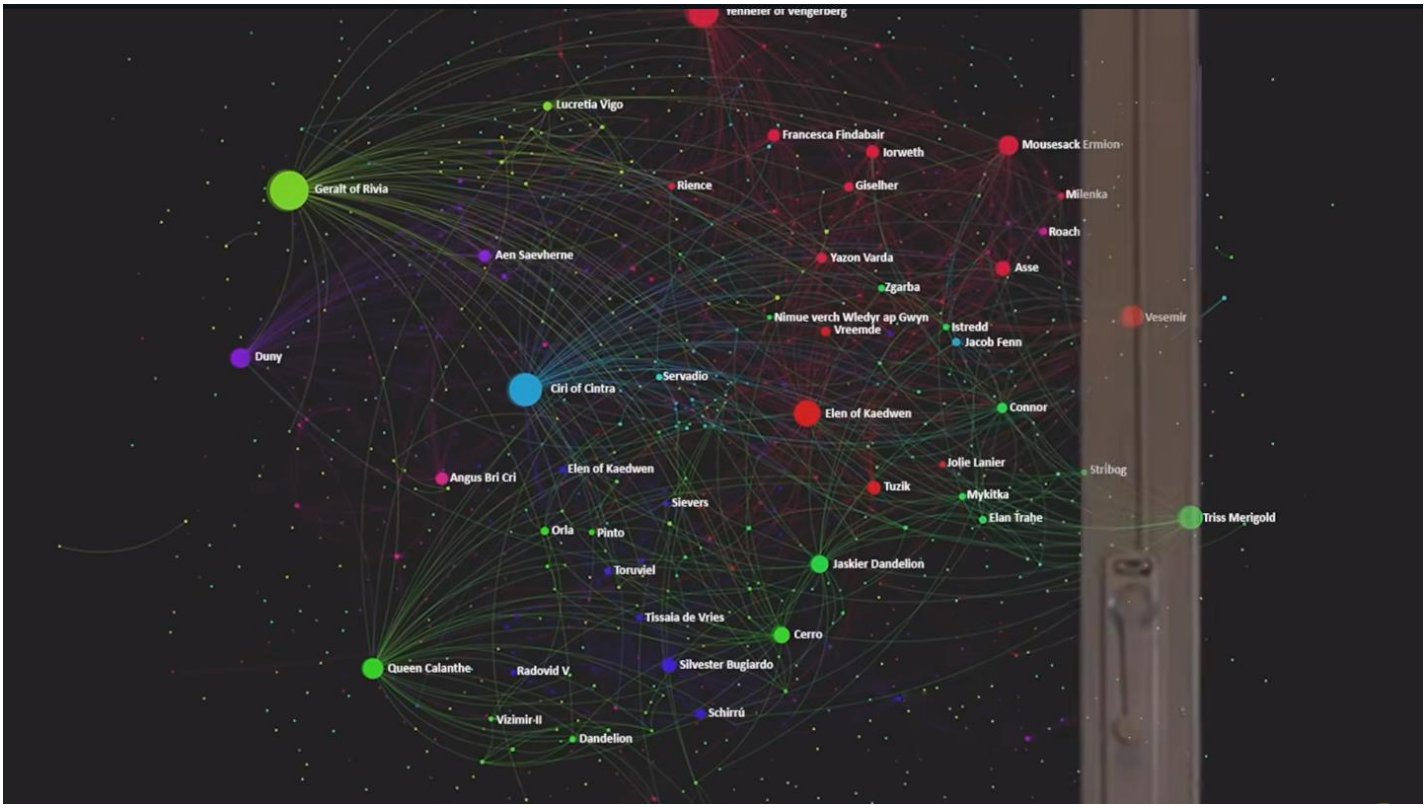
Jupyter Notebook 46.4% | HTML 41.6%
Python 12.0%

README | MIT license

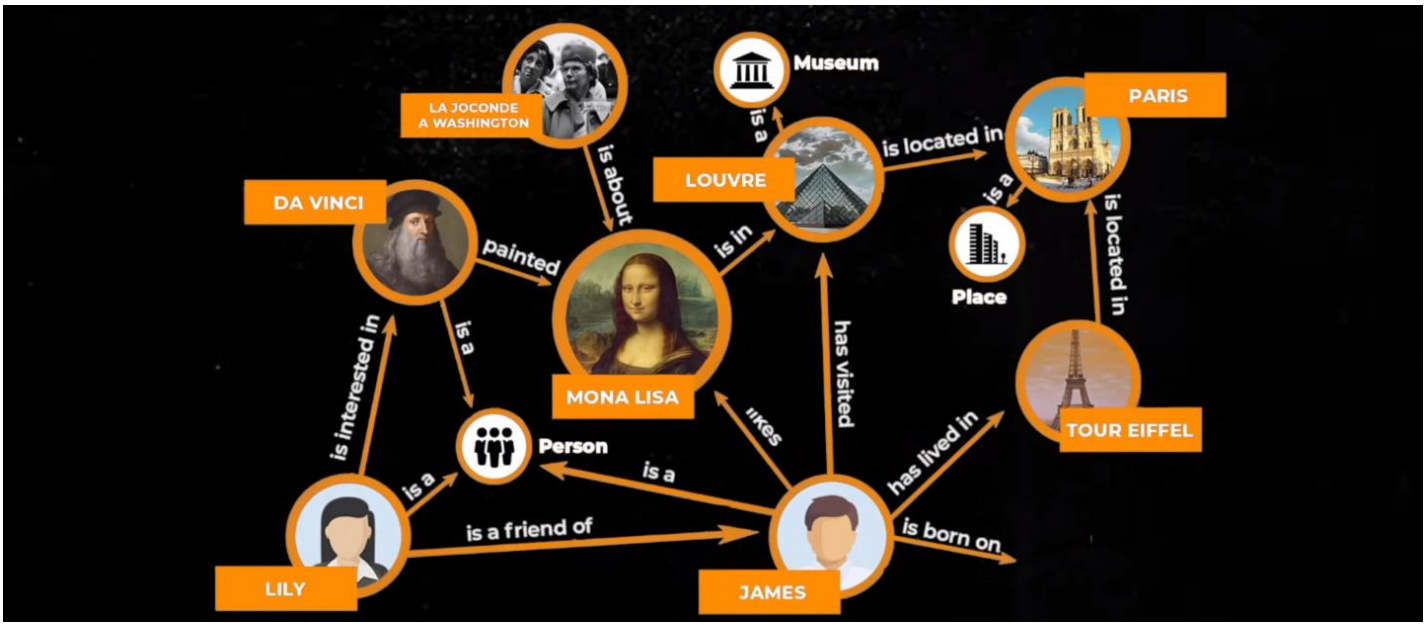
Knowledge Graph Generator

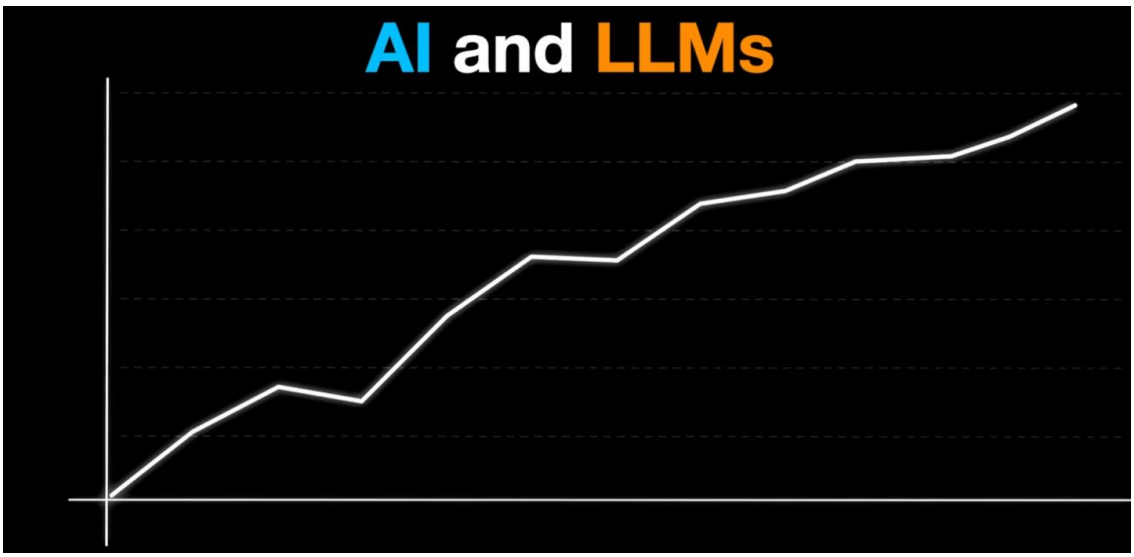
A Streamlit application that extract graph data (entities and relationships) from text input using LangChain and OpenAI's GPT models, and generates interactive graphs.

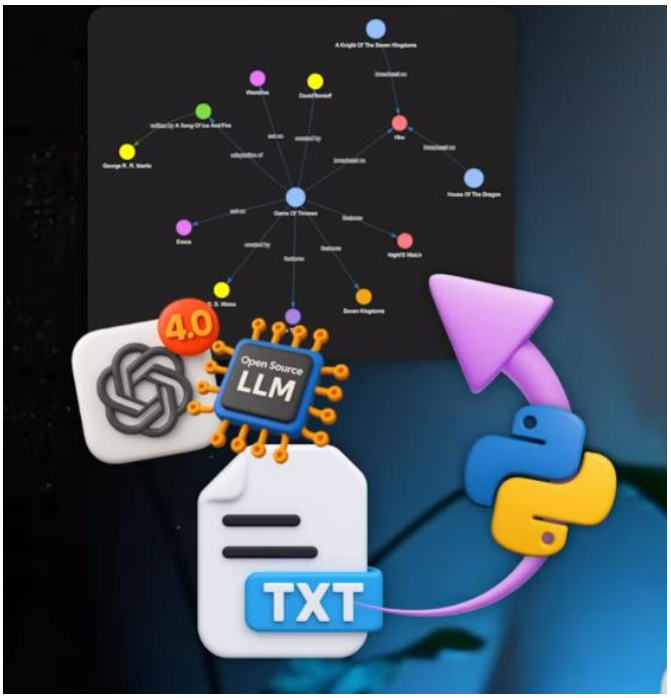




KNOWLEDGE GRAPH

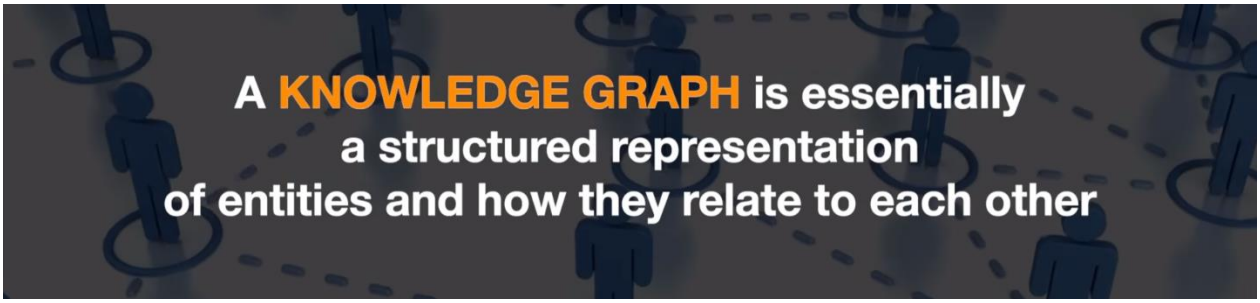

LA JOCONDE A WASHINGTON
Museum
is a
PARIS
is located in
LOUVRE
is a
is about
DA VINCI
painted
is in
Place
is located in
is interested in
is a
MONA LISA
has visited
likes
TOUR EIFFEL
Person
is a
has lived in
is a
is a friend of
is born on
LILY
JAMES

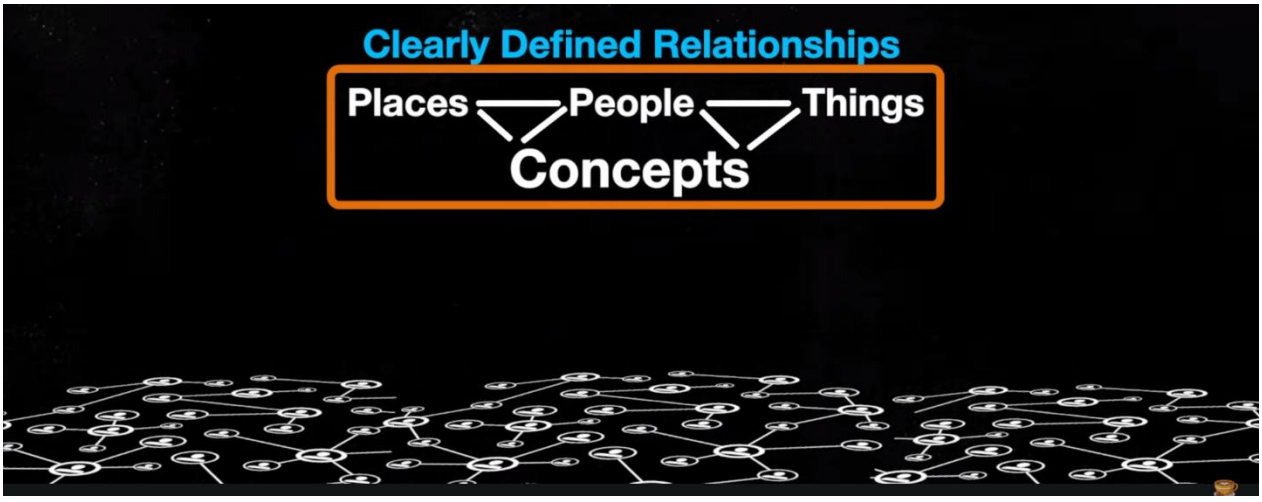

AI and LLMs


4.0
Open Source
LLM
TXT

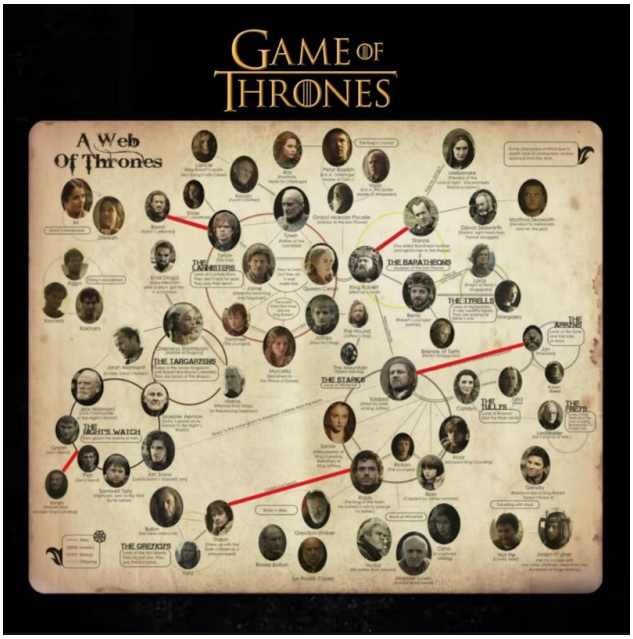

In the end...
TXT


In the end...
TXT

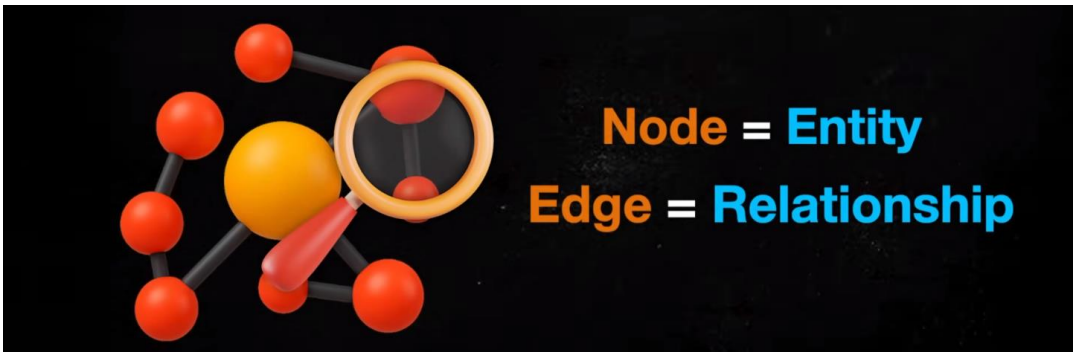

A KNOWLEDGE GRAPH is essentially
a structured representation
of entities and how they relate to each other

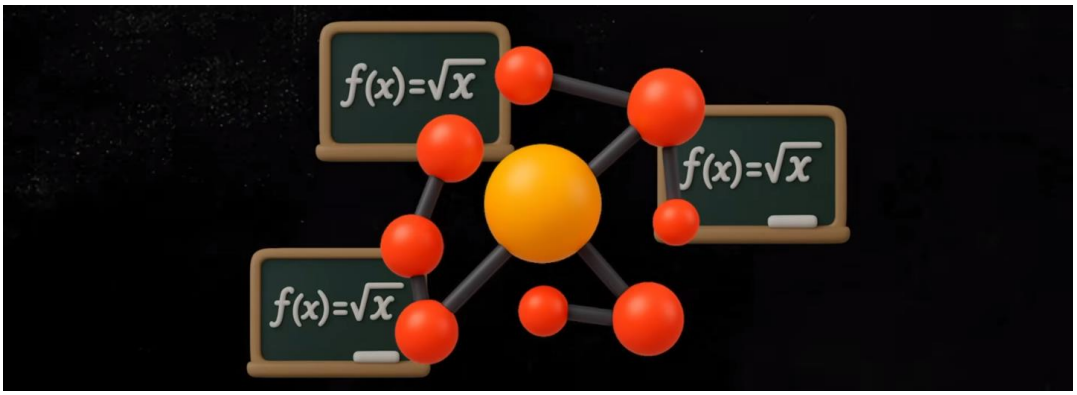

Clearly Defined Relationships
Places
People
Things
Concepts

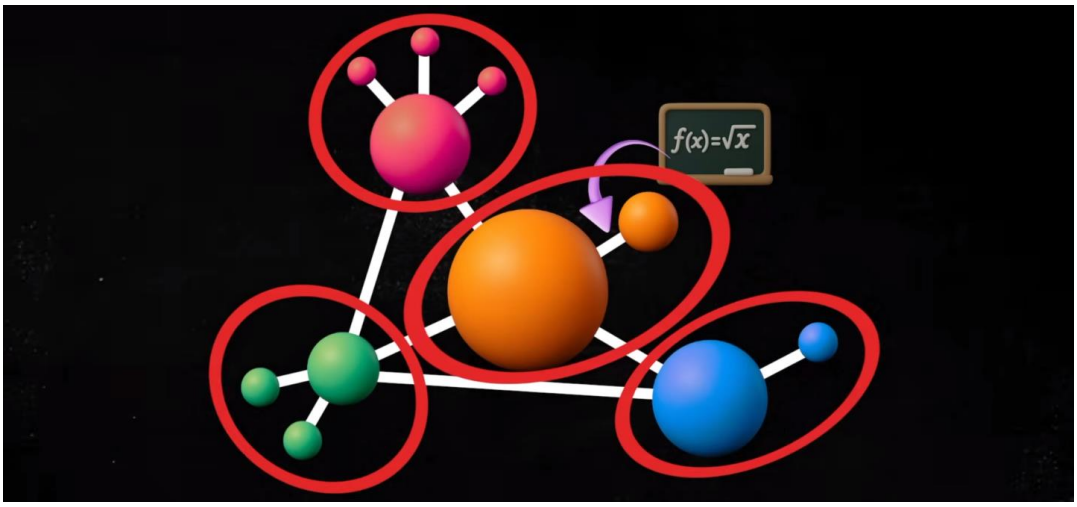

GAME OF THRONES
YOU WIN OR YOU DIE
AN EPIC NEW SERIES ONLY ON HBO SUNDAY, APRIL 17, 9PM


GAME OF THRONES
A Web Of Thrones

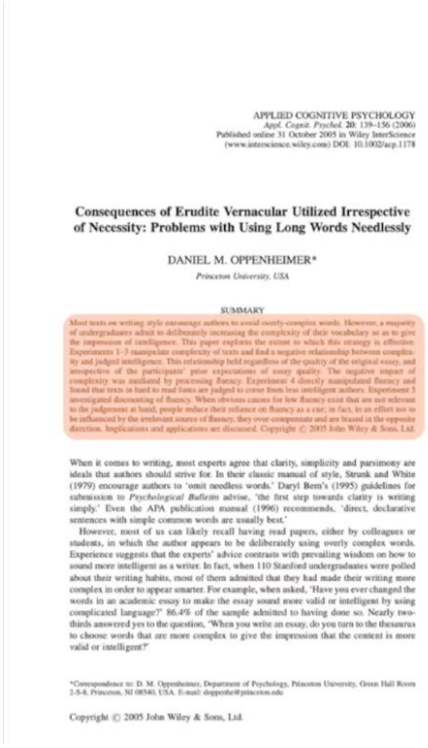

SQL


Node = Entity
Edge = Relationship


f(x)=√x
f(x)=√x
f(x)=√x


f(x)=√x

RAG



APPLIED COGNITIVE PSYCHOLOGY
Appl. Cognit. Psychol. **20**: 139–156 (2006)
Published online 31 October 2005 in Wiley InterScience (www.interscience.wiley.com) DOI: 10.1002/acp.1178

Consequences of Erudite Vernacular Utilized Irrespective of Necessity: Problems with Using Long Words Needlessly

DANIEL M. OPPENHEIMER*
Princeton University, USA

SUMMARY
Most texts on writing style encourage authors to avoid overly-complex words. However, a majority of undergraduates admit to deliberately increasing the complexity of their vocabulary so as to give the impression of intelligence. This paper explores the extent to which this strategy is effective. Experiments 1–3 manipulate complexity of texts and find a negative relationship between complexity and judged intelligence. This relationship held regardless of the quality of the original essay, and irrespective of the participants' prior expectations of essay quality. The negative impact of complexity was mediated by processing fluency. Experiment 4 directly manipulated fluency and found that texts in hard to read fonts are judged to come from less intelligent authors. Experiment 5 investigated discounting of fluency. When obvious causes for low fluency exist that are not relevant to the judgement at hand, people reduce their reliance on fluency as a cue; in fact, in an effort not to be influenced by the irrelevant source of fluency, they over-compensate and are biased in the opposite direction. Implications and applications are discussed. Copyright © 2005 John Wiley & Sons, Ltd.

When it comes to writing, most experts agree that clarity, simplicity and parsimony are ideals that authors should strive for. In their classic manual of style, Strunk and White (1979) encourage authors to 'omit needless words.' Daryl Bem's (1995) guidelines for submission to *Psychological Bulletin* advise, 'the first step towards clarity is writing simply.' Even the APA publication manual (1996) recommends, 'direct, declarative sentences with simple common words are usually best.'

However, most of us can likely recall having read papers, either by colleagues or students, in which the author appears to be deliberately using overly complex words. Experience suggests that the experts' advice contrasts with prevailing wisdom on how to sound more intelligent as a writer. In fact, when 110 Stanford undergraduates were polled about their writing habits, most of them admitted that they had made their writing more complex in order to appear smarter. For example, when asked, 'Have you ever changed the words in an academic essay to make the essay sound more valid or intelligent by using complicated language?' 86.4% of the sample admitted to having done so. Nearly two-thirds answered yes to the question, 'When you write an essay, do you turn to the thesaurus to choose words that are more complex to give the impression that the content is more valid or intelligent?'

*Correspondence to: D. M. Oppenheimer, Department of Psychology, Princeton University, Green Hall Room 2-S-8, Princeton, NJ 08540, USA. E-mail: doppenhe@princeton.edu

There are many plausible reasons that the use of million-dollar words would lead readers to believe that an author is smart. Intelligence and large vocabularies are positively correlated (Spearman, 1904). Therefore, by displaying a large vocabulary, one may be providing cues that he or she is intelligent as well. Secondly, writers are assumed to be conforming to the Gricean maxim of manner, 'avoid obscurity of expression' (Grice, 1975). If authors are believed to be writing as simply as possible, but a text is nonetheless complex, a reader might believe that the ideas expressed in that text are also complex, defying all attempts to simplify the language. Further, individuals forced to struggle through a complex text might experience dissonance if they believe that the ideas being conveyed are simple (Festinger, 1957). Thus, individuals might be motivated to perceive a difficult text as being more worthwhile, thereby justifying the effort of processing.

Indeed, there is some evidence that complex vocabulary can be indicative of a more intelligent author. For example, Pennebaker and King (1999) have shown that the percentage of long words used in class assignments positively correlates with SAT scores and exam grades on both multiple choice and essay tests. However it is difficult to draw conclusions about the effectiveness of a strategy of complexity from this data. The study did not look at how readers of the texts containing the long words perceived the authors' intelligence. Thus, it is possible that although students using complex vocabularies are objectively very knowledgeable, they might nonetheless be perceived as being less so.

Why might we believe that the experts might be correct in recommending simplicity in writing? One theory that predicts the effectiveness of straightforward writing is that of processing fluency. Simpler writing is easier to process, and studies have demonstrated that processing fluency is associated with a variety of positive dimensions. Fluency leads to higher judgements of truth (Reber & Schwarz, 1999), confidence (Norwick & Epley, 2002), frequency (Tversky & Kahneman, 1973), fame (Jacoby, Kelley, Brown, & Jasechko, 1989), and even liking (Reber, Winkelman, & Schwarz, 1998). Furthermore, the effects of fluency are strongest when the fluency is discrepant—when the amount of experienced fluency is surprising (Whittlesea & Williams, 2001a, 2001b). As such, it would not be surprising if the lower fluency of overly complex texts caused readers to have negative evaluations of those texts and the associated authors, especially if the complexity was unnecessary and thus surprising readers with the relative disfluency of the text.

Both the experts and prevailing wisdom present plausible views, but which (if either) is correct? The present paper provides an empirical investigation of the strategy of complexity, and finds such a strategy to be unsuccessful. Five studies demonstrate that the loss of fluency due to needless complexity in a text negatively impacts raters' assessments of the text's authors.

EXPERIMENT 1

Experiment 1 aimed to answer several simple questions. First, does increasing the complexity of text succeed in making the author appear more intelligent? Second, to what extent does the success of this strategy depend on the quality of the original, simpler writing? Finally, if the strategy is unsuccessful, is the failure of the strategy due to loss of fluency? To answer these questions, graduate school admission essays were made more complex by substituting some of the original words with their longest applicable thesaurus entries.

While word length is not perfectly interchangeable with sentence complexity—for example, complexity can come from grammatical structure or infrequent words as well—it is a useful proxy. Using length as a manipulation of complexity allows for a simple, easily replicable word replacement algorithm. By keeping content constant and varying the complexity of vocabulary, it was possible to investigate the effectiveness of complexity.

Participants and procedure

Seventy-one Stanford University undergraduates participated to fulfil part of a course requirement. The survey was included in a packet of unrelated one-page questionnaires. Packets were distributed in class, and participants were given a week to complete the entire packet.

Stimuli and design

Six personal statements for admissions to graduate studies in English Literature were downloaded from writing improvement websites. The essays varied greatly both in content and quality of writing. Logical excerpts ranging from 138 to 253 words in length were then taken from each essay. A 'highly complex' version of each excerpt was prepared by replacing every noun, verb and adjective with its longest entry in the Microsoft Word 2000 thesaurus. Words that were longer than any thesaurus entry, were not listed in the thesaurus, or for which there was no entry with the same linguistic sense were not replaced. If two entries were of the same length, the replacement was chosen alphabetically. When necessary, minor modifications were made to the essay to maintain the grammatical structure of a sentence (e.g. replacing 'an' with 'a' for replacement words beginning with consonants). A 'moderately complex' version of each excerpt was created using the same algorithm as above, except replacing only every third applicable word. Examples of the stimuli can be found in the appendix.

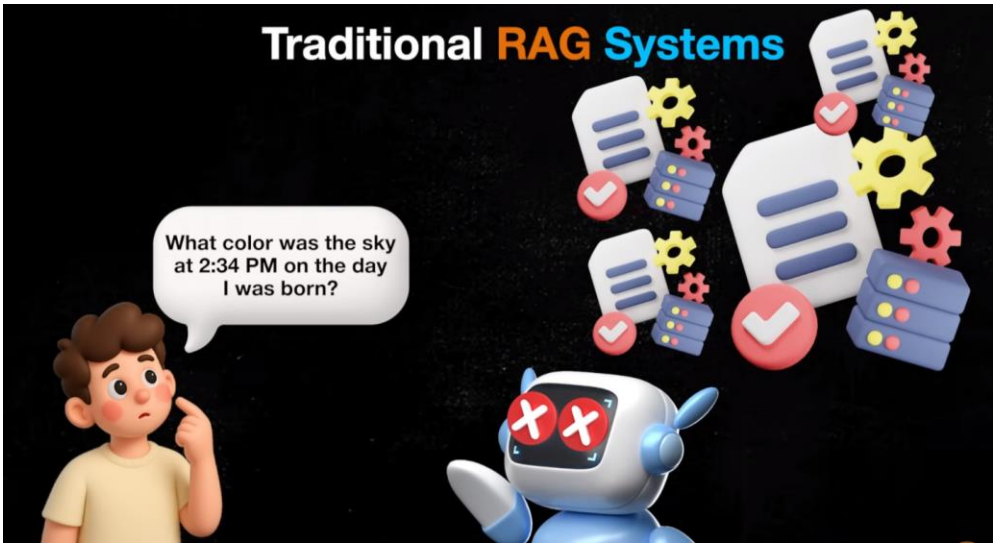
Each participant received only one excerpt. Participants were informed that the excerpt came from a personal statement for graduate study in the Stanford English department. They were instructed to read the passage, decide whether or not to accept the applicant, and rate their confidence in their decision on a 7-point scale.[1] They were then asked how difficult the passage was to understand, also on a seven-point scale.

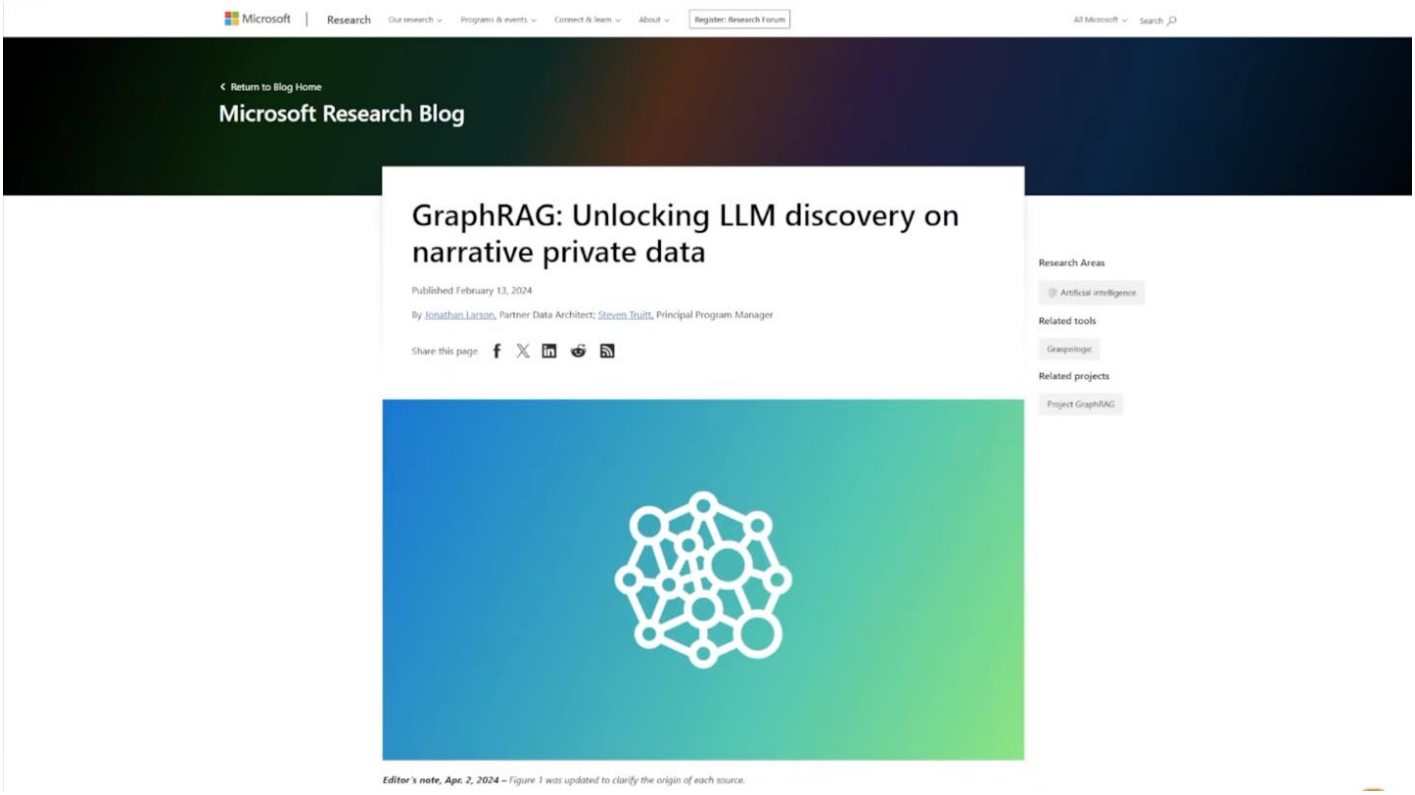
Results

The data of one participant was discarded due to an illegible answer. Analysis of the manipulation check showed that more complex texts were more difficult to read. ($\alpha = 2.9$, 4.0 and 4.3 for simple, moderately complex and highly complex, respectively). These differences were reliable, $F(2, 68) = 4.46$, $p < 0.05$, Cohen's $f = 0.18$. For other analyses, acceptance ratings (+1 for accept, −1 for reject) were multiplied by confidence ratings to create a −7 to 7 scale of admission confidence. Level of complexity had a reliable influence on admission confidence ratings, $F(2, 70) = 2.46$, $p < 0.05$, Cohen's $f = 0.12$.

[1]With the exception of the dichotomous admissions decision, all dependent measures reported in this paper are seven point scales ranging from 1 = 'not at all' to 7 = 'very'.


Traditional RAG Systems
What color is the sky?
It's blue!


Traditional RAG Systems
What color was the sky at 2:34 PM on the day I was born?


Microsoft | Research
Return to Blog Home
Microsoft Research Blog
GraphRAG: Unlocking LLM discovery on narrative private data
Published February 13, 2024
By Jonathan Larson, Partner Data Architect; Steven Truitt, Principal Program Manager
Share this page
Research Areas
Artificial intelligence
Related tools
Graspologic
Related projects
Project GraphRAG
Editor's note, Apr. 2, 2024 – Figure 1 was updated to clarify the origin of each source.

The Graph is Used to

Create a bottom-up clustering that organizes the data hierarchically into semantic clusters.

Clustering

Holistic understanding

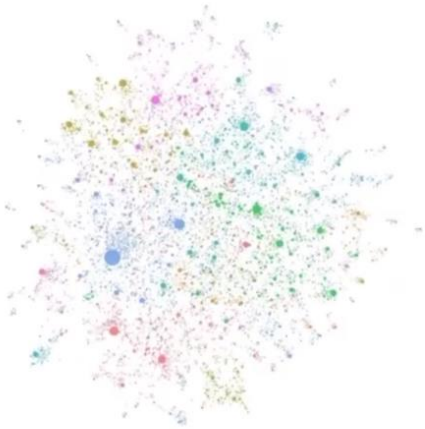
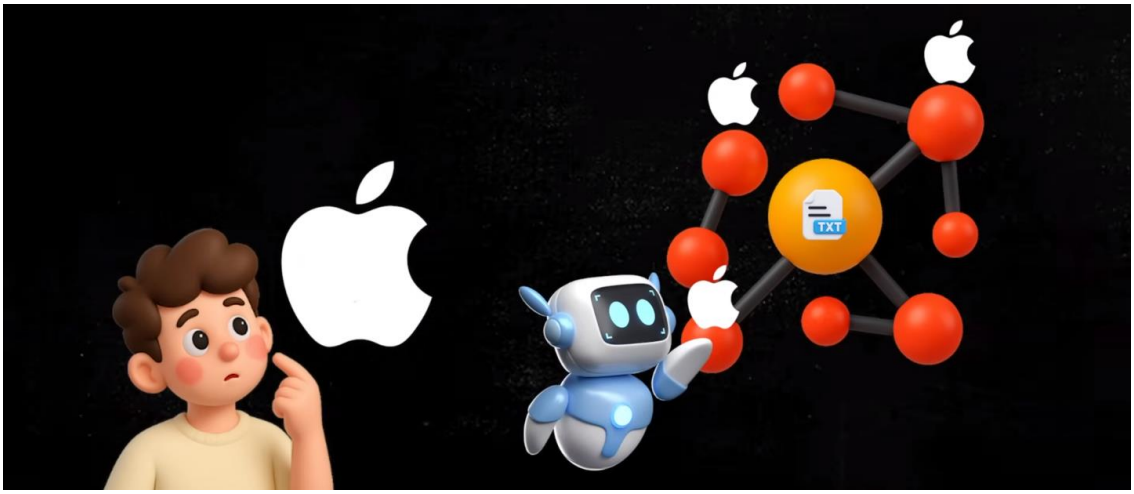


Figure 3: LLM-generated knowledge graph built from a private dataset using GPT-4 Turbo.

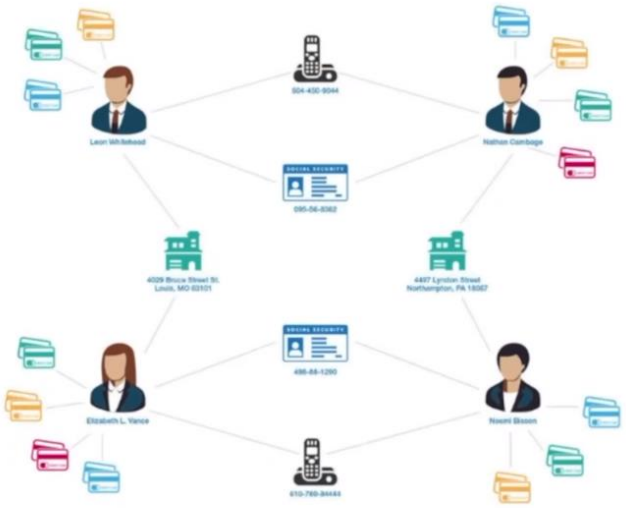


What are the TOP 5 themes in the data?

Baseline RAG	GraphRAG
1. Improving the quality of life in cities and urban development. 2. The meeting between Vladimir Putin and the President of the United Arab Emirates, Mohammed bin Zayed Al Nahyan. 3. The National Rating of the Investment Climate in the Regions of Russia. 4. The state of the Russian economy and the growth of domestic brands. 5. The discussion on the meaning of life and the role of science in understanding it.	The data presents a wide range of themes, but the top five most prevalent themes can be identified as follows: 1. **Conflict and Military Activity**: A significant portion of the data revolves around the ongoing conflict in Ukraine, with numerous reports detailing military activities, occupations, and attacks by various entities. This theme includes the actions of the Ukrainian military, Russian forces, and various other military units and entities [Data: Reports (513, 241, 151, 493, 343, +more)]. 2. **Political and Governmental Entities**: Many reports focus on political figures and governmental entities, highlighting their roles, relationships, and actions in the context of the current situation. This includes entities like the Kremlin, the President of Ukraine, the US State Department, and various regional administrations [Data: Reports (98, 493, 265, 129, 196, +more)]. 3. **Infrastructure and Environmental Concerns**: Several reports highlight potential environmental impacts and infrastructure threats. These include potential environmental damage due to industrial activities, flooding crises, and threats to infrastructure due to military conflicts [Data: Reports (308, 154, 201, 348, 216, +more)]. 4. **Community Analysis and Threat Assessment**: Several reports provide detailed analyses of specific communities, often centered around a particular location or entity. These analyses typically discuss the key entities within the community, their relationships, and any associated threats. Many reports also include assessments of potential threats associated with the entities or situations they discuss [Data: Reports (323, 596, 518, 583, 179, +more)]. 5. **Health, Humanitarian Concerns, and Emergency Situations**: Some reports focus on health and humanitarian issues, such as the Shereshevsky-Turner Syndrome community and the potential risk of intestinal diseases in the Odessa region. The data also covers emergency situations, including natural disasters, accidents, and attacks [Data: Reports (204, 149, 372, 154, 532, +more)]. These themes provide a comprehensive overview of the current situation, highlighting the complexity and interconnectedness of the issues at hand.

Fraud Detection

Below is an example of a simple fraud ring where two actors use their real street address, stolen social security numbers, and burner phones to create and use four synthetic identities with fake names to open the accounts:



In the example above the fake identities opened 14 credit cards. Assuming all the credit cards average out to a $8,000 limit, that is a potential $84,000 loss for any given bank, and this would be an example of an extremely small fraud ring in today's context.

Drug Research & Discovery

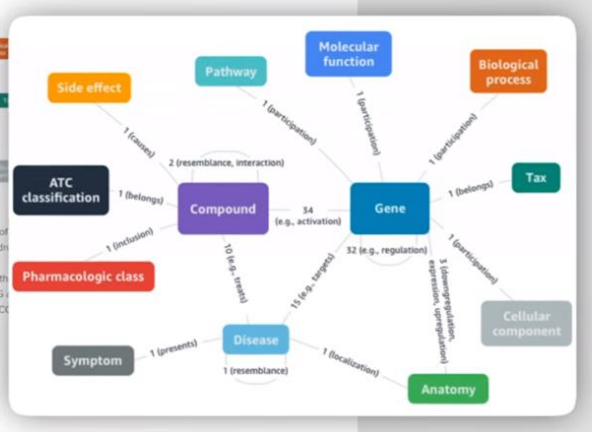
THE OHIO STATE UNIVERSITY

NING LAB
Biomedical Informatics
Computer Science and Engineering

HOME RESEARCH EDUCATION PUBLICATIONS MEMBERS SOFTWARE
DATASETS OPENINGS WEBPORTALS

DRKG: Drug Repositioning Knowledge Graph

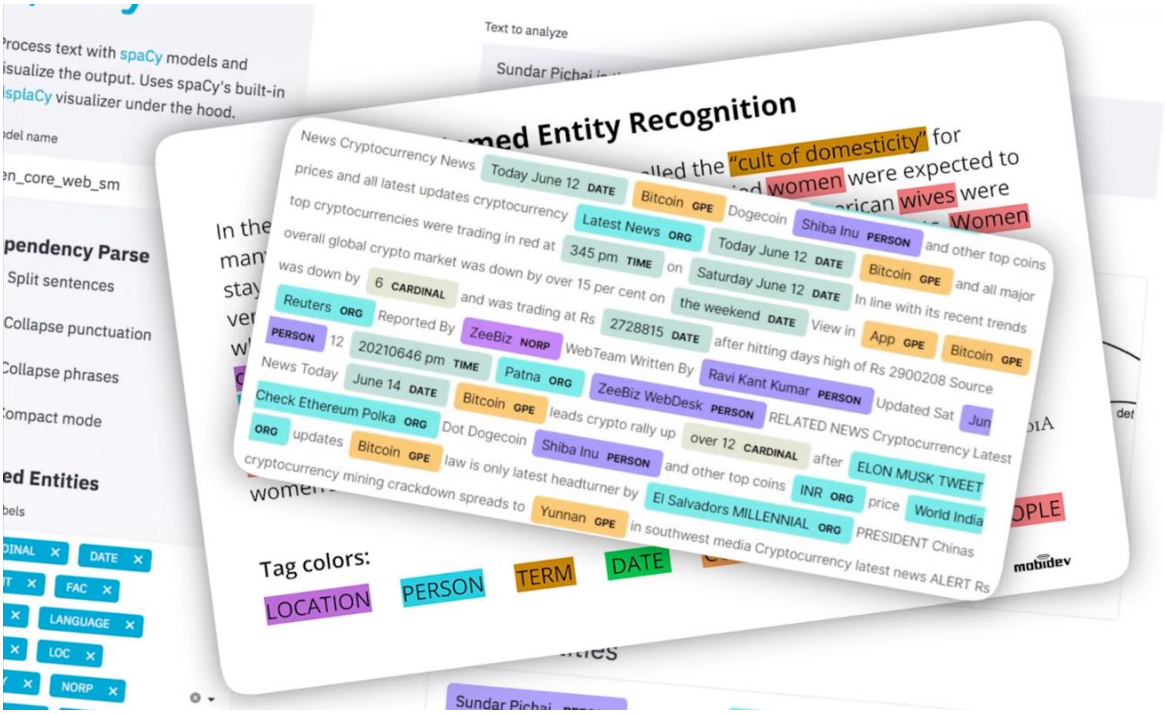
Release news at AWS



graph-learning methods (DGL-KE) that take advantage of
as PyTorch and MXNet) to predict the likelihood that a dr

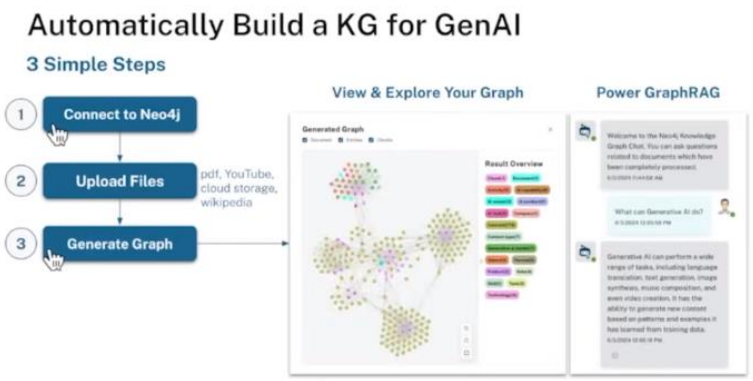
When tested against the human proteins associated with
19 drug candidates currently in clinical trials. Both DRKG
should help make computational drug repurposing for CO
effective.


4.0
Open Source
LLM


Text to analyze
Named Entity Recognition
Tag colors:
LOCATION
PERSON
TERM
DATE
Dependency Parse
Split sentences
Collapse punctuation
Collapse phrases
Compact mode
Named Entities
mobidev


Traditional ML Models
Struggle with:
Non-English languages
Nuances
Context
Ambiguity

Neo4j LLM Knowledge Graph Builder - Extract Nodes and Relationships from Unstructured Text



→

The Neo4j LLM Knowledge Graph Builder is an online application → for turning unstructured text into a knowledge graph, it provides a magical text to graph experience.

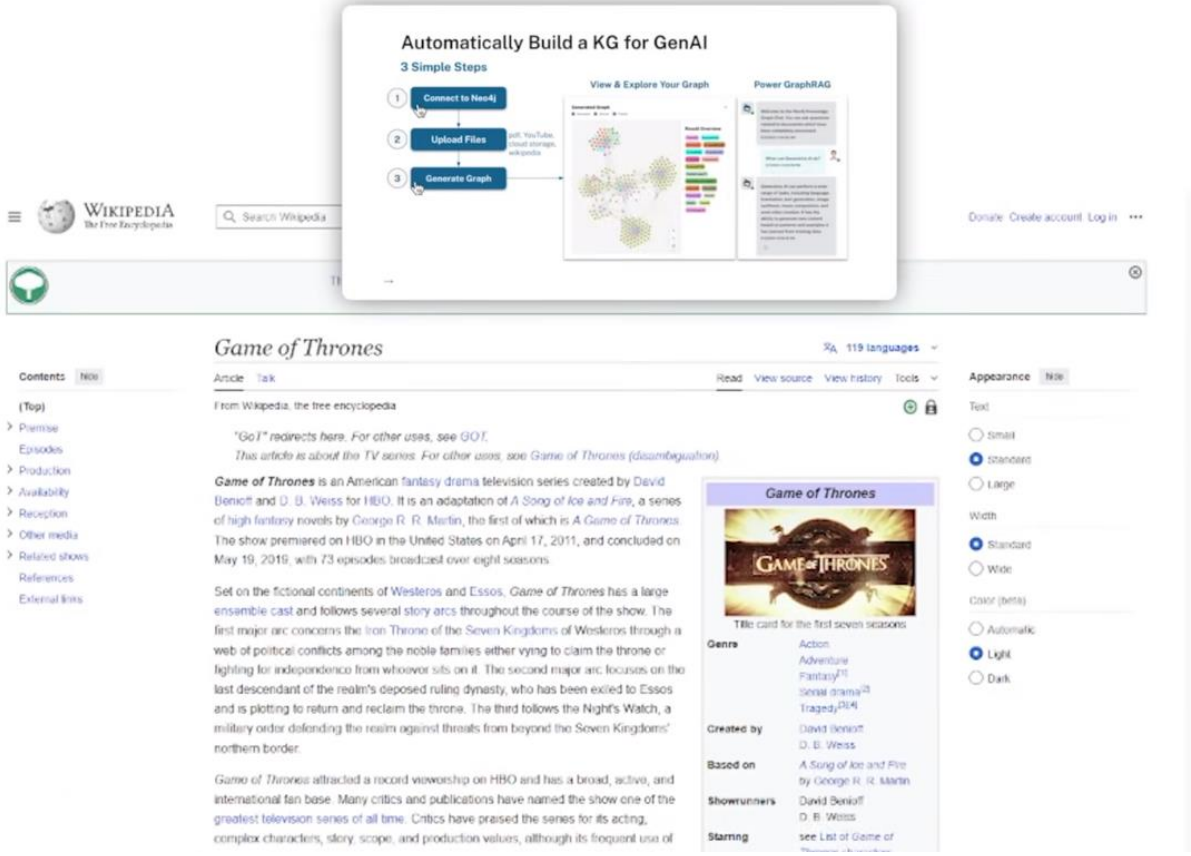
Contents

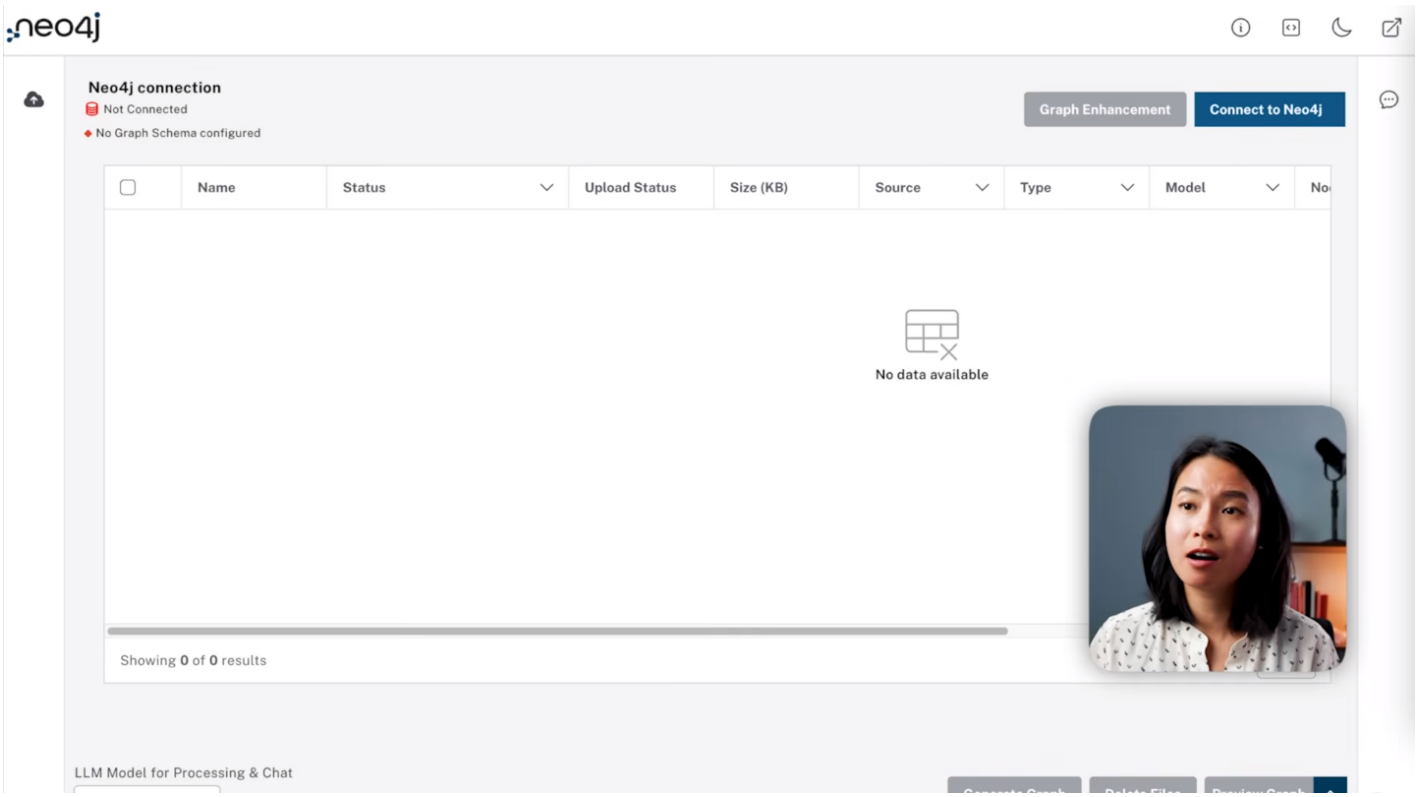
Nov 6 2025

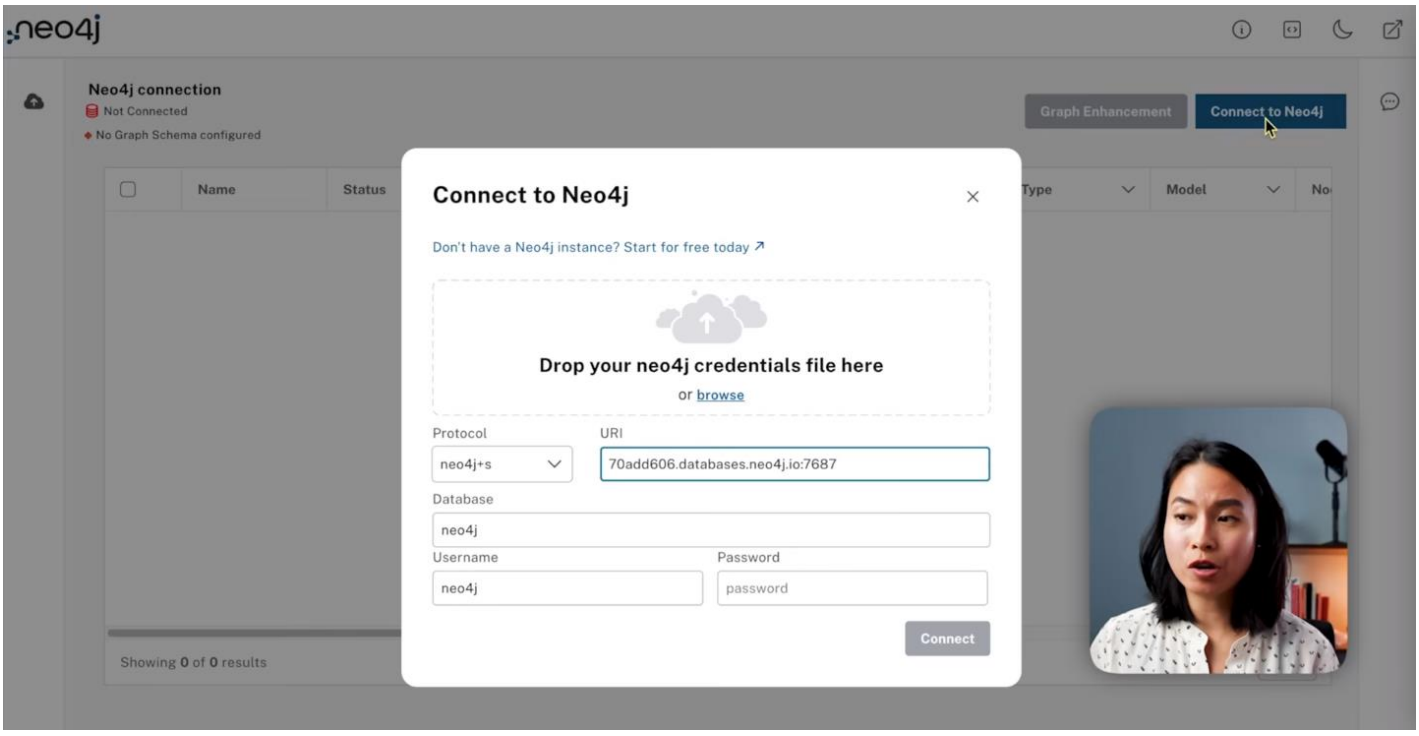
NODES 2025

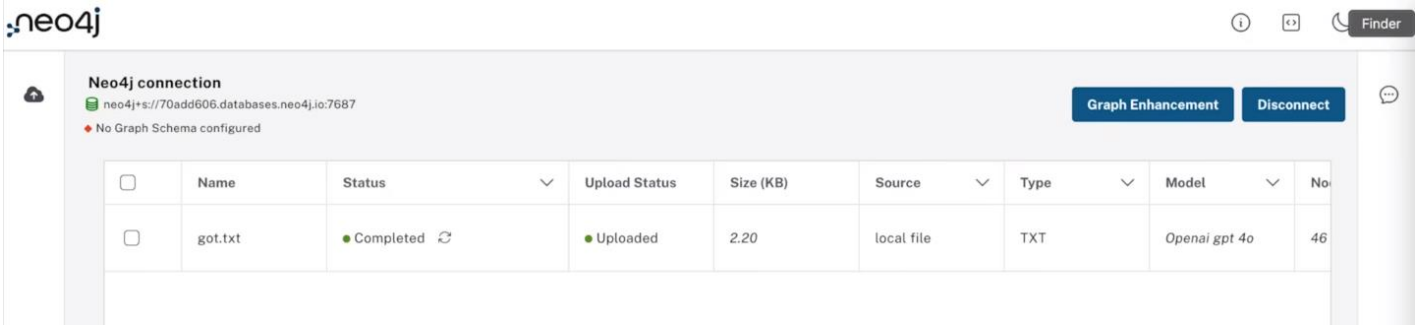
The Call for Papers is now open and we want to hear about your graph-related projects. Submit your talks by June 15

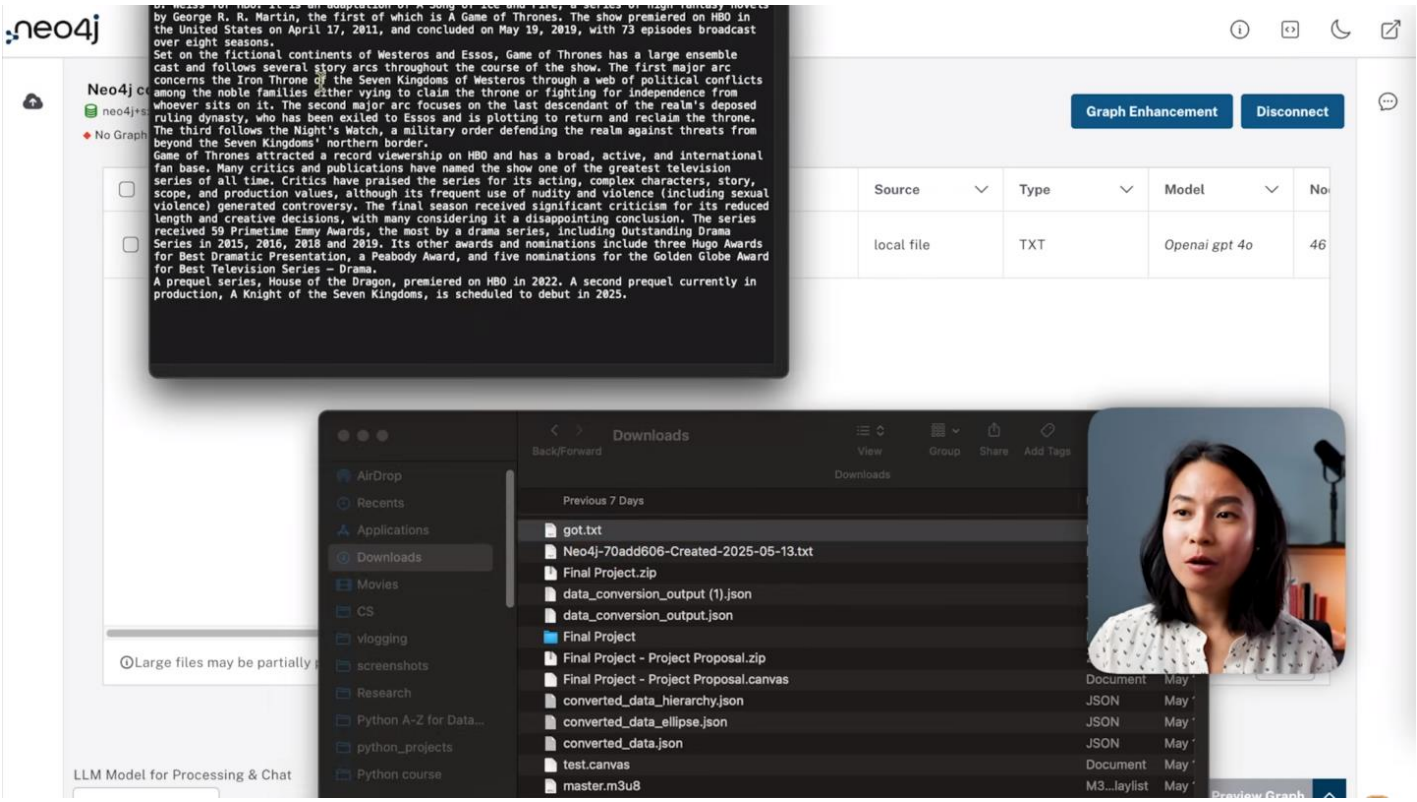
Submit your talk

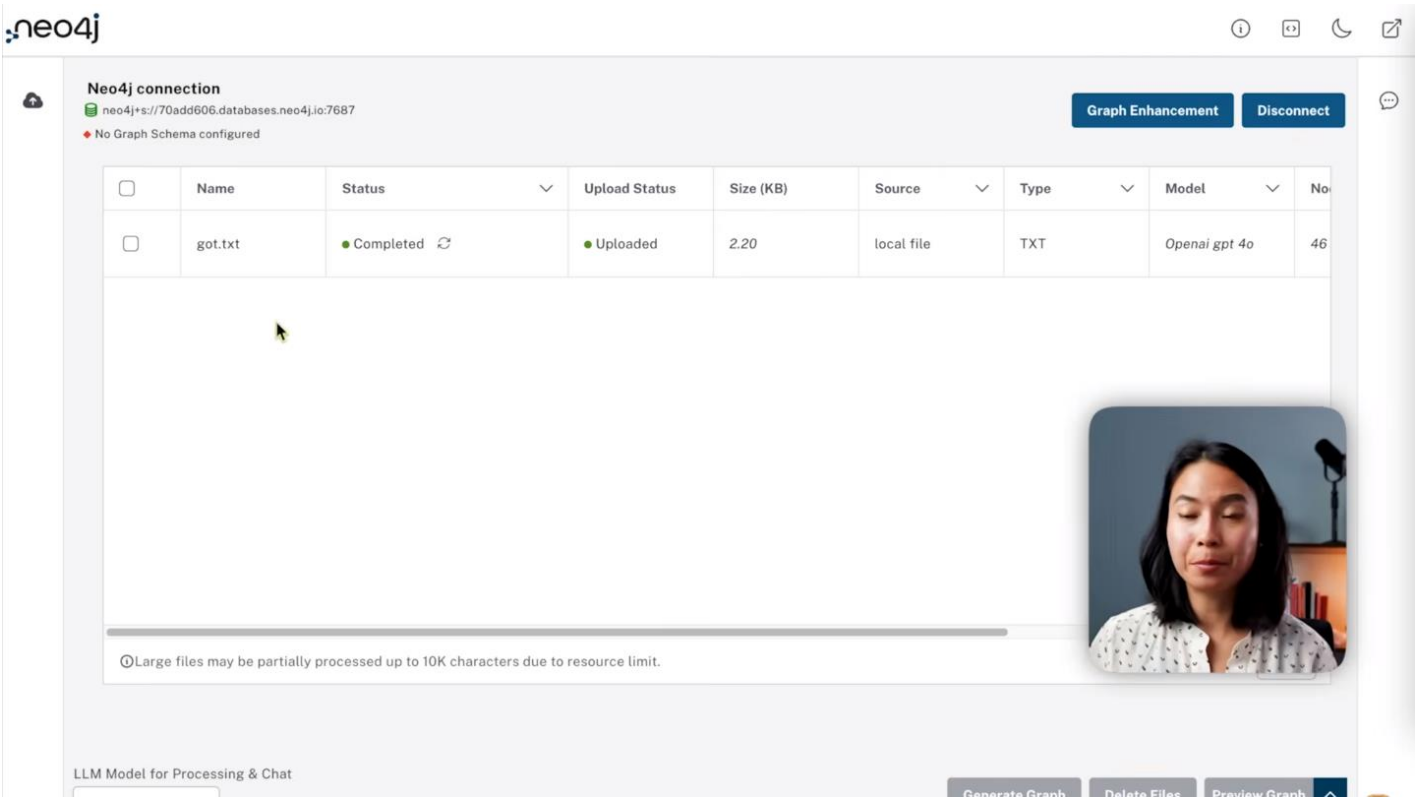


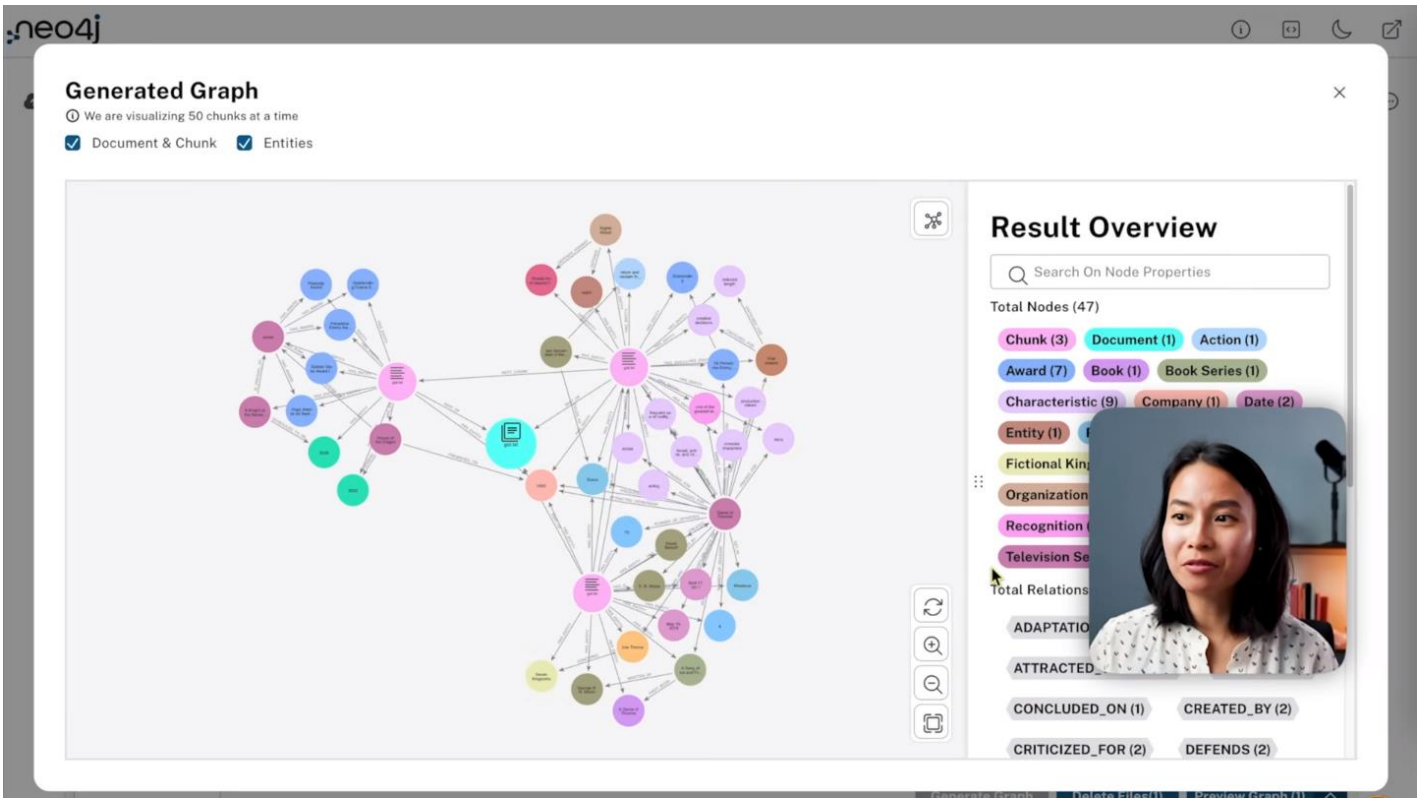

neo4j
Neo4j connection
Not Connected
No Graph Schema configured
Graph Enhancement
Connect to Neo4j
Name
Status
Upload Status
Size (KB)
Source
Type
Model
No data available
Showing 0 of 0 results
LLM Model for Processing & Chat

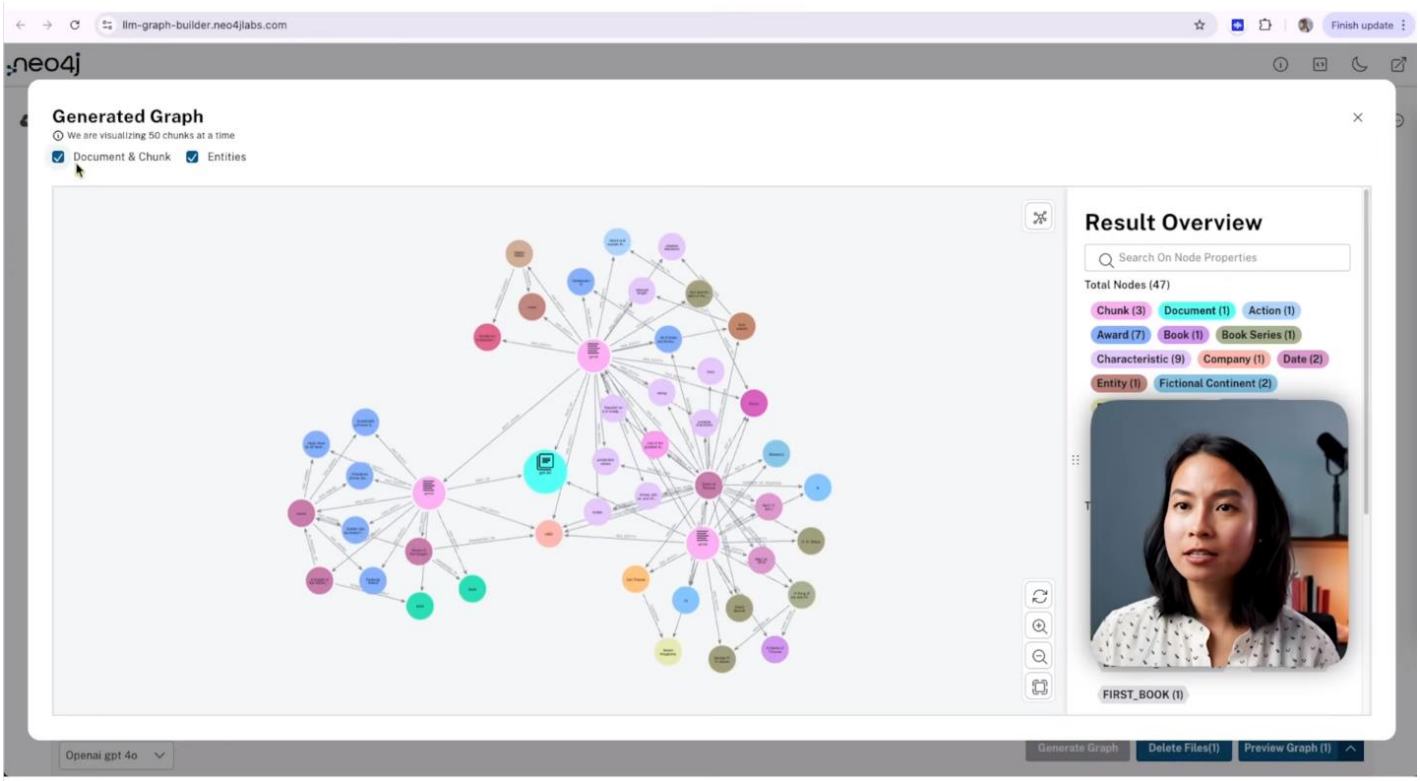

neo4j
Neo4j connection
Not Connected
No Graph Schema configured
Graph Enhancement
Connect to Neo4j
Connect to Neo4j
Don't have a Neo4j instance? Start for free today ↗
Drop your neo4j credentials file here
or browse
Protocol
neo4j+s
URI
70add606.databases.neo4j.io:7687
Database
neo4j
Username
neo4j
Password
password
Connect
Showing 0 of 0 results

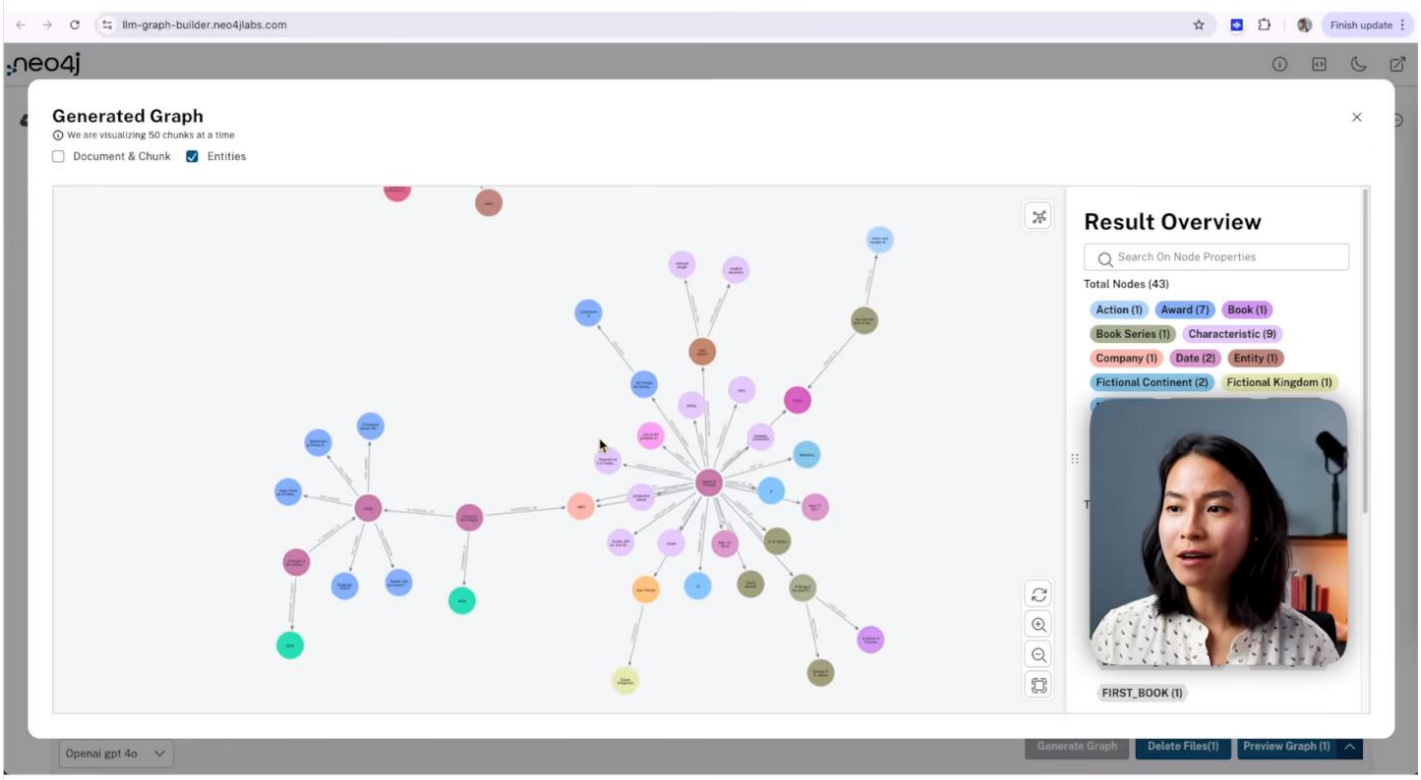

neo4j
Finder
Neo4j connection
neo4j+s://70add606.databases.neo4j.io:7687
No Graph Schema configured
Graph Enhancement
Disconnect
Name
Status
Upload Status
Size (KB)
Source
Type
Model
got.txt
Completed
Uploaded
2.20
local file
TXT
Openai gpt 4o
46

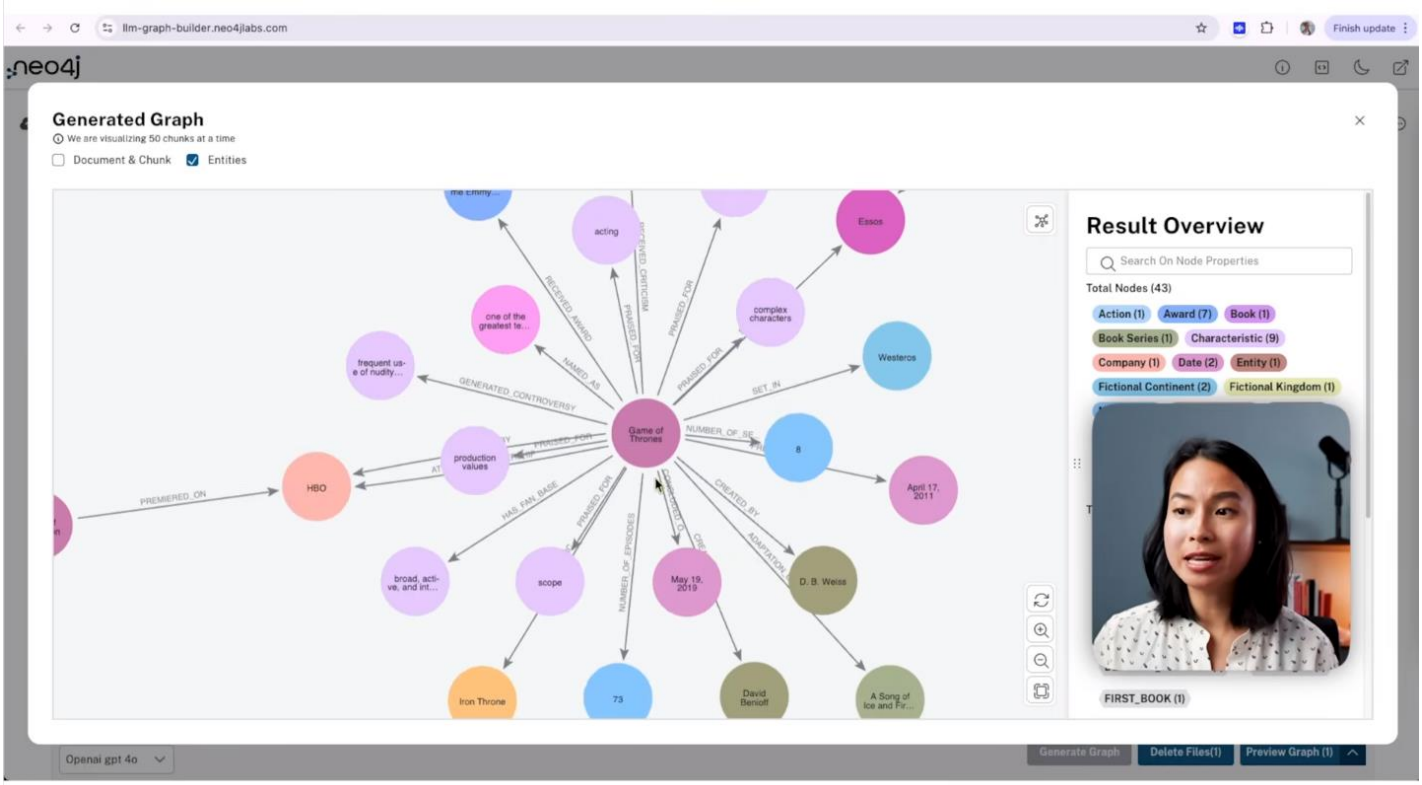

by George R. R. Martin, the first of which is A Game of Thrones. The show premiered on HBO in the United States on April 17, 2011, and concluded on May 19, 2019, with 73 episodes broadcast over eight seasons.
Set on the fictional continents of Westeros and Essos, Game of Thrones has a large ensemble cast and follows several story arcs throughout the course of the show. The first major arc concerns the Iron Throne of the Seven Kingdoms of Westeros through a web of political conflicts among the noble families either vying to claim the throne or fighting for independence from whoever sits on it. The second major arc focuses on the last descendant of the realm's deposed ruling dynasty, who has been exiled to Essos and is plotting to return and reclaim the throne. The third follows the Night's Watch, a military order defending the realm against threats from beyond the Seven Kingdoms' northern border.
Game of Thrones attracted a record viewership on HBO and has a broad, active, and international fan base. Many critics and publications have named the show one of the greatest television series of all time. Critics have praised the series for its acting, complex characters, story, scope, and production values, although its frequent use of nudity and violence (including sexual violence) generated controversy. The final season received significant criticism for its reduced length and creative decisions, with many considering it a disappointing conclusion. The series received 59 Primetime Emmy Awards, the most by a drama series, including Outstanding Drama Series in 2015, 2016, 2018 and 2019. Its other awards and nominations include three Hugo Awards for Best Dramatic Presentation, a Peabody Award, and five nominations for the Golden Globe Award for Best Television Series – Drama.
A prequel series, House of the Dragon, premiered on HBO in 2022. A second prequel currently in production, A Knight of the Seven Kingdoms, is scheduled to debut in 2025.
Graph Enhancement
Disconnect
Source
Type
Model
local file
TXT
Openai gpt 4o
Large files may be partially
Downloads
Back/Forward
View
Group
Share
Add Tags
Previous 7 Days
AirDrop
Recents
Applications
Downloads
Movies
CS
vlogging
screenshots
Research
Python A-Z for Data...
python_projects
Python course
got.txt
Neo4j-70add606-Created-2025-05-13.txt
Final Project.zip
data_conversion_output (1).json
data_conversion_output.json
Final Project
Final Project - Project Proposal.zip
Final Project - Project Proposal.canvas
converted_data_hierarchy.json
converted_data_ellipse.json
converted_data.json
test.canvas
master.m3u8
Document
JSON
May
LLM Model for Processing & Chat

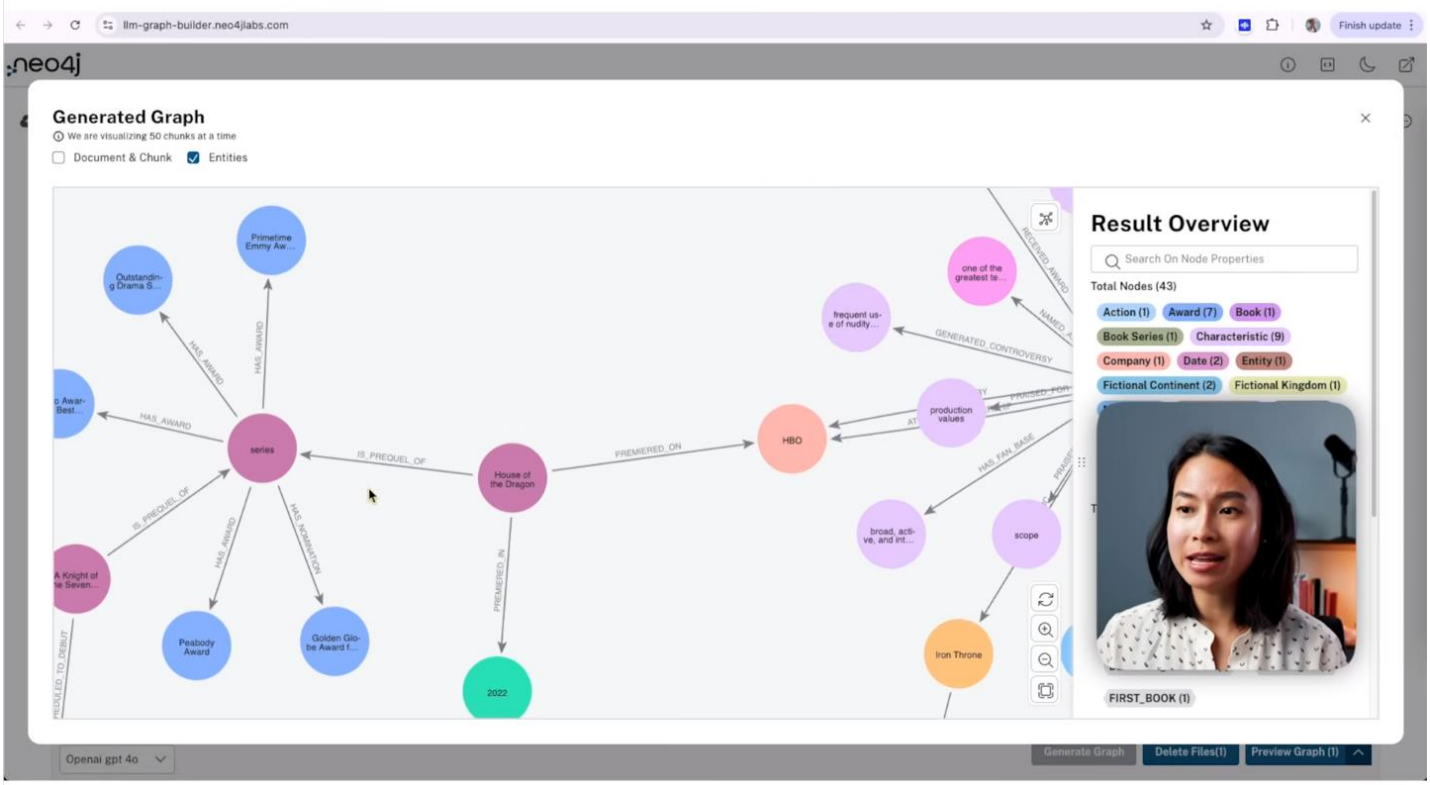

Neo4j connection
neo4j+s://70add606.databases.neo4j.io:7687
No Graph Schema configured
Graph Enhancement
Disconnect
Name
Status
Upload Status
Size (KB)
Source
Type
Model
got.txt
Completed
Uploaded
2.20
local file
TXT
Openai gpt 4o
Large files may be partially processed up to 10K characters due to resource limit.
LLM Model for Processing & Chat

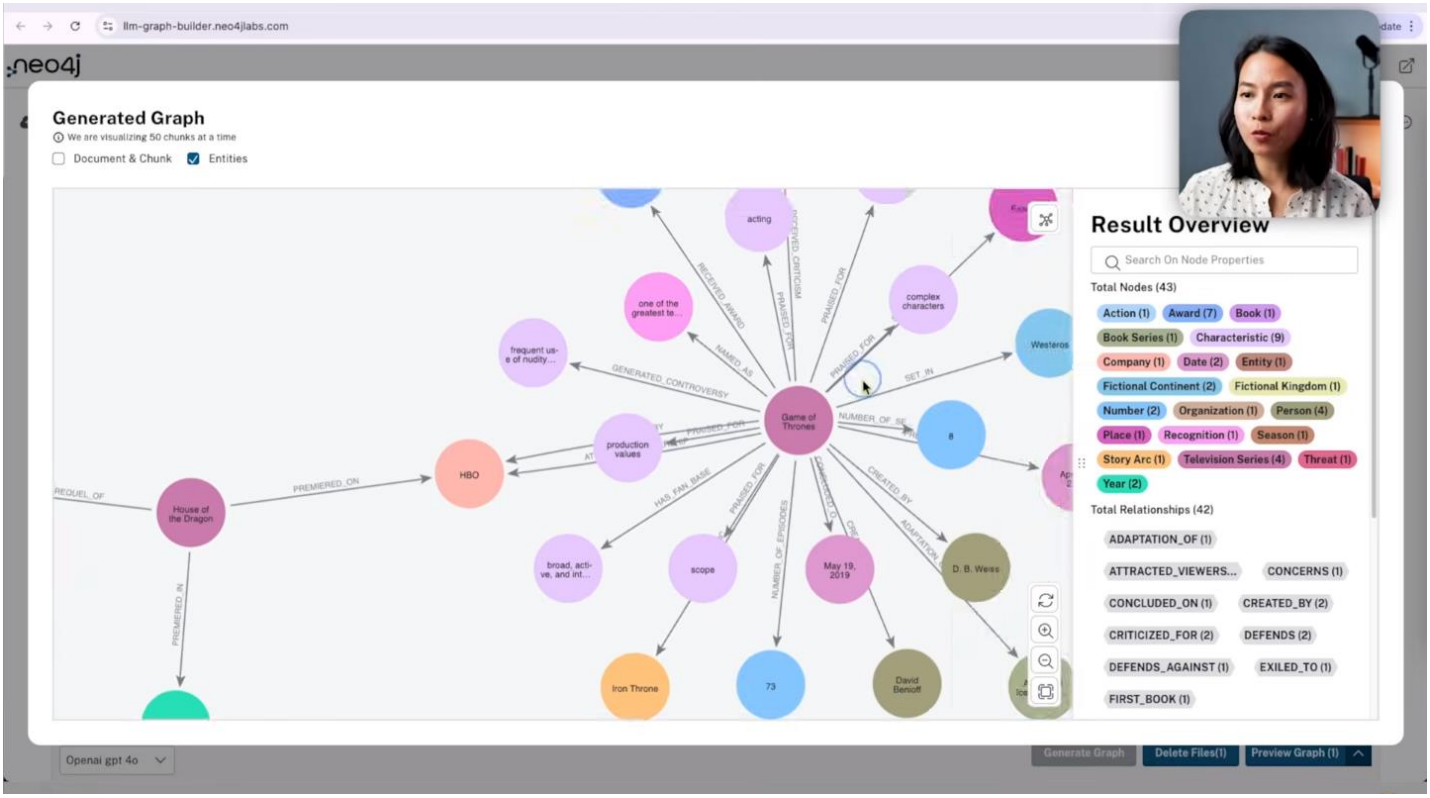

neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (47)
Chunk (3)
Document (1)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Kin
Organization
Recognition
Television Se
Total Relations
ADAPTATIO
ATTRACTED_
CONCLUDED_ON (1)
CREATED_BY (2)
CRITICIZED_FOR (2)
DEFENDS (2)

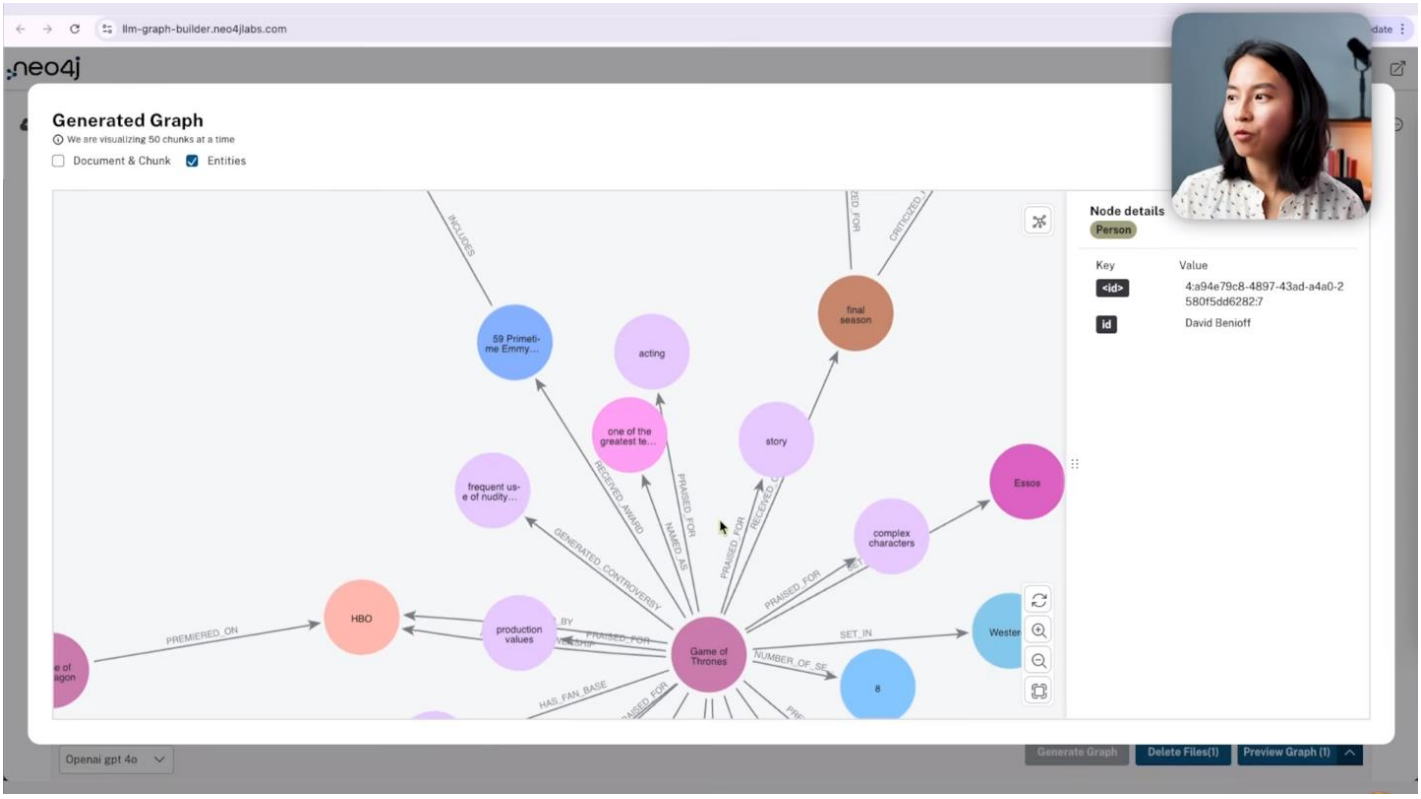

llm-graph-builder.neo4jlabs.com
Finish update
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (47)
Chunk (3)
Document (1)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
FIRST_BOOK (1)
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

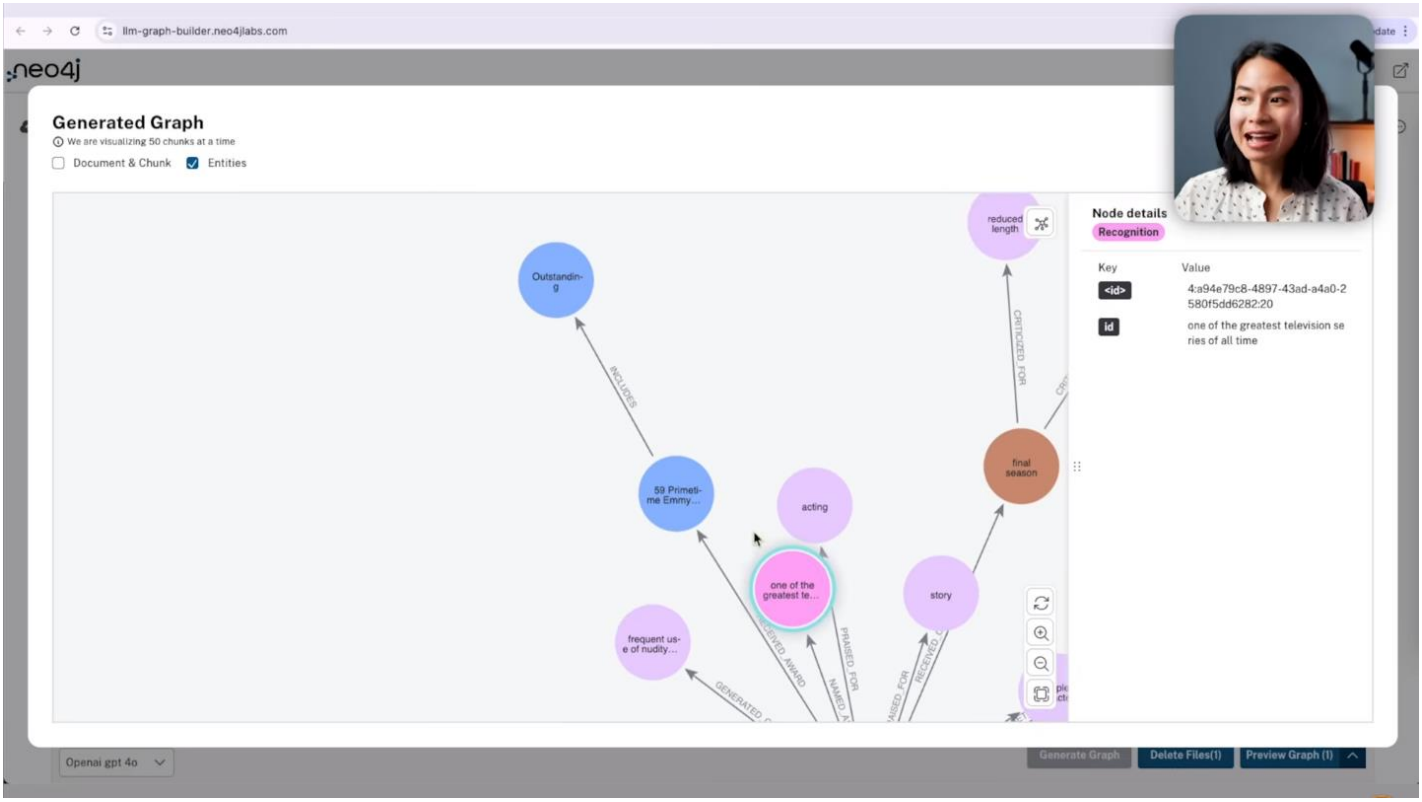

llm-graph-builder.neo4jlabs.com
Finish update
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
FIRST_BOOK (1)
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

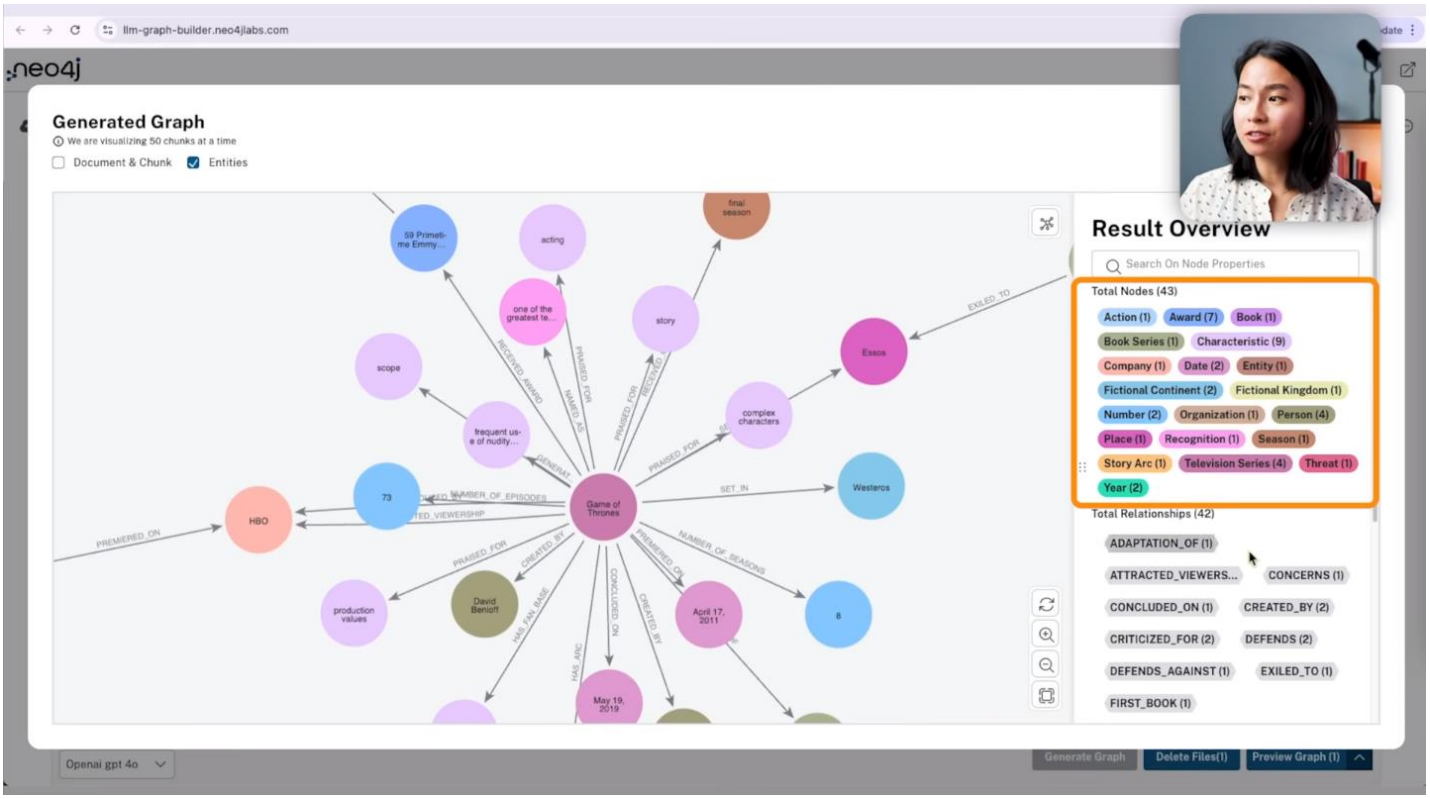

llm-graph-builder.neo4jlabs.com
Finish update
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
acting
Essos
one of the greatest te...
complex characters
frequent us- e of nudity...
Westeros
Game of Thrones
production values
HBO
8
April 17, 2011
broad, acti- ve, and int...
scope
May 19, 2019
D. B. Weiss
Iron Throne
73
David Benioff
A Song of Ice and Fir...
RECEIVED_AWARD
NAMED_AS
PRAISED_FOR
GENERATED_CONTROVERSY
SET_IN
PREMIERED_ON
HAS_FAN_BASE
CREATED_BY
ADAPTATION
NUMBER_OF_EPISODES
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
FIRST_BOOK (1)
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

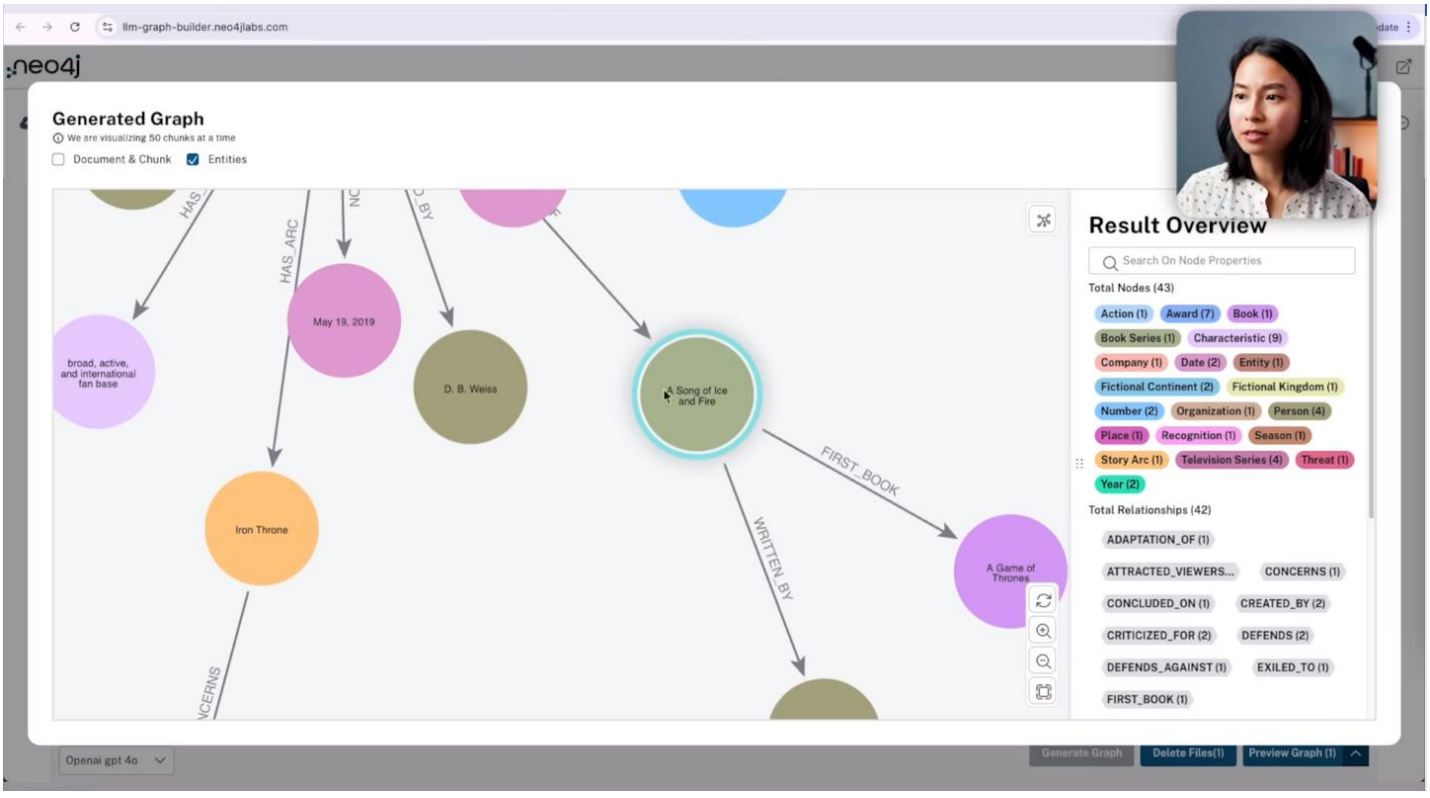

llm-graph-builder.neo4jlabs.com
Finish update
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
FIRST_BOOK (1)
series
House of the Dragon
HBO
Primetime Emmy Aw...
Outstandin- g Drama S...
Peabody Award
Golden Glo- be Award f...
2022
Iron Throne
scope
broad, acti- ve, and int...
production values
frequent us- e of nudity...
one of the greatest te...
HAS_AWARD
IS_PREQUEL_OF
PREMIERED_ON
PREMIERED_IN
HAS_NOMINATION
GENERATED_CONTROVERSY
HAS_FAN_BASE
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

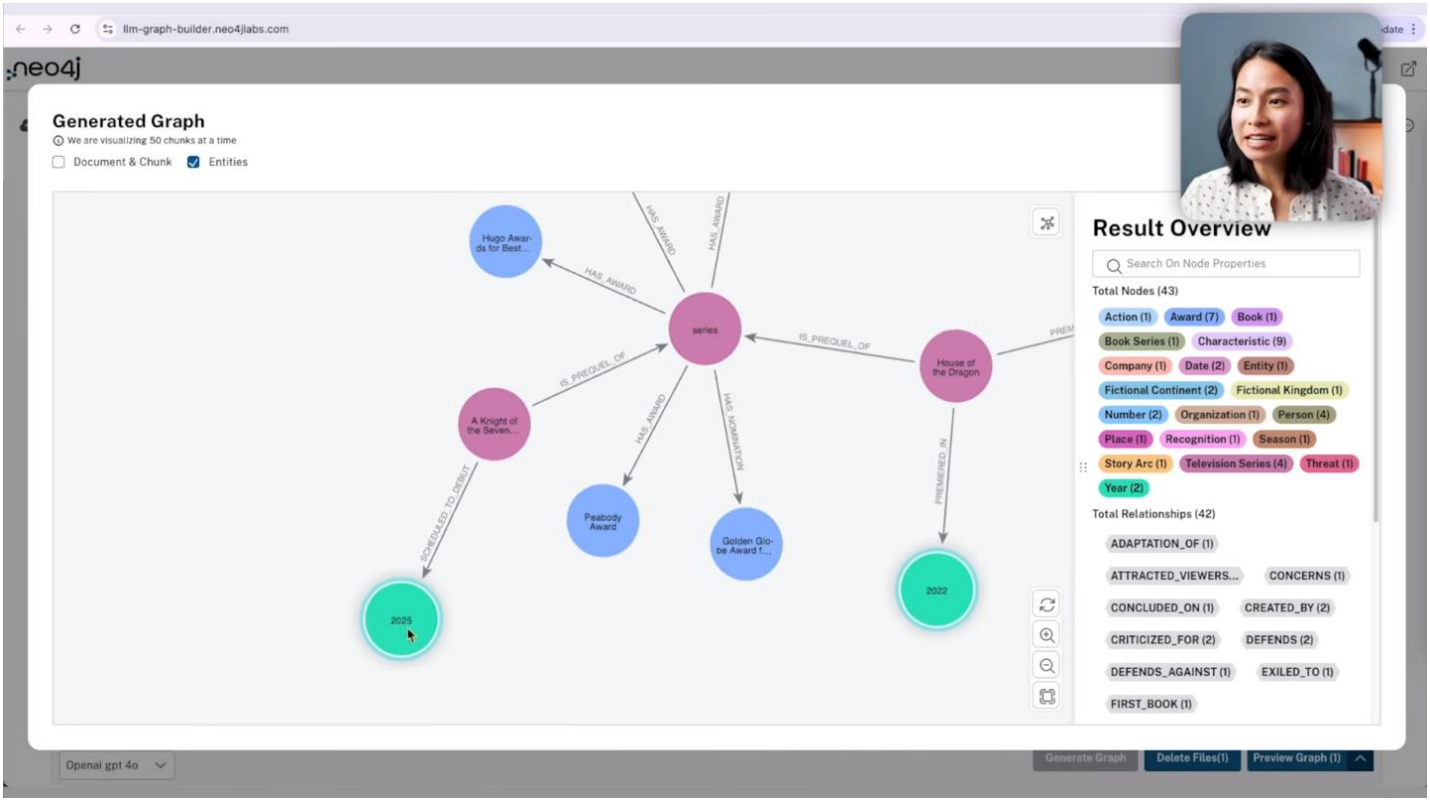

llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
Number (2)
Organization (1)
Person (4)
Place (1)
Recognition (1)
Season (1)
Story Arc (1)
Television Series (4)
Threat (1)
Year (2)
Total Relationships (42)
ADAPTATION_OF (1)
ATTRACTED_VIEWERS...
CONCERNS (1)
CONCLUDED_ON (1)
CREATED_BY (2)
CRITICIZED_FOR (2)
DEFENDS (2)
DEFENDS_AGAINST (1)
EXILED_TO (1)
FIRST_BOOK (1)
Game of Thrones
House of the Dragon
HBO
acting
complex characters
Westeros
8
D. B. Weiss
David Benioff
73
May 19, 2019
RECEIVED_AWARD
PRAISED_FOR
NAMED_AS
SET_IN
NUMBER_OF_EPISODES
CREATED_BY
ADAPTATION_OF
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

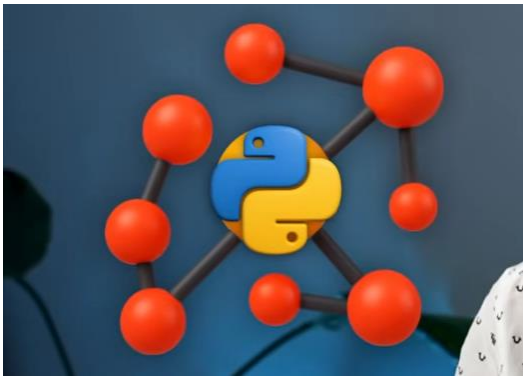

llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Node details
Person
Key
Value
<id>
4:a94e79c8-4897-43ad-a4a0-2580f5dd6282:7
id
David Benioff
59 Primeti-me Emmy...
acting
final season
one of the greatest te...
story
frequent us-e of nudity...
Essos
complex characters
HBO
production values
Game of Thrones
8
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

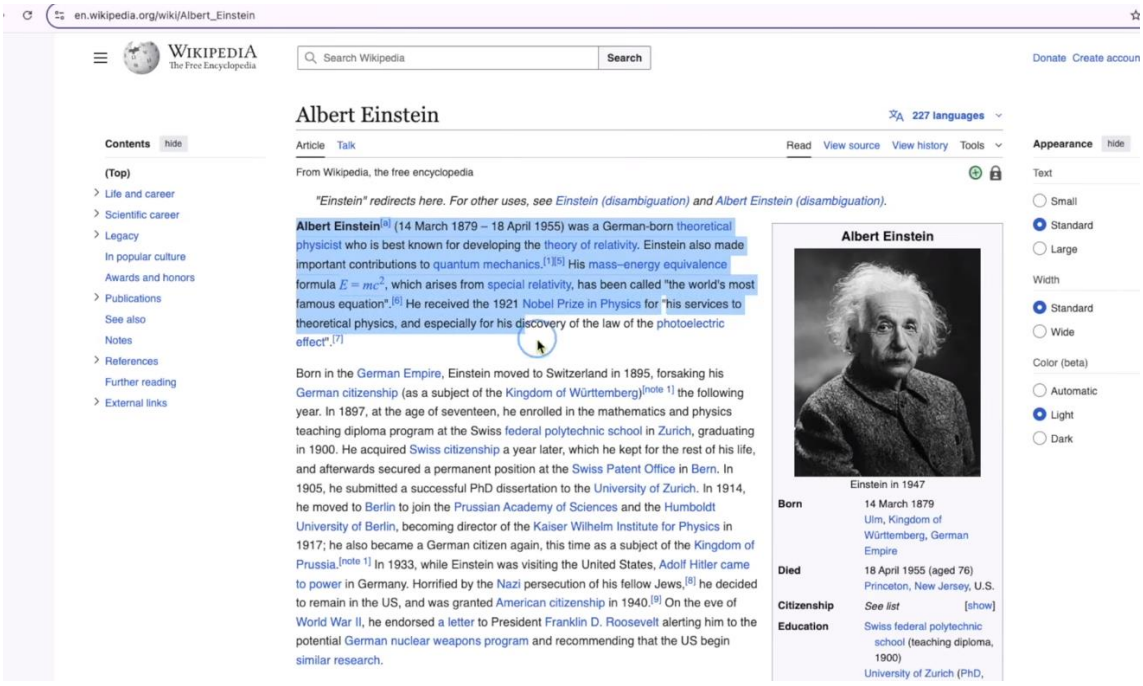

llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Node details
Recognition
Key
Value
<id>
4:a94e79c8-4897-43ad-a4a0-2580f5dd6282:20
id
one of the greatest television series of all time
reduced length
Outstandin-g
59 Primeti-me Emmy...
acting
final season
one of the greatest te...
story
frequent us-e of nudity...
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)


llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
Number (2)
Organization (1)
Person (4)
Place (1)
Recognition (1)
Season (1)
Story Arc (1)
Television Series (4)
Threat (1)
Year (2)
Total Relationships (42)
ADAPTATION_OF (1)
ATTRACTED_VIEWERS...
CONCERNS (1)
CONCLUDED_ON (1)
CREATED_BY (2)
CRITICIZED_FOR (2)
DEFENDS (2)
DEFENDS_AGAINST (1)
EXILED_TO (1)
FIRST_BOOK (1)
Game of Thrones
HBO
Westeros
Essos
David Benioff
production values
April 17, 2011
May 19, 2019
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)


llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Result Overview
Search On Node Properties
Total Nodes (43)
Total Relationships (42)
May 19, 2019
D. B. Weiss
A Song of Ice and Fire
Iron Throne
A Game of Thrones
broad, active, and international fan base
HAS_ARC
FIRST_BOOK
WRITTEN_BY
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)


llm-graph-builder.neo4jlabs.com
neo4j
Generated Graph
We are visualizing 50 chunks at a time
Document & Chunk
Entities
Hugo Awards for Best...
HAS_AWARD
series
IS_PREQUEL_OF
House of the Dragon
A Knight of the Seven...
Peabody Award
Golden Globe Award f...
SCHEDULED_TO_DEBUT
HAS_NOMINATION
PREMIERED_IN
2025
2022
Result Overview
Search On Node Properties
Total Nodes (43)
Action (1)
Award (7)
Book (1)
Book Series (1)
Characteristic (9)
Company (1)
Date (2)
Entity (1)
Fictional Continent (2)
Fictional Kingdom (1)
Number (2)
Organization (1)
Person (4)
Place (1)
Recognition (1)
Season (1)
Story Arc (1)
Television Series (4)
Threat (1)
Year (2)
Total Relationships (42)
ADAPTATION_OF (1)
ATTRACTED_VIEWERS...
CONCERNS (1)
CONCLUDED_ON (1)
CREATED_BY (2)
CRITICIZED_FOR (2)
DEFENDS (2)
DEFENDS_AGAINST (1)
EXILED_TO (1)
FIRST_BOOK (1)
Openai gpt 4o
Generate Graph
Delete Files(1)
Preview Graph (1)

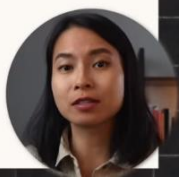


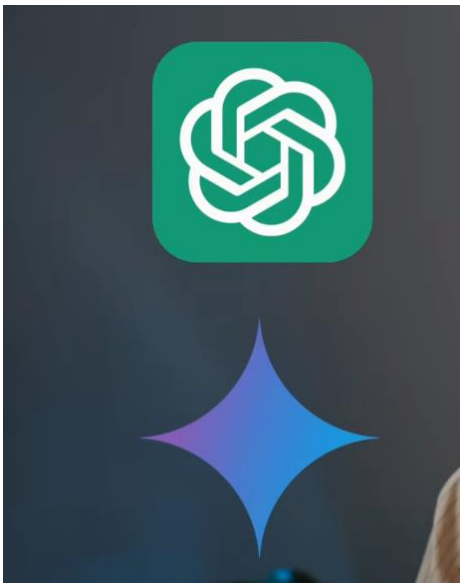

en.wikipedia.org/wiki/Albert_Einstein
WIKIPEDIA
The Free Encyclopedia
Search Wikipedia
Search
Donate
Create account
Albert Einstein
227 languages
Article
Talk
Read
View source
View history
Tools
Contents
hide
(Top)
Life and career
Scientific career
Legacy
In popular culture
Awards and honors
Publications
See also
Notes
References
Further reading
External links
From Wikipedia, the free encyclopedia
"Einstein" redirects here. For other uses, see Einstein (disambiguation) and Albert Einstein (disambiguation).
Albert Einstein[a] (14 March 1879 – 18 April 1955) was a German-born theoretical physicist who is best known for developing the theory of relativity. Einstein also made important contributions to quantum mechanics.[1][5] His mass–energy equivalence formula $E = mc^2$, which arises from special relativity, has been called "the world's most famous equation".[6] He received the 1921 Nobel Prize in Physics for "his services to theoretical physics, and especially for his discovery of the law of the photoelectric effect".[7]
Born in the German Empire, Einstein moved to Switzerland in 1895, forsaking his German citizenship (as a subject of the Kingdom of Württemberg)[note 1] the following year. In 1897, at the age of seventeen, he enrolled in the mathematics and physics teaching diploma program at the Swiss federal polytechnic school in Zurich, graduating in 1900. He acquired Swiss citizenship a year later, which he kept for the rest of his life, and afterwards secured a permanent position at the Swiss Patent Office in Bern. In 1905, he submitted a successful PhD dissertation to the University of Zurich. In 1914, he moved to Berlin to join the Prussian Academy of Sciences and the Humboldt University of Berlin, becoming director of the Kaiser Wilhelm Institute for Physics in 1917; he also became a German citizen again, this time as a subject of the Kingdom of Prussia.[note 1] In 1933, while Einstein was visiting the United States, Adolf Hitler came to power in Germany. Horrified by the Nazi persecution of his fellow Jews,[8] he decided to remain in the US, and was granted American citizenship in 1940.[9] On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential German nuclear weapons program and recommending that the US begin similar research.
Albert Einstein
Einstein in 1947
Born
14 March 1879
Ulm, Kingdom of Württemberg, German Empire
Died
18 April 1955 (aged 76)
Princeton, New Jersey, U.S.
Citizenship
See list
[show]
Education
Swiss federal polytechnic school (teaching diploma, 1900)
University of Zurich (PhD,
Appearance
hide
Text
Small
Standard
Large
Width
Standard
Wide
Color (beta)
Automatic
Light
Dark

BUILDING KNOWLEDGE GRAPH (from text)

Few-shot Prompting

```
"""You are a top-tier algorithm designed for extracting
information in structured formats to build a knowledge
graph. Your task is to identify the entities and relations
requested with the user prompt from a given text. You must
generate the output in a JSON format containing a list with
JSON objects. Each object should have the keys: "head",
"head_type", "relation", "tail", and "tail_type". Here is
one example:
---
Give the text: "Adam is a software engineer in Microsoft
since 2009"
You can extract a relationship in the following format:
{
  "head": "Adam",
  "head_type": "Person",
  "relation": "WORKS_FOR",
  "tail": "Microsoft",
  "tail_type": "Company"
}
```





BUILDING KNOWLEDGE GRAPH (from text)

- Few-shot Prompting
- Using tools (structured output)

```python
class Node(BaseNode):
    id: str
    label: str
    properties: Optional[List[Property]]

class Relationship(BaseRelationship):
    source: Node
    target: Node
    type: str
    properties: Optional[List[Property]]

class KnowledgeGraph(BaseModel):
    """Generate a knowledge graph with entities and relationships."""
    nodes: List[Node] = Field(
        ..., description="List of nodes in the knowledge graph"
    )
    rels: List[Relationship] = Field(
        ..., description="List of relationships in the knowledge graph"
    )
```

API Reference Legacy reference — Search ⌘ + K — Docs

- fallacy_removal
- generative_agents
- graph_transformers
  - DiffbotGraphTransformer
  - NodesList
  - SimplifiedSchema
  - TypeOption
  - GlinerGraphTransformer
  - LLMGraphTransformer
  - UnstructuredRelation
  - RelikGraphTransformer
  - format_property_key
  - create_simple_model
  - create_unstructured_prompt
  - format_property_key
  - map_to_base_node
  - map_to_base_relationship
  - optional_enum_field
- llm_bash
- llm_symbolic_math
- llms
- open_clip
- pal_chain
- plan_and_execute
- prompt_injection_identifier
- recommenders
- retrievers
- rl_chain

LangChain Python API Reference > ··· > graph_transformers > LLMGraphTransformer

LLMGraphTransformer

class langchain_experimental.graph_transformers.llm.LLMGraphTransformer(llm: BaseLanguageModel, allowed_nodes: List[str] = [], allowed_relationships: List[str] = [], prompt: ChatPromptTemplate | None = None, strict_mode: bool = True, node_properties: bool | List[str] = False, relationship_properties: bool | List[str] = False, ignore_tool_usage: bool = False) [source]

Transform documents into graph-based documents using a LLM.

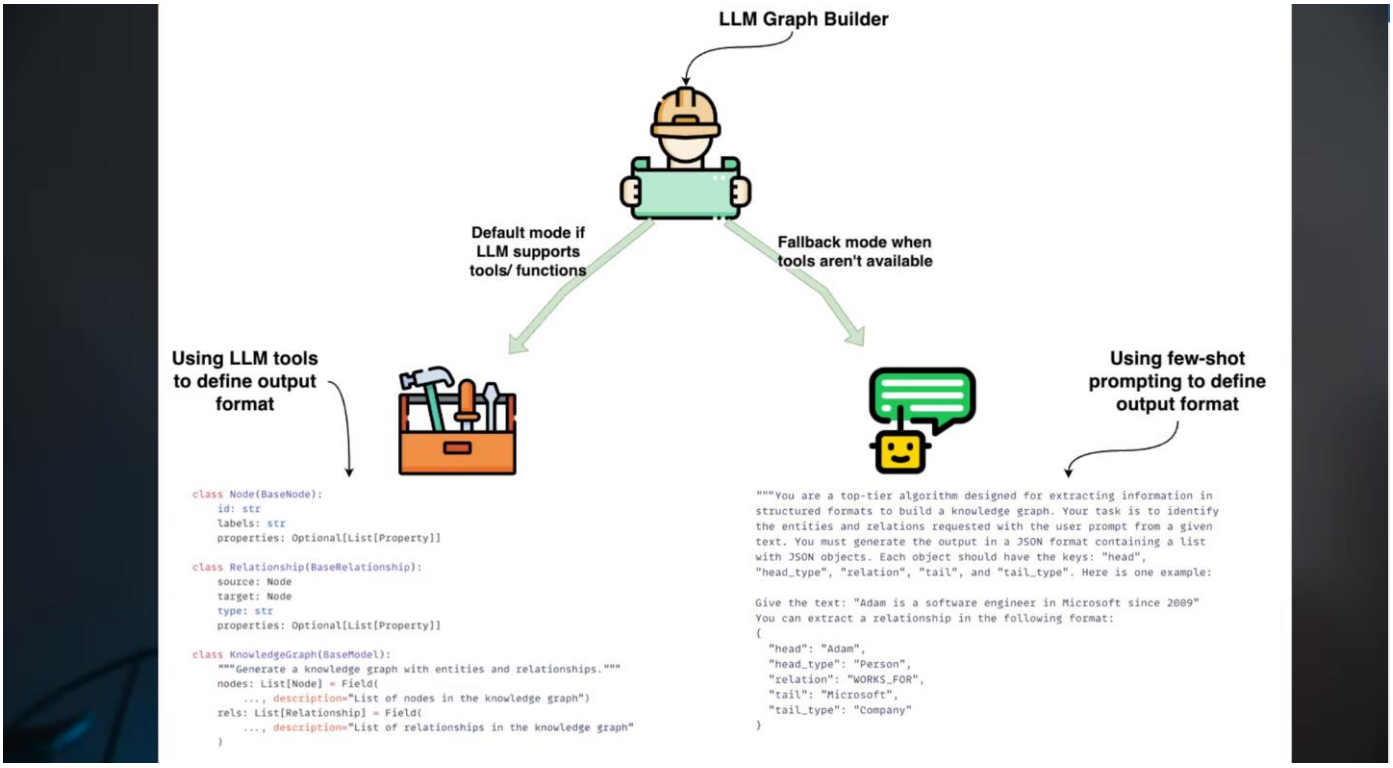
It allows specifying constraints on the types of nodes and relationships to include in the output graph. The class supports extracting properties for both nodes and relationships.

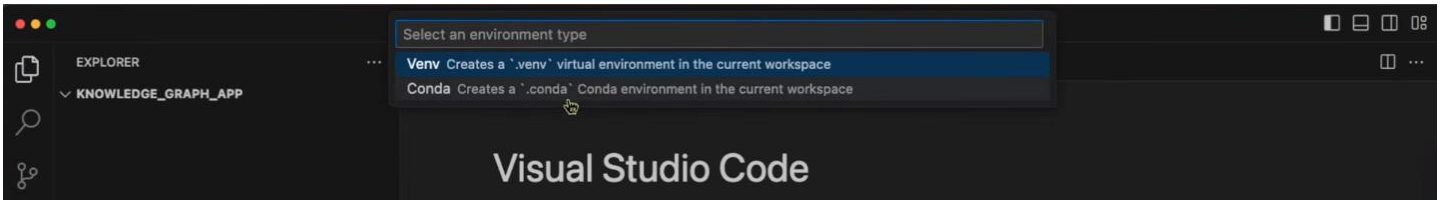
Parameters:

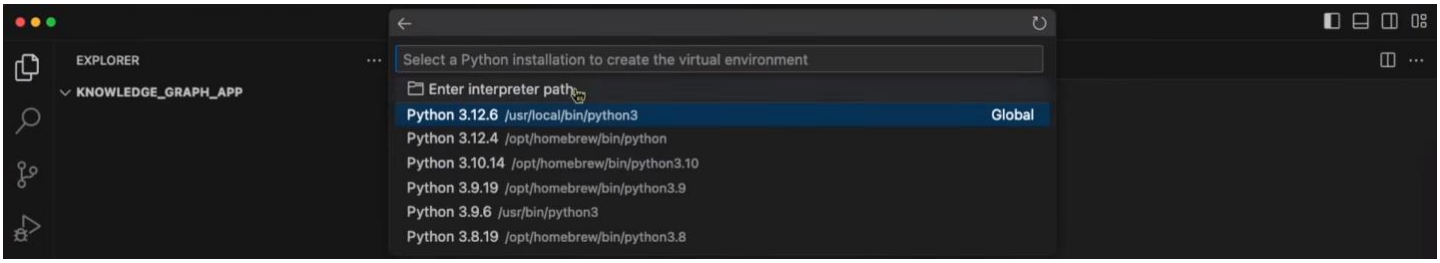
- **llm** (BaseLanguageModel) – An instance of a language model supporting structured output.
- **allowed_nodes** (List[str], optional) – Specifies which node types are allowed in the graph. Defaults to an empty list, allowing all node types.
- **allowed_relationships** (List[str], optional) – Specifies which relationship types are allowed in the graph. Defaults to an empty list, allowing all relationship types.
- **prompt** (Optional[ChatPromptTemplate], optional) – The prompt to pass to the LLM with additional instructions.
- **strict_mode** (bool, optional) – Determines whether the transformer should apply filtering to strictly adhere to allowed_nodes and allowed_relationships. Defaults to True.
- **node_properties** (Union[bool, List[str]]) – If True, the LLM can extract any node properties from text. Alternatively, a list of valid properties can be provided for the LLM to extract, restricting extraction to those specified.
- **relationship_properties** (Union[bool, List[str]]) – If True, the LLM can extract any relationship properties from text. Alternatively, a list of valid properties can be provided for the LLM to extract, restricting extraction to those specified.

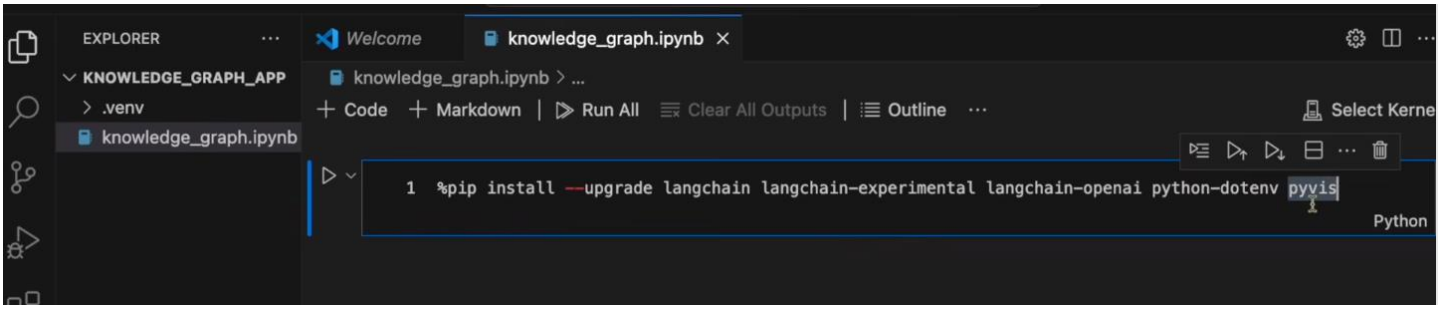
On this page

- LLMGraphTransformer
  - __init__()
  - aconvert_to_graph_documents()
  - aprocess_response()
  - convert_to_graph_documents()
  - process_response()

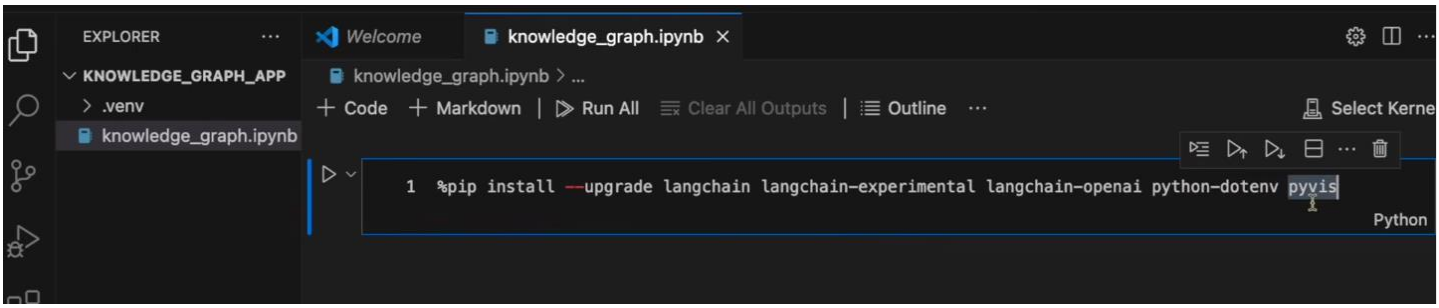


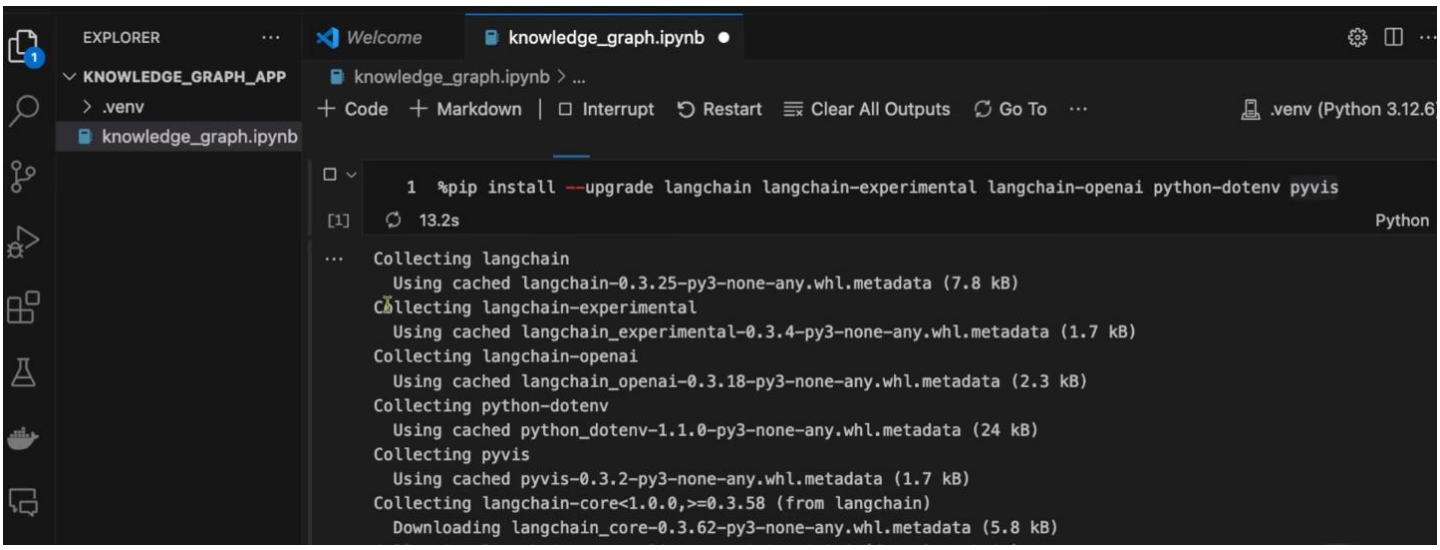


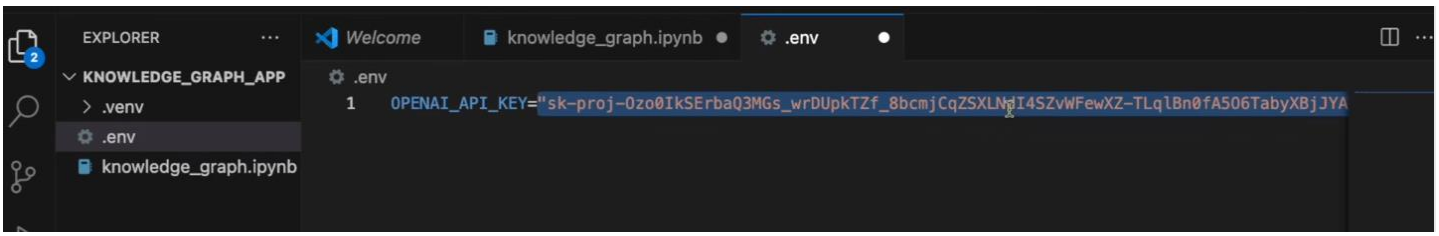


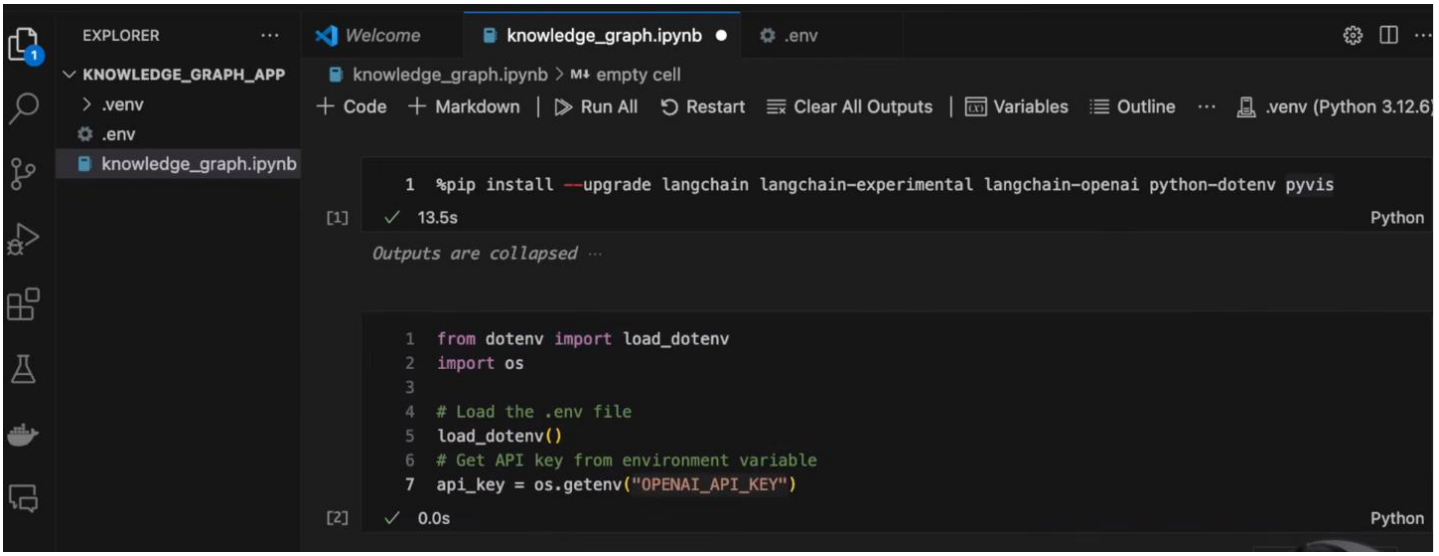


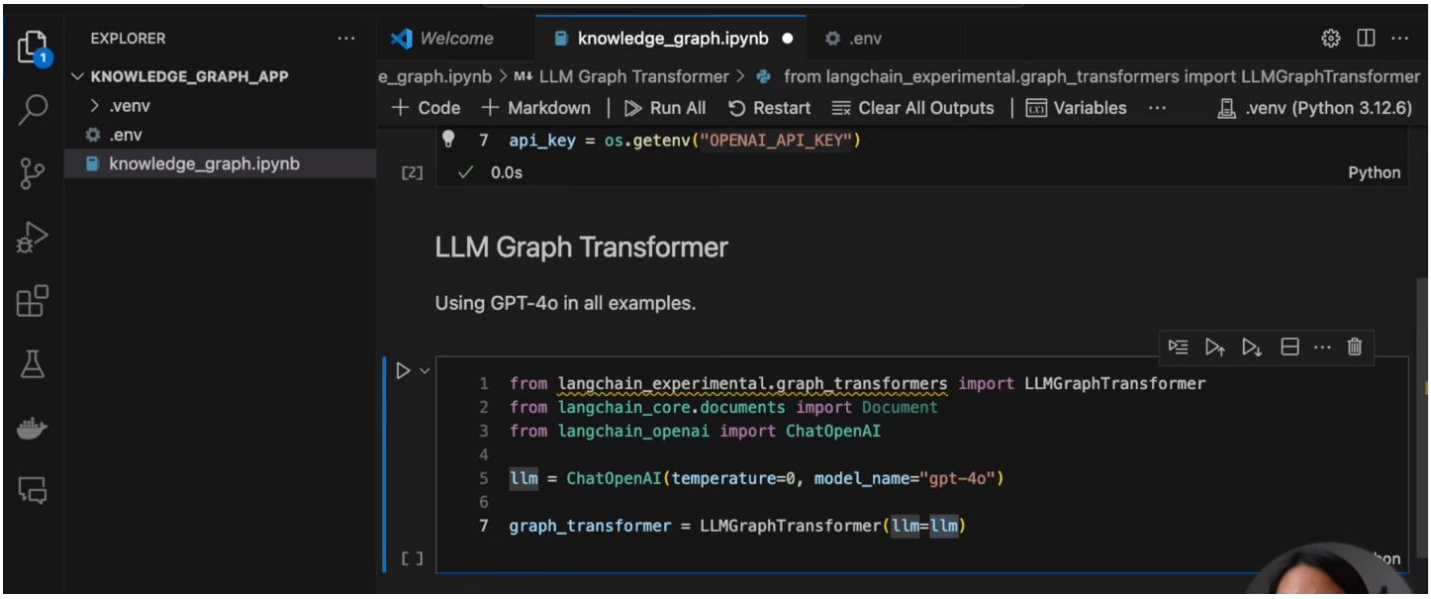
Let us code in the **Notebook** before we build out the app in **Streamlit**, **pyvis** is a network graph visualization package in **Python**.

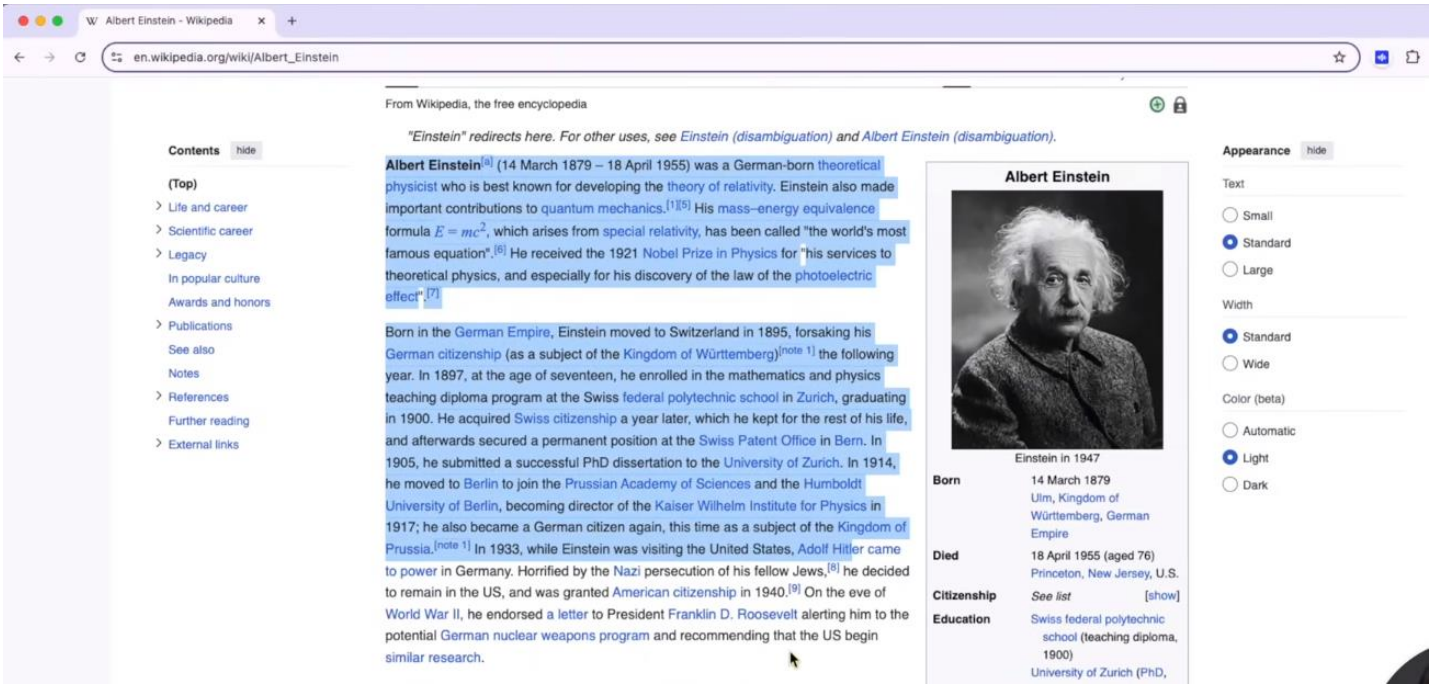

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
knowledge_graph.ipynb > ...
Code
Markdown
Run All
Clear All Outputs
Outline
Select Kerne
1 %pip install --upgrade langchain langchain-experimental langchain-openai python-dotenv pyvis
Python

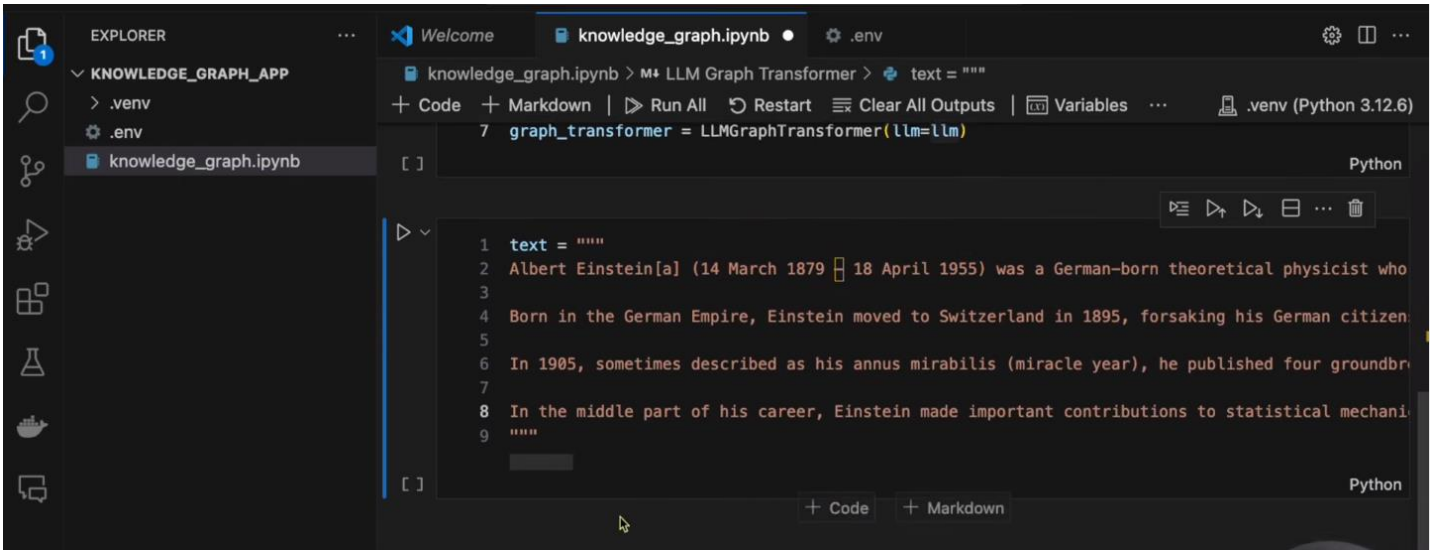

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
knowledge_graph.ipynb > ...
Code
Markdown
Interrupt
Restart
Clear All Outputs
Go To
.venv (Python 3.12.6)
1 %pip install --upgrade langchain langchain-experimental langchain-openai python-dotenv pyvis
[1] 13.2s
Python
Collecting langchain
Using cached langchain-0.3.25-py3-none-any.whl.metadata (7.8 kB)
Collecting langchain-experimental
Using cached langchain_experimental-0.3.4-py3-none-any.whl.metadata (1.7 kB)
Collecting langchain-openai
Using cached langchain_openai-0.3.18-py3-none-any.whl.metadata (2.3 kB)
Collecting python-dotenv
Using cached python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Collecting pyvis
Using cached pyvis-0.3.2-py3-none-any.whl.metadata (1.7 kB)
Collecting langchain-core<1.0.0,>=0.3.58 (from langchain)
Downloading langchain_core-0.3.62-py3-none-any.whl.metadata (5.8 kB)

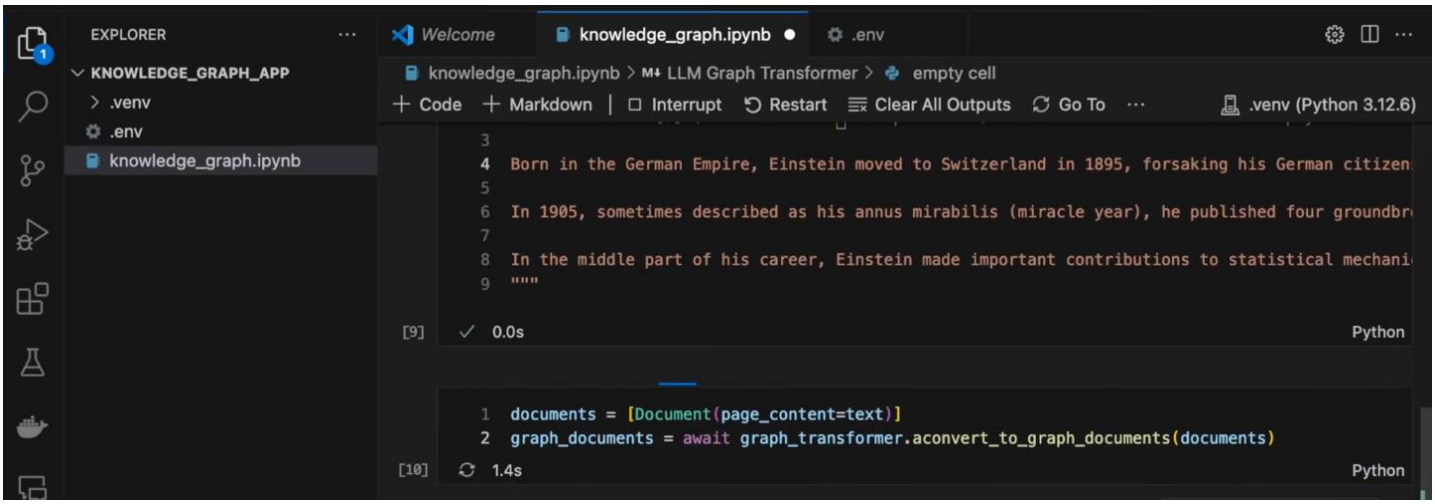

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
.env
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
.env
1 OPENAI_API_KEY="sk-proj-Ozo0IkSErbaQ3MGs_wrDUpkTZf_8bcmjCqZSXLNqI4SZvWFewXZ-TLqlBn0fA5O6TabyXBjJYA


EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
.env
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > empty cell
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
.venv (Python 3.12.6)
1 %pip install --upgrade langchain langchain-experimental langchain-openai python-dotenv pyvis
[1] 13.5s
Python
Outputs are collapsed ...
1 from dotenv import load_dotenv
2 import os
3
4 # Load the .env file
5 load_dotenv()
6 # Get API key from environment variable
7 api_key = os.getenv("OPENAI_API_KEY")
[2] 0.0s
Python

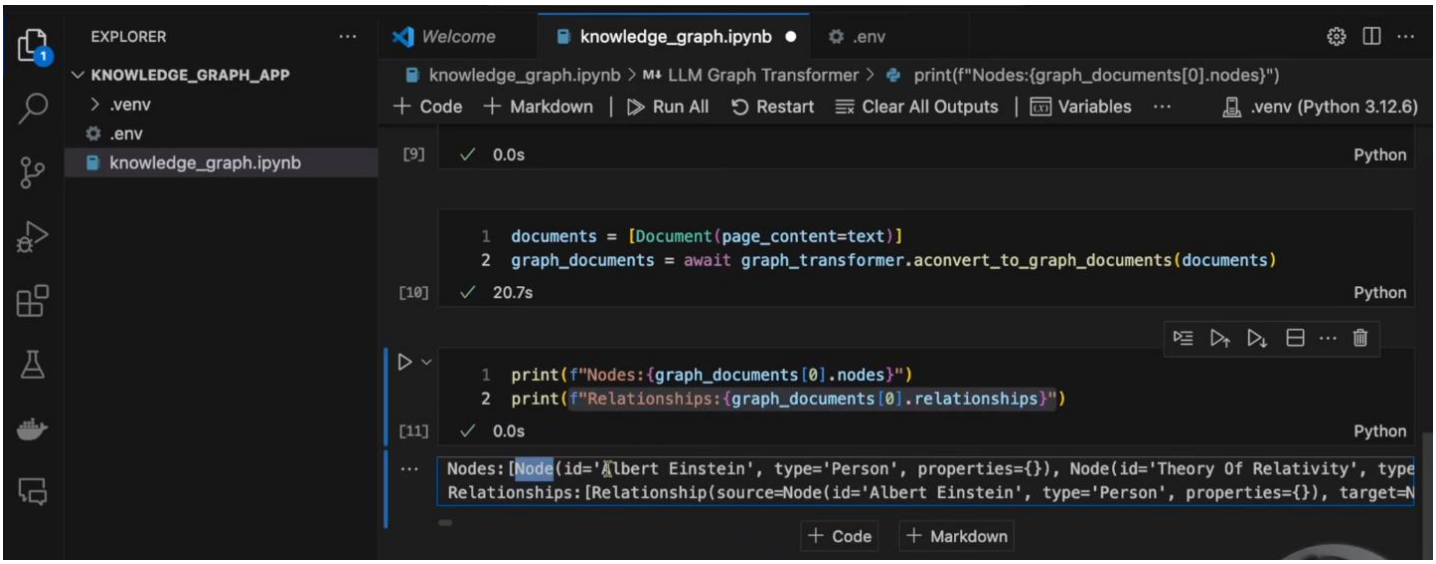

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
.env
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
e_graph.ipynb > LLM Graph Transformer > from langchain_experimental.graph_transformers import LLMGraphTransformer
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
7 api_key = os.getenv("OPENAI_API_KEY")
[2] 0.0s
Python
LLM Graph Transformer
Using GPT-4o in all examples.
1 from langchain_experimental.graph_transformers import LLMGraphTransformer
2 from langchain_core.documents import Document
3 from langchain_openai import ChatOpenAI
4
5 llm = ChatOpenAI(temperature=0, model_name="gpt-4o")
6
7 graph_transformer = LLMGraphTransformer(llm=llm)
[ ]


Albert Einstein - Wikipedia
en.wikipedia.org/wiki/Albert_Einstein
From Wikipedia, the free encyclopedia
"Einstein" redirects here. For other uses, see Einstein (disambiguation) and Albert Einstein (disambiguation).
Contents hide
(Top)
Life and career
Scientific career
Legacy
In popular culture
Awards and honors
Publications
See also
Notes
References
Further reading
External links
Albert Einstein[a] (14 March 1879 – 18 April 1955) was a German-born theoretical physicist who is best known for developing the theory of relativity. Einstein also made important contributions to quantum mechanics.[1][5] His mass–energy equivalence formula $E = mc^2$, which arises from special relativity, has been called "the world's most famous equation".[6] He received the 1921 Nobel Prize in Physics for "his services to theoretical physics, and especially for his discovery of the law of the photoelectric effect".[7]
Born in the German Empire, Einstein moved to Switzerland in 1895, forsaking his German citizenship (as a subject of the Kingdom of Württemberg)[note 1] the following year. In 1897, at the age of seventeen, he enrolled in the mathematics and physics teaching diploma program at the Swiss federal polytechnic school in Zurich, graduating in 1900. He acquired Swiss citizenship a year later, which he kept for the rest of his life, and afterwards secured a permanent position at the Swiss Patent Office in Bern. In 1905, he submitted a successful PhD dissertation to the University of Zurich. In 1914, he moved to Berlin to join the Prussian Academy of Sciences and the Humboldt University of Berlin, becoming director of the Kaiser Wilhelm Institute for Physics in 1917; he also became a German citizen again, this time as a subject of the Kingdom of Prussia.[note 1] In 1933, while Einstein was visiting the United States, Adolf Hitler came to power in Germany. Horrified by the Nazi persecution of his fellow Jews,[8] he decided to remain in the US, and was granted American citizenship in 1940.[9] On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential German nuclear weapons program and recommending that the US begin similar research.
Albert Einstein
Einstein in 1947
Born 14 March 1879 Ulm, Kingdom of Württemberg, German Empire
Died 18 April 1955 (aged 76) Princeton, New Jersey, U.S.
Citizenship See list [show]
Education Swiss federal polytechnic school (teaching diploma, 1900) University of Zurich (PhD,
Appearance hide
Text
Small
Standard
Large
Width
Standard
Wide
Color (beta)
Automatic
Light
Dark

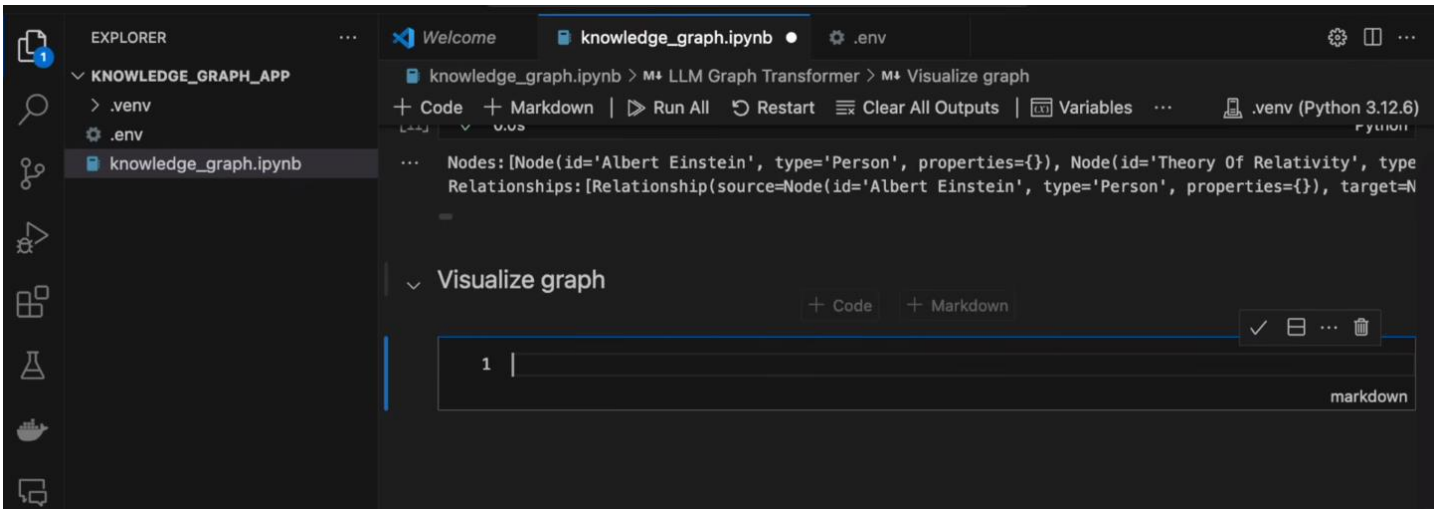

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
.env
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > text = """
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
7 graph_transformer = LLMGraphTransformer(llm=llm)
[ ]
Python
1 text = """
2 Albert Einstein[a] (14 March 1879 – 18 April 1955) was a German-born theoretical physicist who
3
4 Born in the German Empire, Einstein moved to Switzerland in 1895, forsaking his German citizen
5
6 In 1905, sometimes described as his annus mirabilis (miracle year), he published four groundbr
7
8 In the middle part of his career, Einstein made important contributions to statistical mechani
9 """
[ ]
Python
Code
Markdown

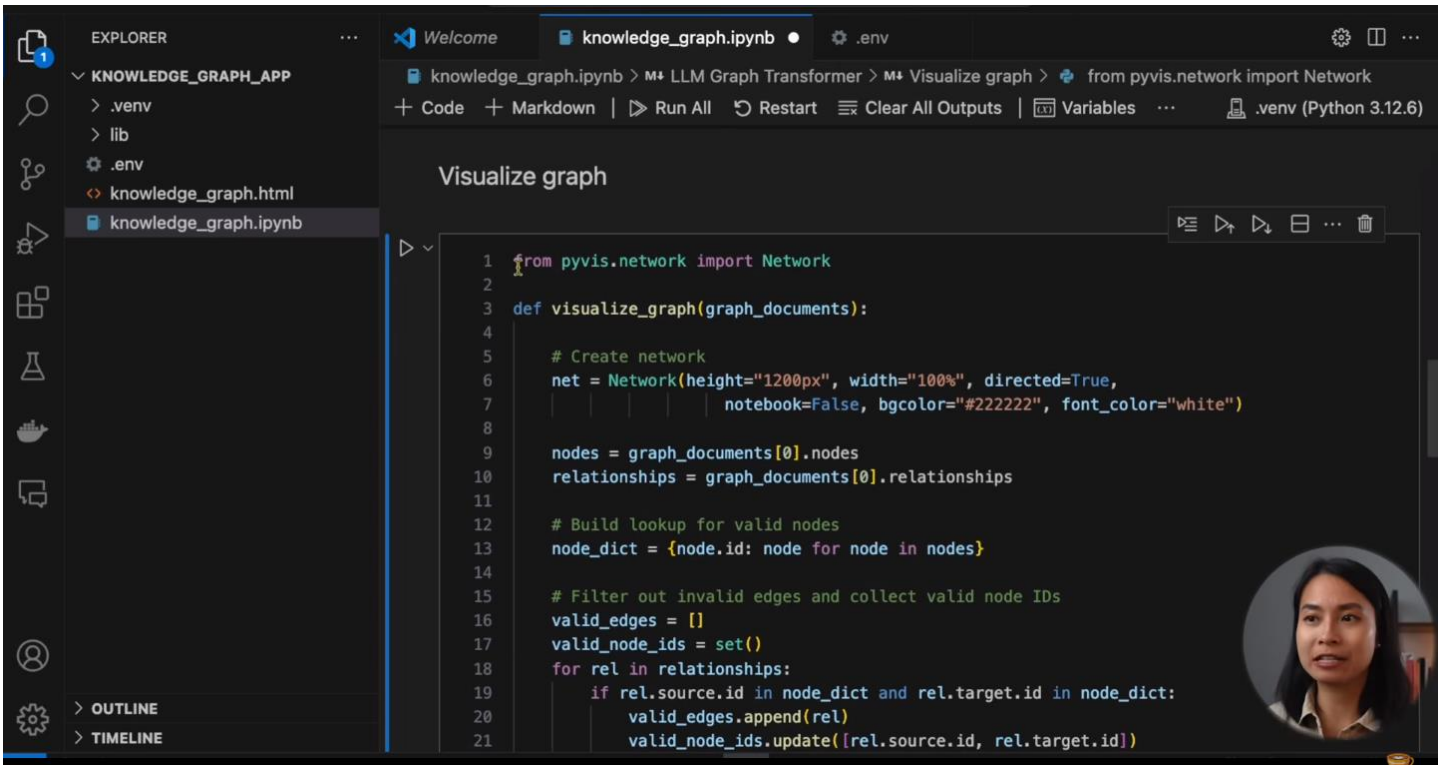


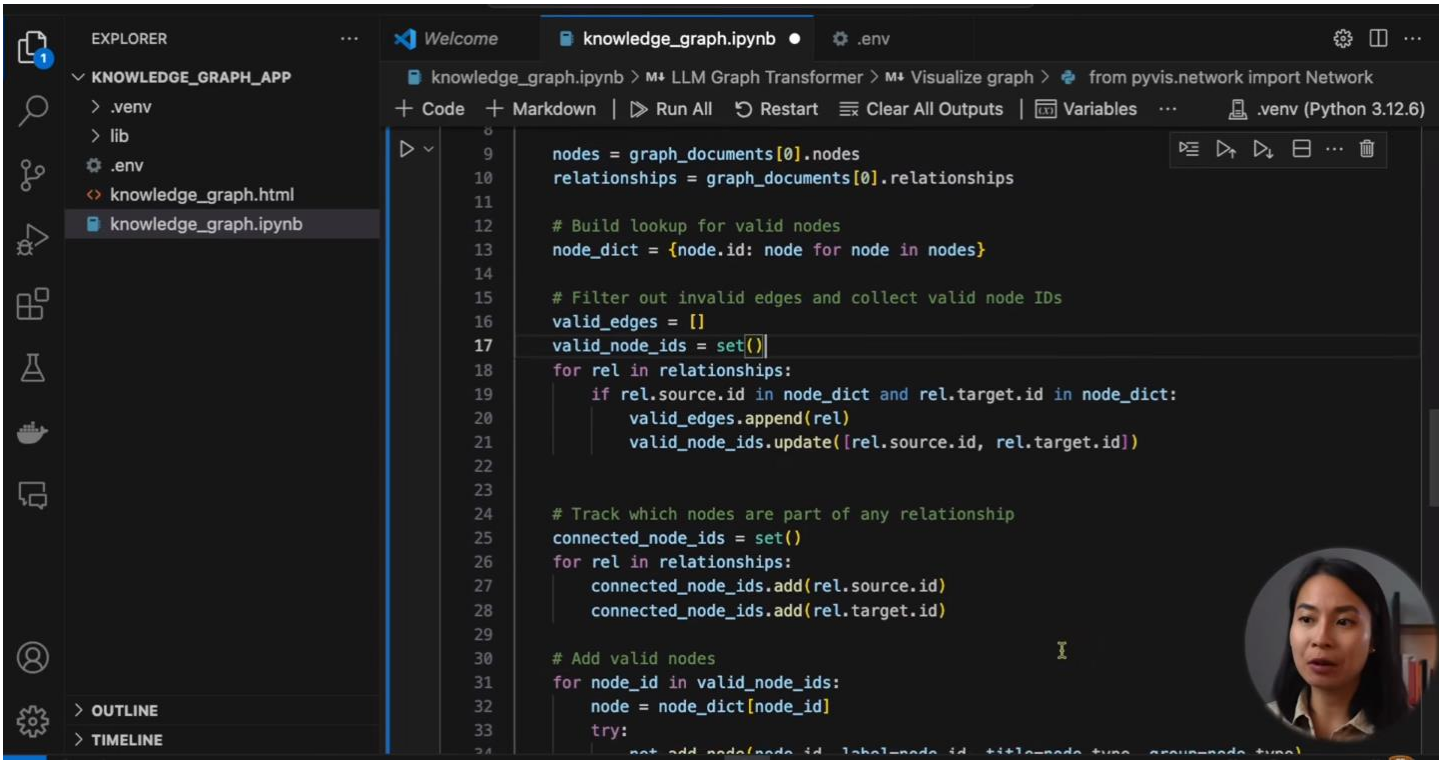
This async function will convert the document into a list of graph documents, it also allows us to process multiple documents in parallel.

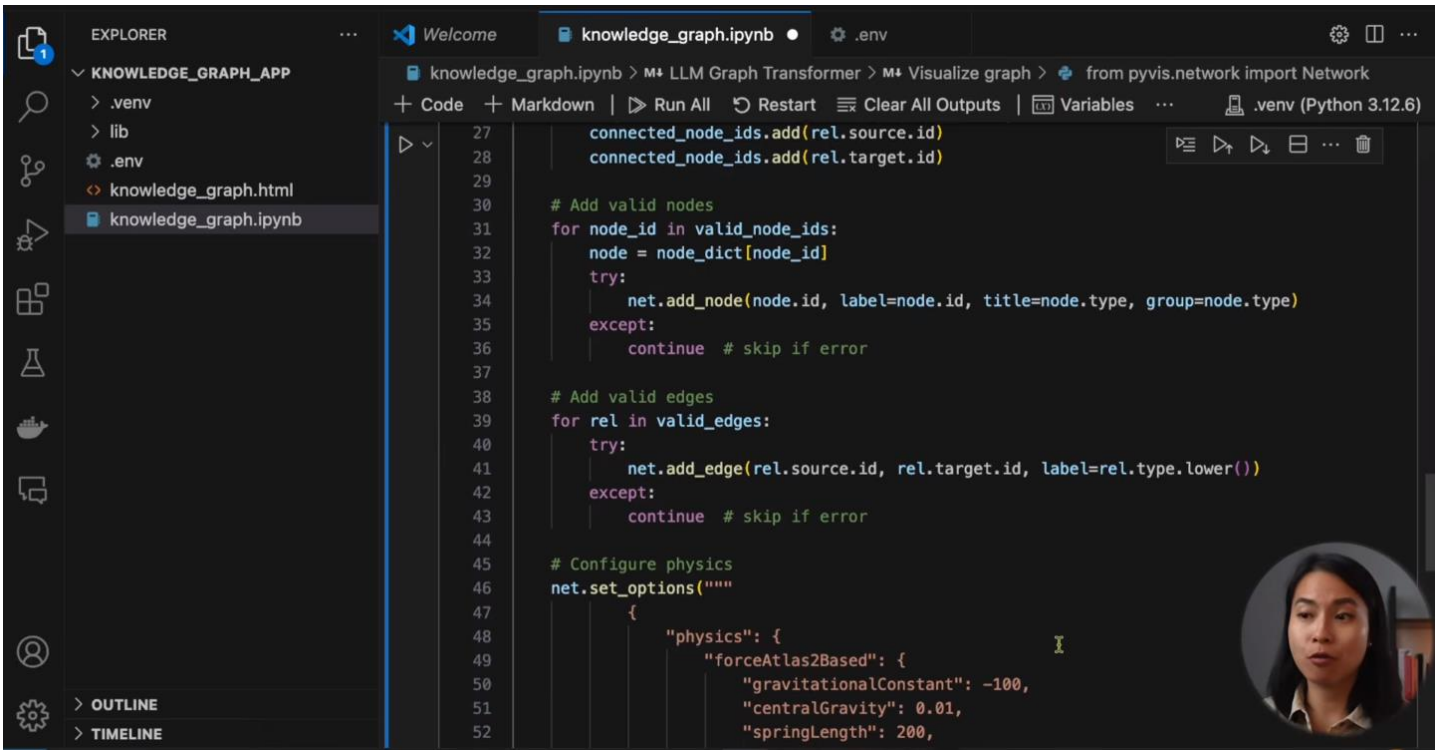


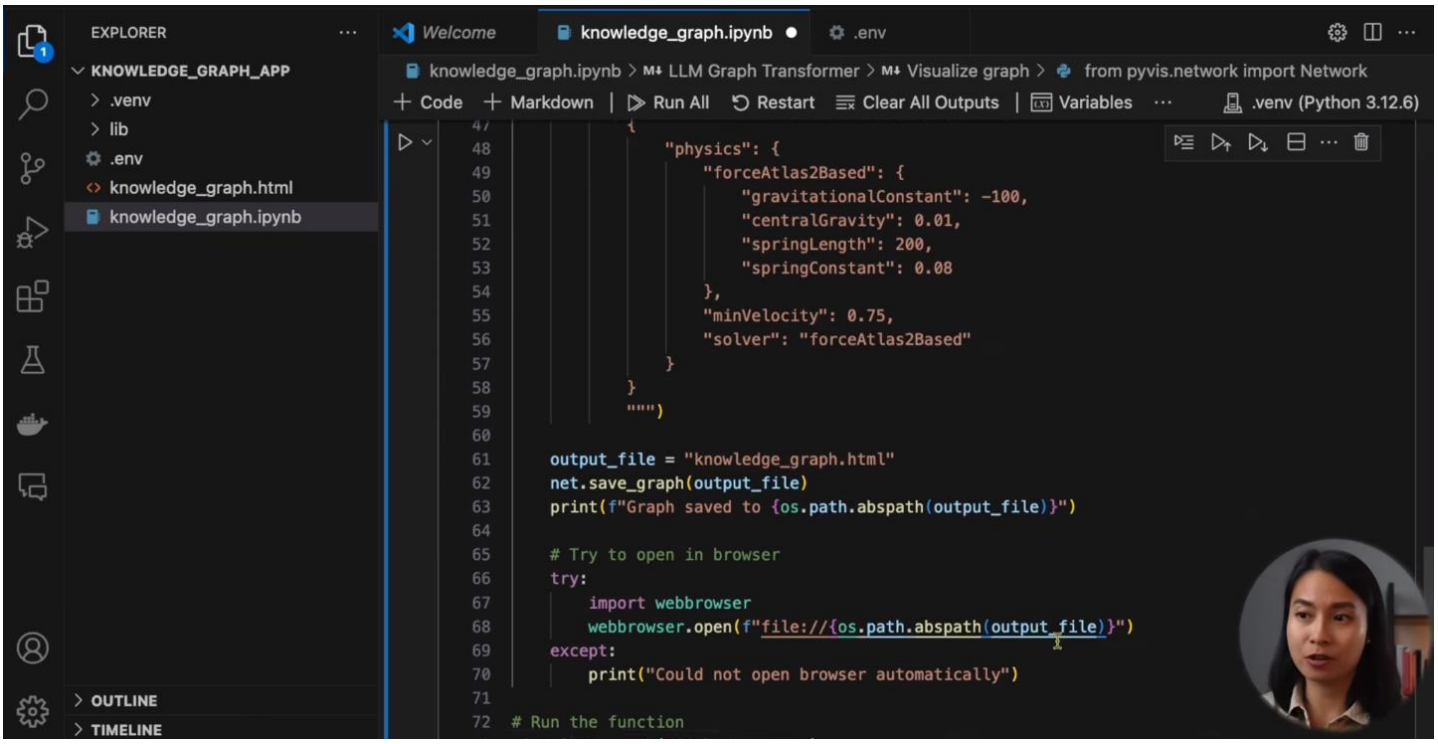
We now have some output to build our knowledge graph

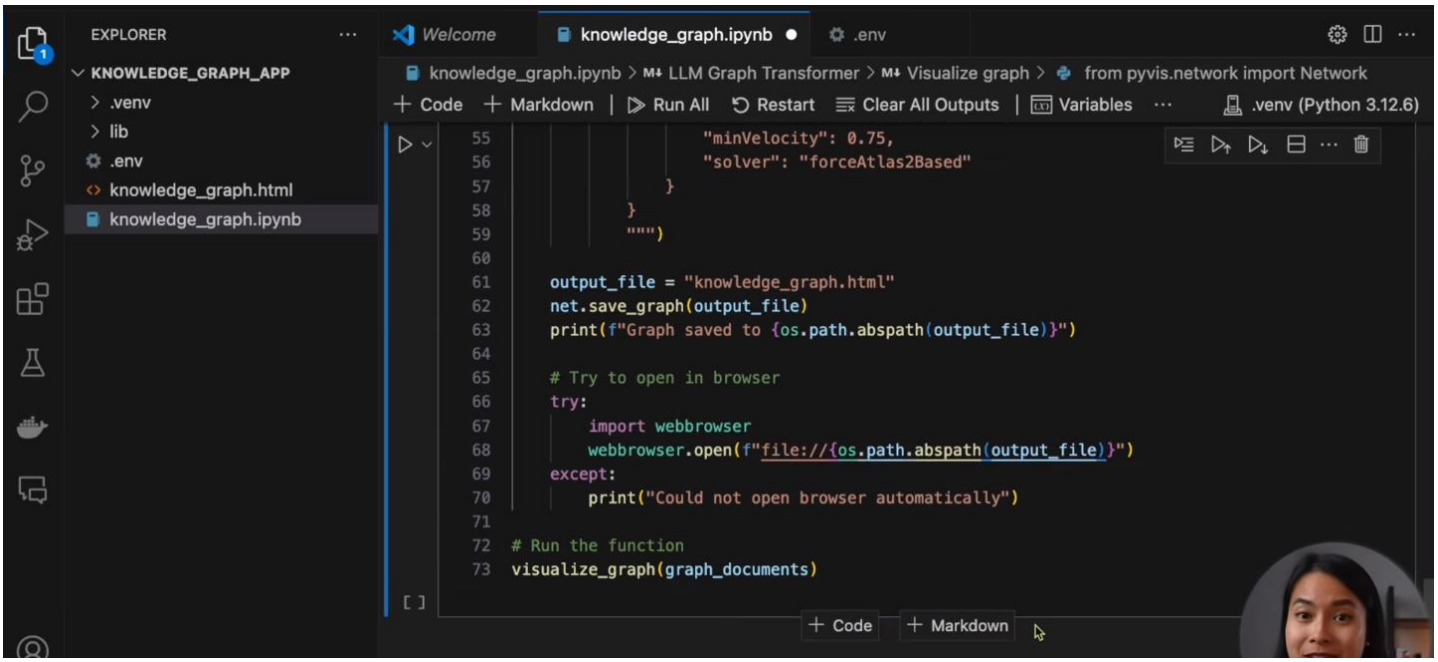



EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > Visualize graph > from pyvis.network import Network
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
Visualize graph
from pyvis.network import Network
def visualize_graph(graph_documents):
# Create network
net = Network(height="1200px", width="100%", directed=True,
notebook=False, bgcolor="#222222", font_color="white")
nodes = graph_documents[0].nodes
relationships = graph_documents[0].relationships
# Build lookup for valid nodes
node_dict = {node.id: node for node in nodes}
# Filter out invalid edges and collect valid node IDs
valid_edges = []
valid_node_ids = set()
for rel in relationships:
if rel.source.id in node_dict and rel.target.id in node_dict:
valid_edges.append(rel)
valid_node_ids.update([rel.source.id, rel.target.id])

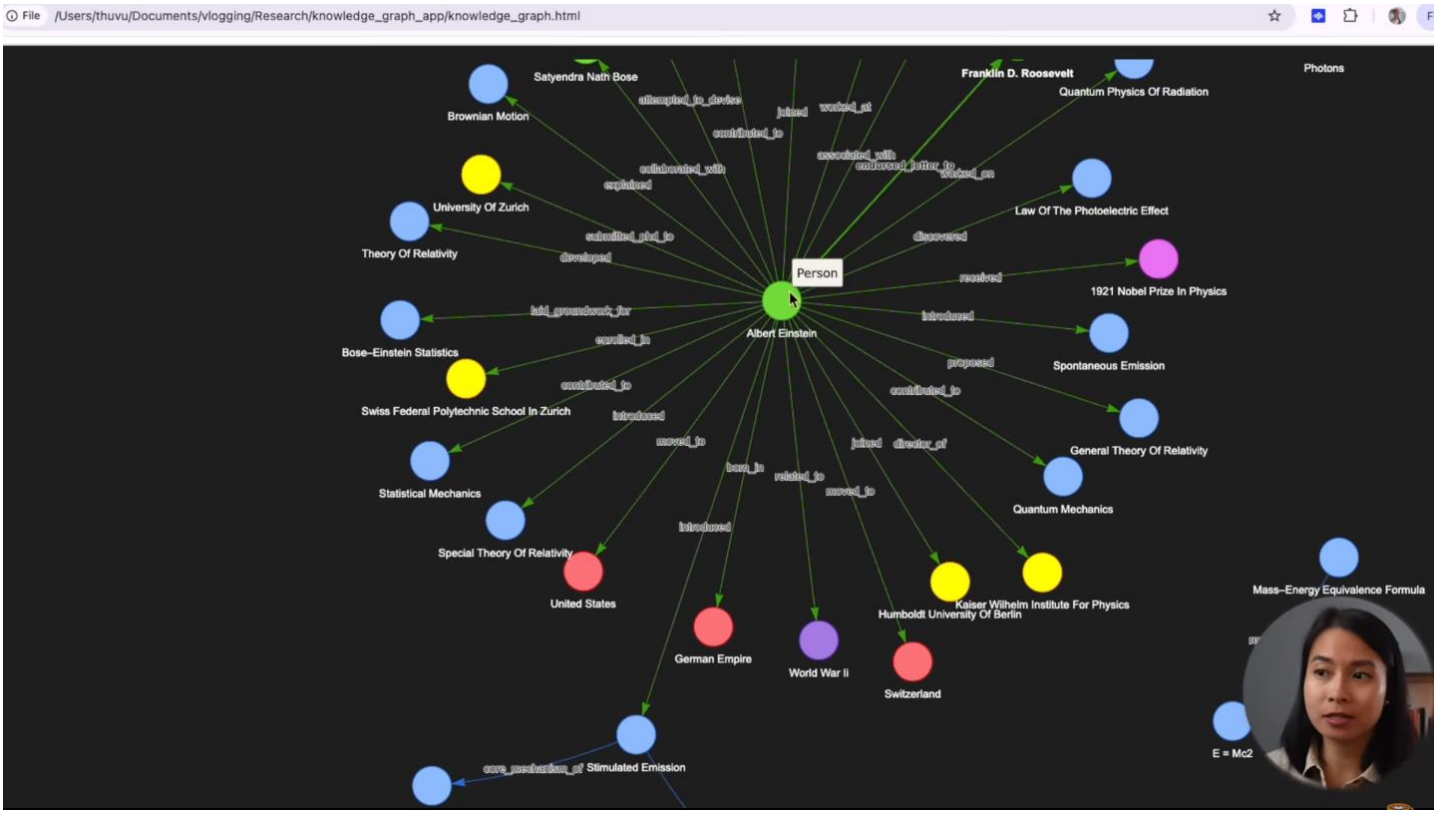

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > Visualize graph > from pyvis.network import Network
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
nodes = graph_documents[0].nodes
relationships = graph_documents[0].relationships
# Build lookup for valid nodes
node_dict = {node.id: node for node in nodes}
# Filter out invalid edges and collect valid node IDs
valid_edges = []
valid_node_ids = set()
for rel in relationships:
if rel.source.id in node_dict and rel.target.id in node_dict:
valid_edges.append(rel)
valid_node_ids.update([rel.source.id, rel.target.id])
# Track which nodes are part of any relationship
connected_node_ids = set()
for rel in relationships:
connected_node_ids.add(rel.source.id)
connected_node_ids.add(rel.target.id)
# Add valid nodes
for node_id in valid_node_ids:
node = node_dict[node_id]
try:


EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > Visualize graph > from pyvis.network import Network
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
connected_node_ids.add(rel.source.id)
connected_node_ids.add(rel.target.id)
# Add valid nodes
for node_id in valid_node_ids:
node = node_dict[node_id]
try:
net.add_node(node.id, label=node.id, title=node.type, group=node.type)
except:
continue # skip if error
# Add valid edges
for rel in valid_edges:
try:
net.add_edge(rel.source.id, rel.target.id, label=rel.type.lower())
except:
continue # skip if error
Configure physics
net.set_options("""
{
"physics": {
"forceAtlas2Based": {
"gravitationalConstant": -100,
"centralGravity": 0.01,
"springLength": 200,

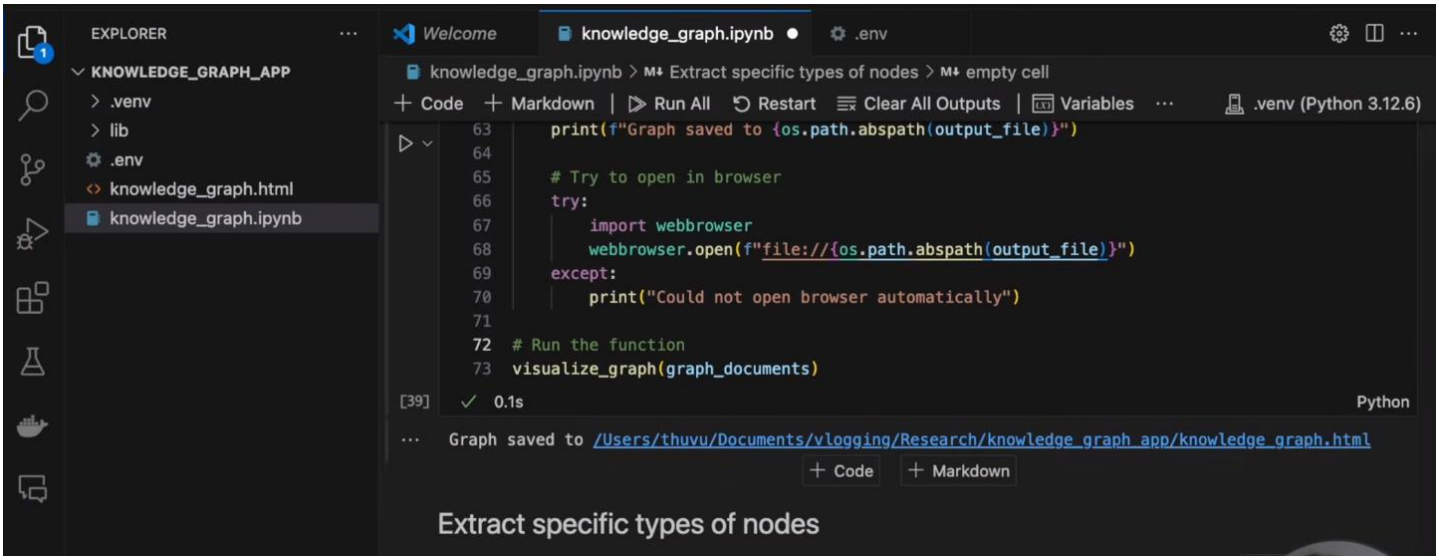

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > Visualize graph > from pyvis.network import Network
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
{
"physics": {
"forceAtlas2Based": {
"gravitationalConstant": -100,
"centralGravity": 0.01,
"springLength": 200,
"springConstant": 0.08
},
"minVelocity": 0.75,
"solver": "forceAtlas2Based"
}
}
""")
output_file = "knowledge_graph.html"
net.save_graph(output_file)
print(f"Graph saved to {os.path.abspath(output_file)}")
# Try to open in browser
try:
import webbrowser
webbrowser.open(f"file://{os.path.abspath(output_file)}")
except:
print("Could not open browser automatically")
# Run the function

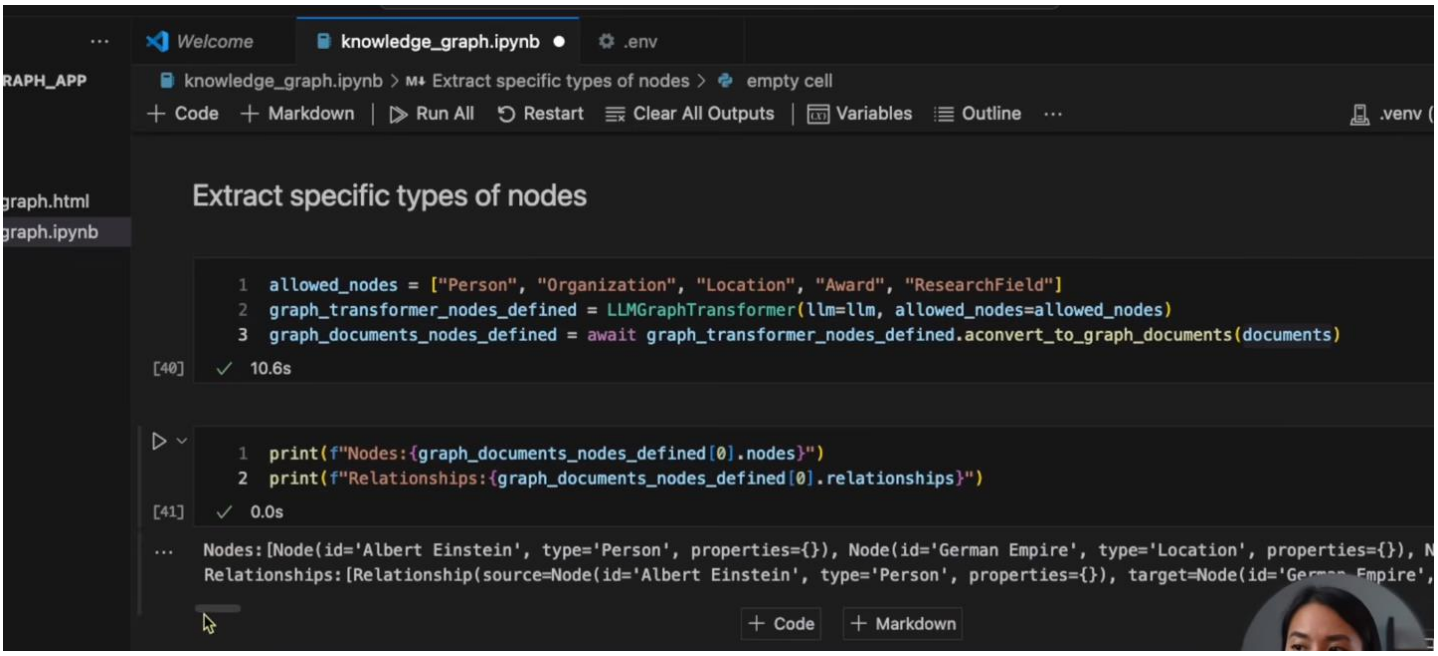

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > LLM Graph Transformer > Visualize graph > from pyvis.network import Network
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
.venv (Python 3.12.6)
"minVelocity": 0.75,
"solver": "forceAtlas2Based"
}
}
"""
output_file = "knowledge_graph.html"
net.save_graph(output_file)
print(f"Graph saved to {os.path.abspath(output_file)}")
# Try to open in browser
try:
import webbrowser
webbrowser.open(f"file://{os.path.abspath(output_file)}")
except:
print("Could not open browser automatically")
# Run the function
visualize_graph(graph_documents)
Code
Markdown

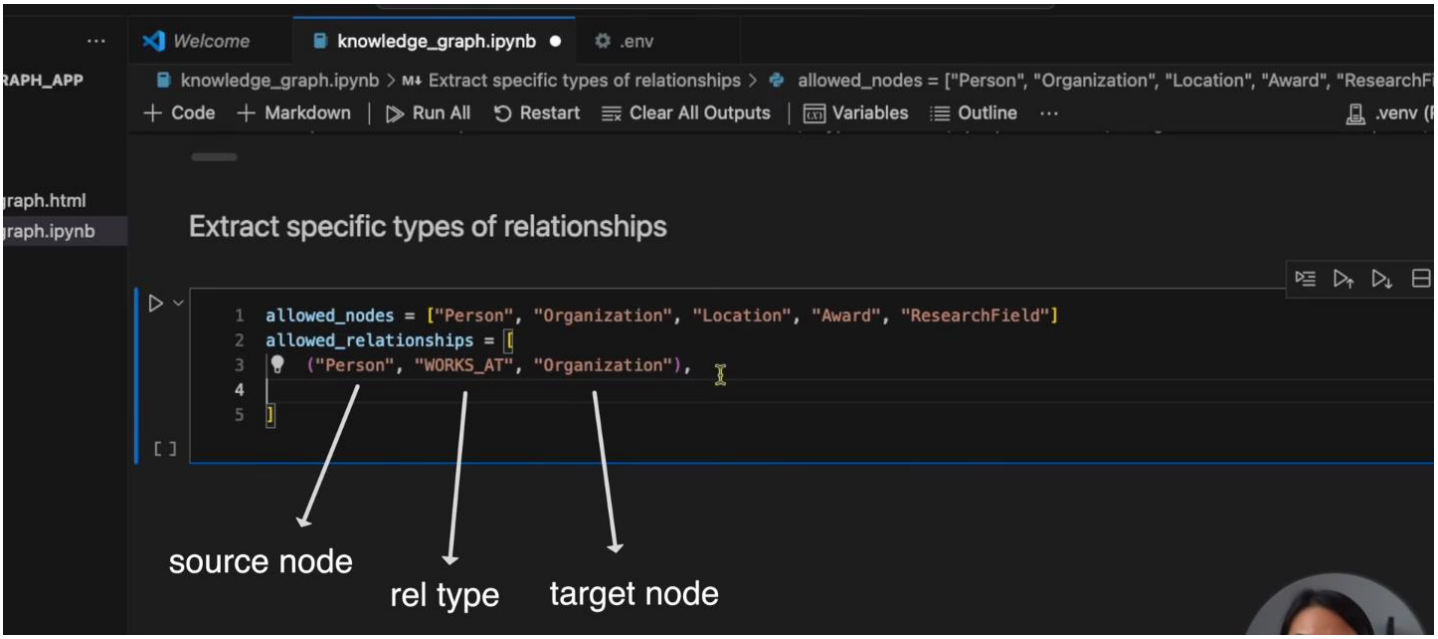

File
/Users/thuvu/Documents/vlogging/Research/knowledge_graph_app/knowledge_graph.html
Finish update
Electromagnetism
Unified Field Theory
Quantum Theory
Swiss Patent Office In Bern
Prussian Academy Of Sciences
Annus Mirabilis
Quantum Physics Of Radiation
Photons
Satyendra Nath Bose
Brownian Motion
Franklin D. Roosevelt
University Of Zurich
Law Of The Photoelectric Effect
Theory Of Relativity
1921 Nobel Prize In Physics
Albert Einstein
Bose–Einstein Statistics
Spontaneous Emission
Swiss Federal Polytechnic School In Zurich
General Theory Of Relativity
Statistical Mechanics
Quantum Mechanics
Special Theory Of Relativity
United States
Mass–Energy Equivalence Formula
German Empire
World War II
Switzerland
E = Mc2



We can also decide to focus on certain things like person (entities) and organizations (relationships), we can provide constraints and guidelines to the LLMs as below




Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > Extract specific types of nodes > empty cell
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
.venv
Extract specific types of nodes
allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchField"]
graph_transformer_nodes_defined = LLMGraphTransformer(llm=llm, allowed_nodes=allowed_nodes)
graph_documents_nodes_defined = await graph_transformer_nodes_defined.aconvert_to_graph_documents(documents)
[40] 10.6s
print(f"Nodes:{graph_documents_nodes_defined[0].nodes}")
print(f"Relationships:{graph_documents_nodes_defined[0].relationships}")
[41] 0.0s
Nodes:[Node(id='Albert Einstein', type='Person', properties={}), Node(id='German Empire', type='Location', properties={}),
Relationships:[Relationship(source=Node(id='Albert Einstein', type='Person', properties={}), target=Node(id='German Empire',
Code
Markdown


Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > Extract specific types of relationships > allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchF
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
.venv
Extract specific types of relationships
allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchField"]
allowed_relationships = [
("Person", "WORKS_AT", "Organization"),
]
source node
rel type
target node

graph.html
graph.ipynb

Extract specific types of relationships

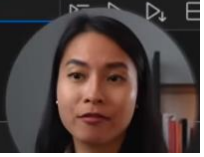
```python
allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchField"]
allowed_relationships = [
    ("Person", "WORKS_AT", "Organization"),
    ("Person", "SPOUSE", "Person"),
    ("Person", "AWARD", "Award"),
    ("Organization", "IN_LOCATION", "Location"),
    ("Person", "FIELD_OF_RESEARCH", "ResearchField")
]
graph_transformer_rel_defined = LLMGraphTransformer(
  llm=llm,
  allowed_nodes=allowed_nodes,
  allowed_relationships=allowed_relationships
)
graph_documents_rel_defined = await graph_transformer_rel_defined.aconvert_to_graph_documents(documents)
```

[42] ✓ 8.2s

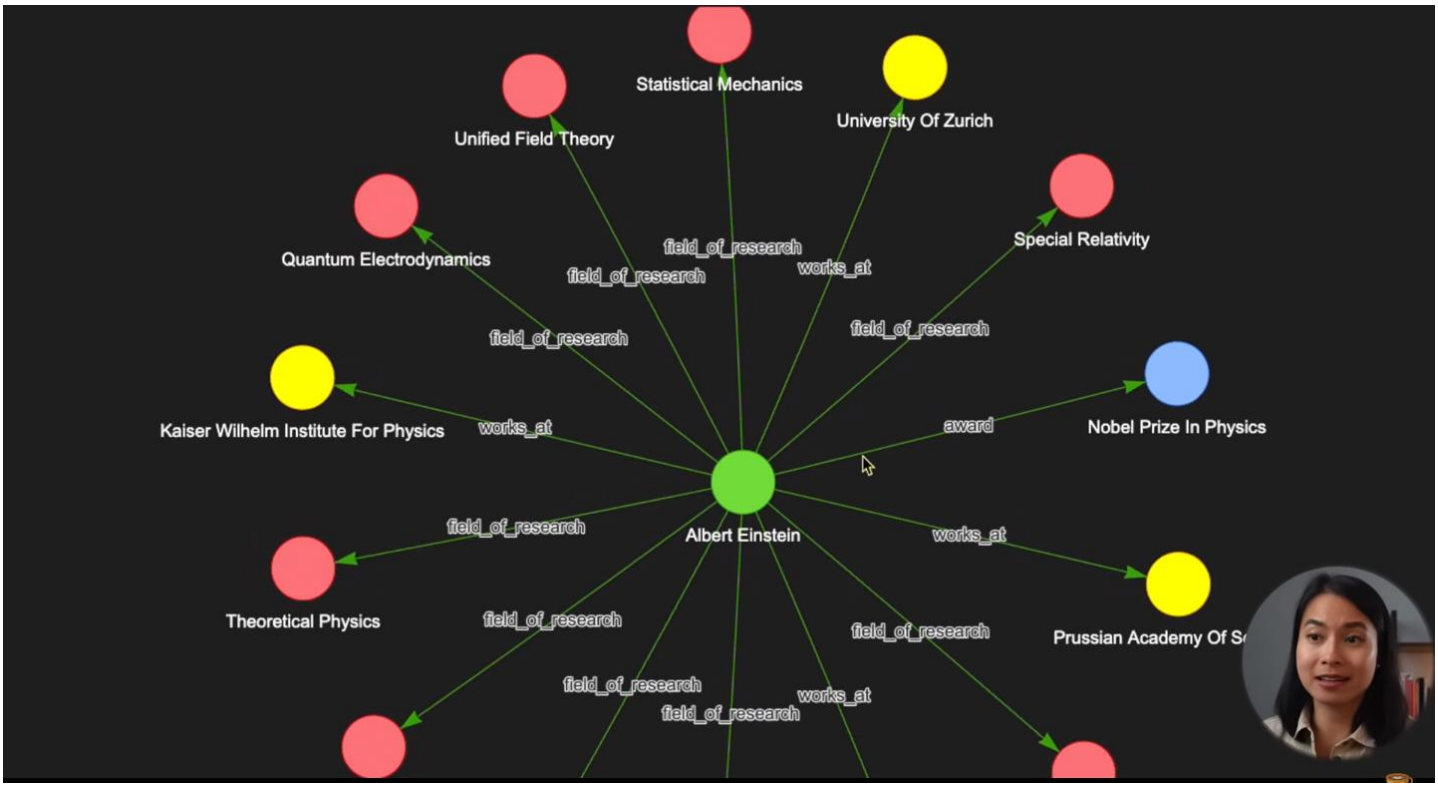
```python
# Visualize graph
visualize_graph()
```

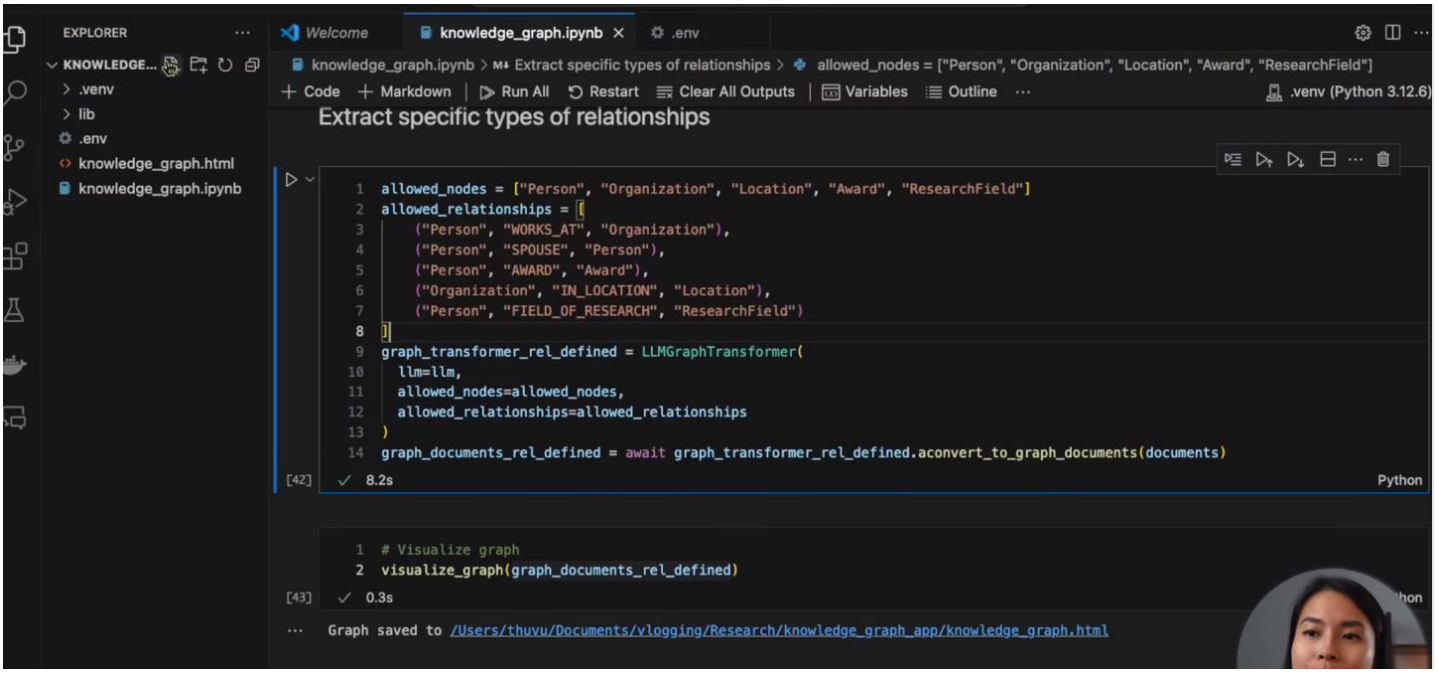
[ ]

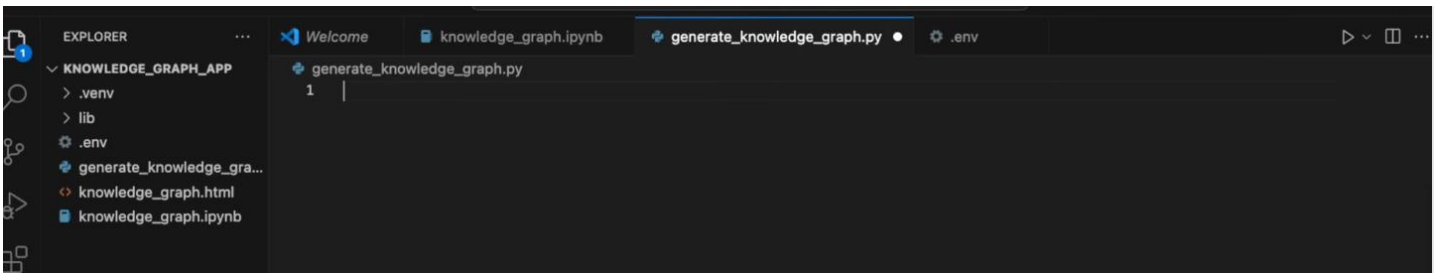
+ Code + Markdown



sers/thuvu/Documents/vlogging/Research/knowledge_graph_app/knowledge_graph.html


Statistical Mechanics
Unified Field Theory
University Of Zurich
Quantum Electrodynamics
Special Relativity
field_of_research
field_of_research
works_at
field_of_research
field_of_research
Kaiser Wilhelm Institute For Physics
works_at
award
Nobel Prize In Physics
Albert Einstein
field_of_research
works_at
Theoretical Physics
field_of_research
field_of_research
Prussian Academy Of S
field_of_research
works_at
field_of_research


EXPLORER
KNOWLEDGE...
.venv
lib
.env
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
.env
knowledge_graph.ipynb > Extract specific types of relationships > allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchField"]
Code
Markdown
Run All
Restart
Clear All Outputs
Variables
Outline
.venv (Python 3.12.6)
Extract specific types of relationships
allowed_nodes = ["Person", "Organization", "Location", "Award", "ResearchField"]
allowed_relationships = [
("Person", "WORKS_AT", "Organization"),
("Person", "SPOUSE", "Person"),
("Person", "AWARD", "Award"),
("Organization", "IN_LOCATION", "Location"),
("Person", "FIELD_OF_RESEARCH", "ResearchField")
]
graph_transformer_rel_defined = LLMGraphTransformer(
llm=llm,
allowed_nodes=allowed_nodes,
allowed_relationships=allowed_relationships
)
graph_documents_rel_defined = await graph_transformer_rel_defined.aconvert_to_graph_documents(documents)
[42]
8.2s
Python
# Visualize graph
visualize_graph(graph_documents_rel_defined)
[43]
0.3s
Graph saved to /Users/thuvu/Documents/vlogging/Research/knowledge_graph_app/knowledge_graph.html


EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py
1

EXPLORER

KNOWLEDGE_GRAPH_APP
- .venv
- lib
- .env
- generate_knowledge_gra...
- knowledge_graph.html
- knowledge_graph.ipynb

Welcome | knowledge_graph.ipynb | generate_knowledge_graph.py ● | .env

generate_knowledge_graph.py > llm

```python
from langchain_experimental.graph_transformers import LLMGraphTransformer
from langchain_core.documents import Document
from langchain_openai import ChatOpenAI
from pyvis.network import Network

from dotenv import load_dotenv
import os
import asyncio


# Load the .env file
load_dotenv()
# Get API key from environment variable
api_key = os.getenv("OPENAI_API_KEY")

llm = ChatOpenAI(temperature=0, model_name="gpt-4o")

graph_transformer = LLMGraphTransformer(llm=llm)

```

EXPLORER

KNOWLEDGE_GRAPH_APP
- .venv
- lib
- .env
- generate_knowledge_gra...
- knowledge_graph.html
- knowledge_graph.ipynb

Welcome | knowledge_graph.ipynb | generate_knowledge_graph.py × | .env

generate_knowledge_graph.py > extract_graph_data

```python
from langchain_core.documents import Document
from langchain_openai import ChatOpenAI
from pyvis.network import Network

from dotenv import load_dotenv
import os
import asyncio


# Load the .env file
load_dotenv()
# Get API key from environment variable
api_key = os.getenv("OPENAI_API_KEY")

llm = ChatOpenAI(temperature=0, model_name="gpt-4o")

graph_transformer = LLMGraphTransformer(llm=llm)


# Extract graph data from input text
async def extract_graph_data(text):
    """
    Asynchronously extracts graph data from input text using a graph transformer.

    Args:
        text (str): Input text to be processed into graph format.

    Returns:
        list: A list of GraphDocument objects containing nodes and relationships.
    """
    documents = [Document(page_content=text)]
    graph_documents = await graph_transformer.aconvert_to_graph_documents(documents)
    return graph_documents
```

OUTLINE

TIMELINE

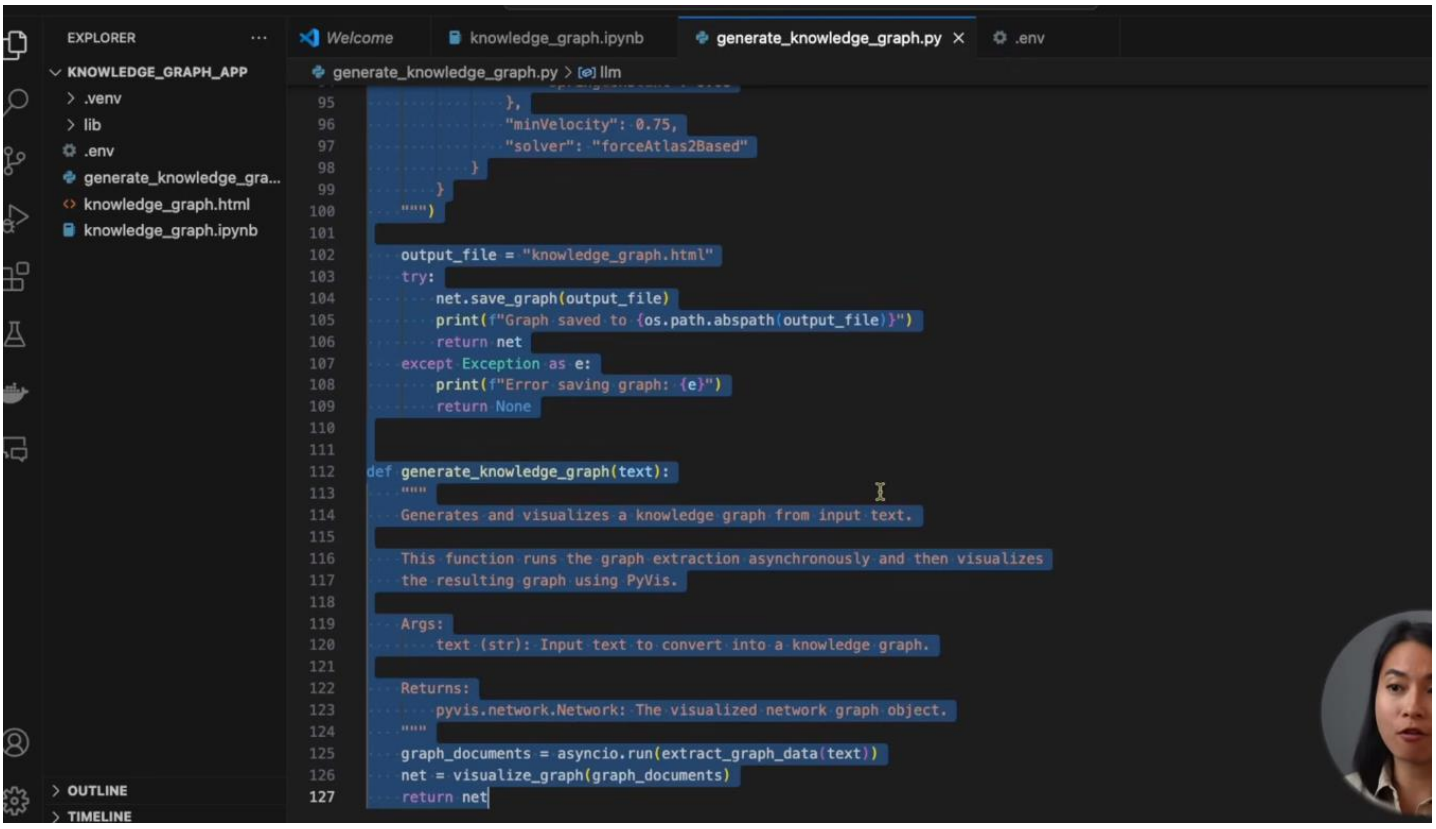
EXPLORER

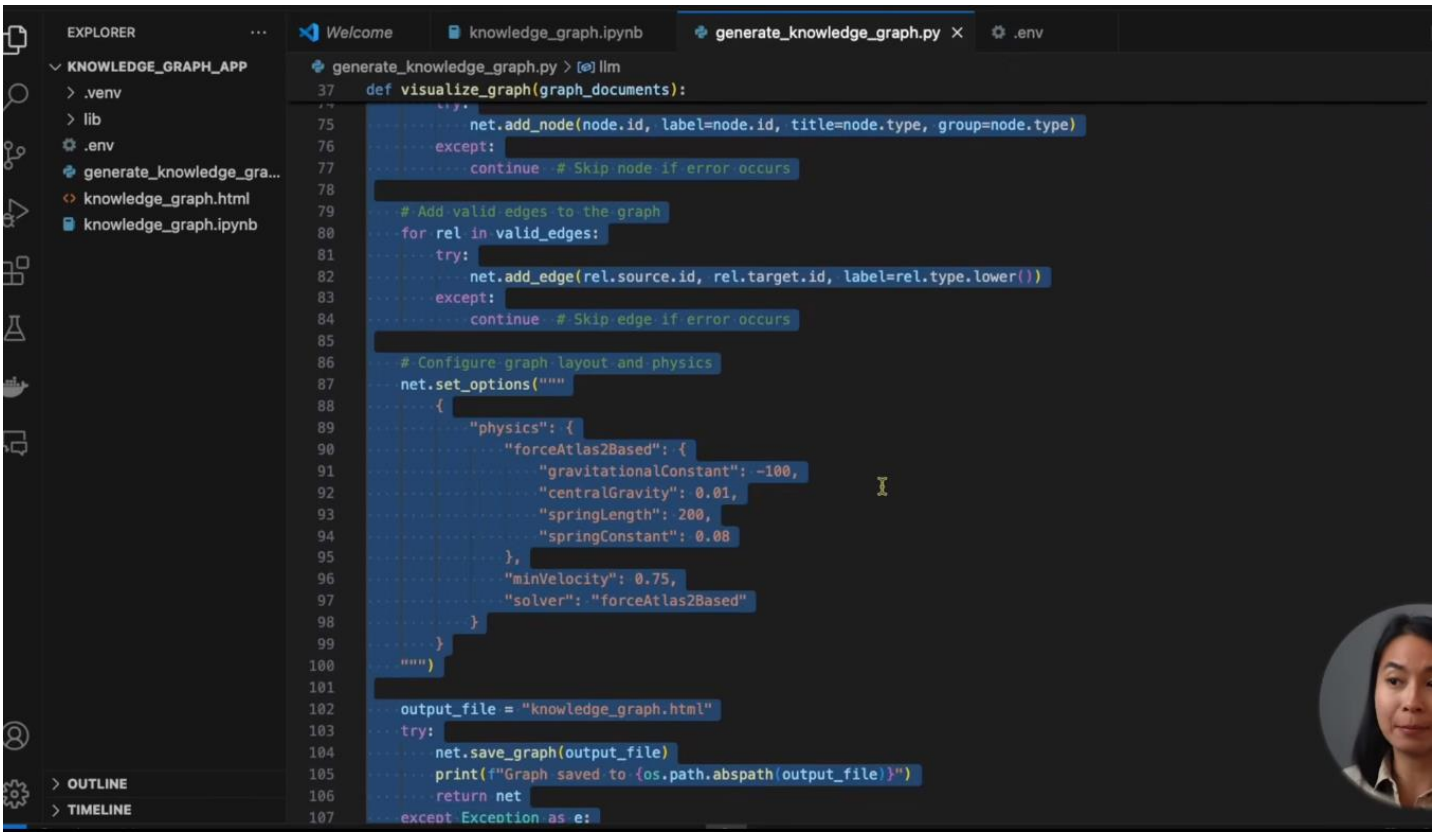
KNOWLEDGE_GRAPH_APP
- .venv
- lib
- .env
- generate_knowledge_gra...
- knowledge_graph.html
- knowledge_graph.ipynb

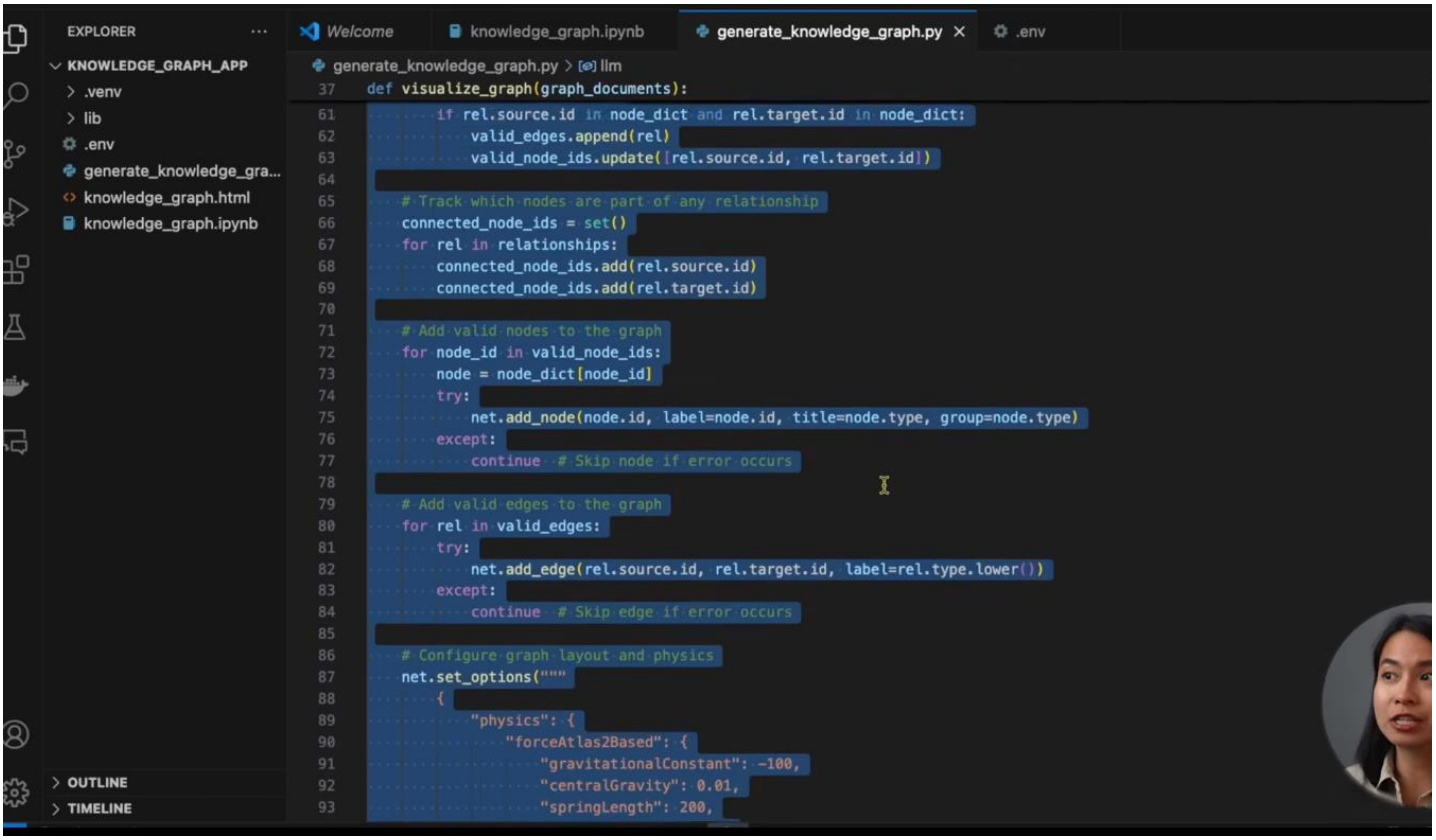
Welcome | knowledge_graph.ipynb | generate_knowledge_graph.py ● | .env

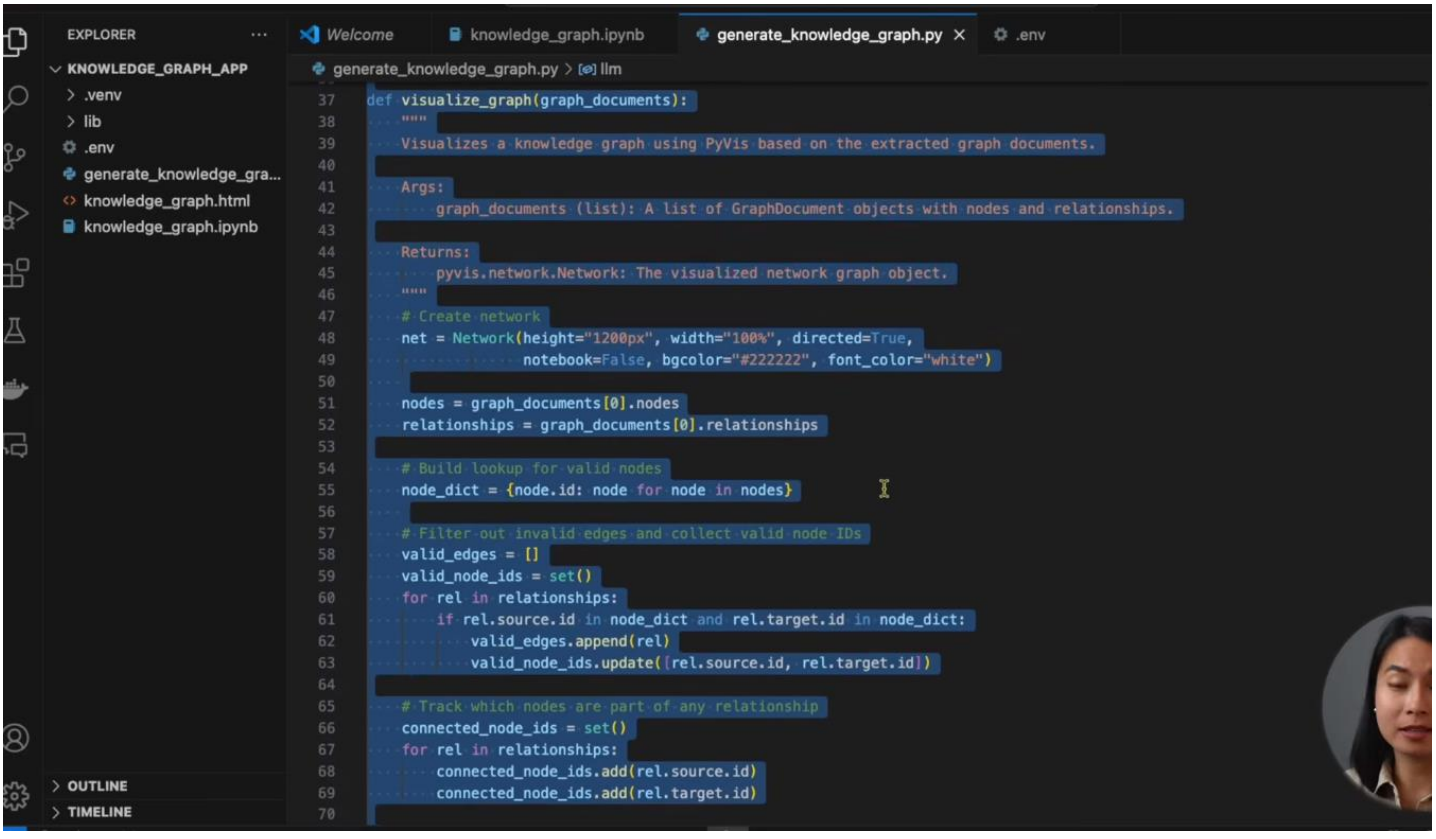
generate_knowledge_graph.py > ...

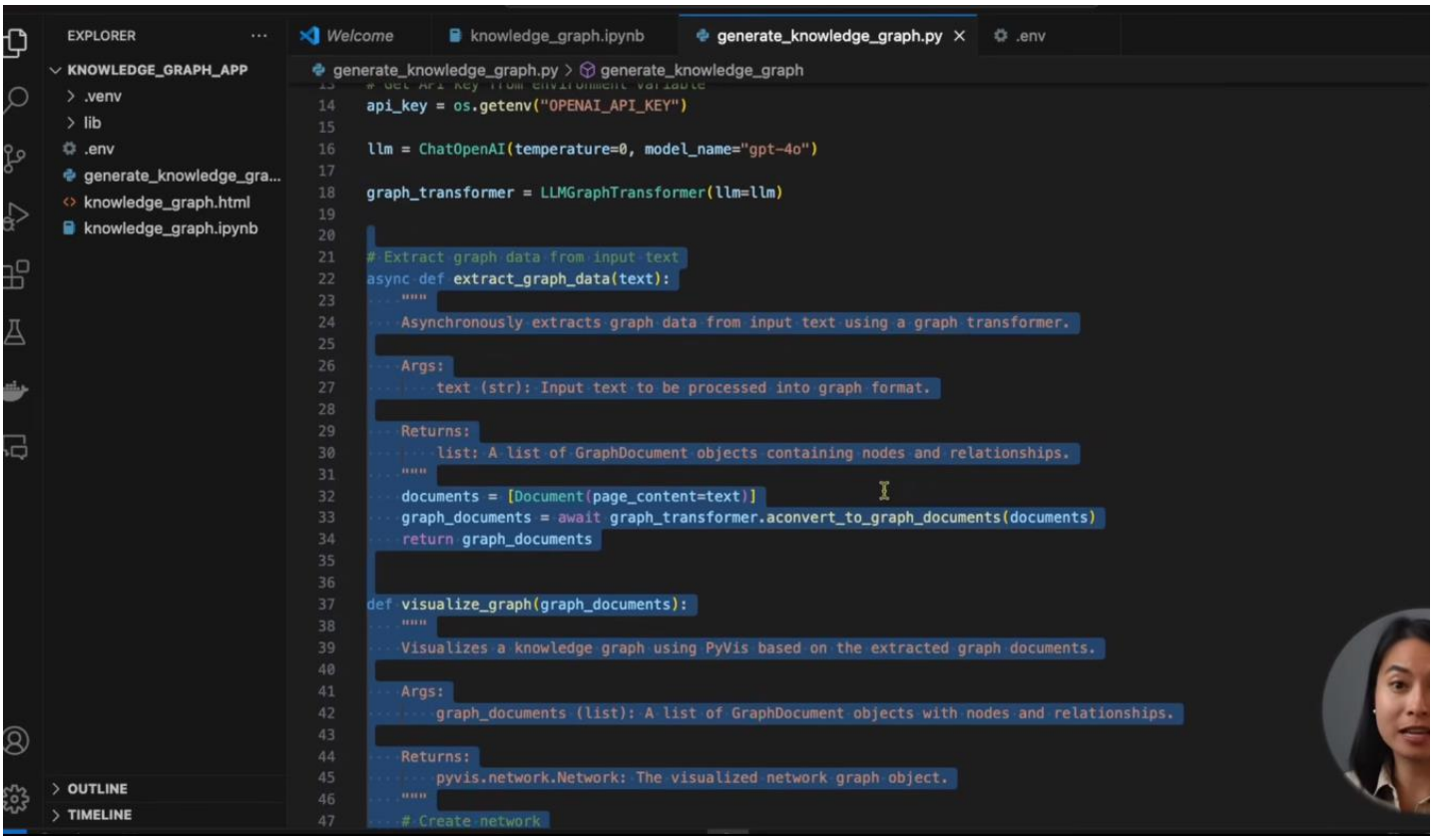
```python
from langchain_experimental.graph_transformers import LLMGraphTransformer
from langchain_core.documents import Document
from langchain_openai import ChatOpenAI
from pyvis.network import Network

from dotenv import load_dotenv
import os
import asyncio


# Load the .env file
load_dotenv()
# Get API key from environment variable
api_key = os.getenv("OPENAI_API_KEY")

llm = ChatOpenAI(temperature=0, model_name="gpt-4o")

graph_transformer = LLMGraphTransformer(llm=llm)

```

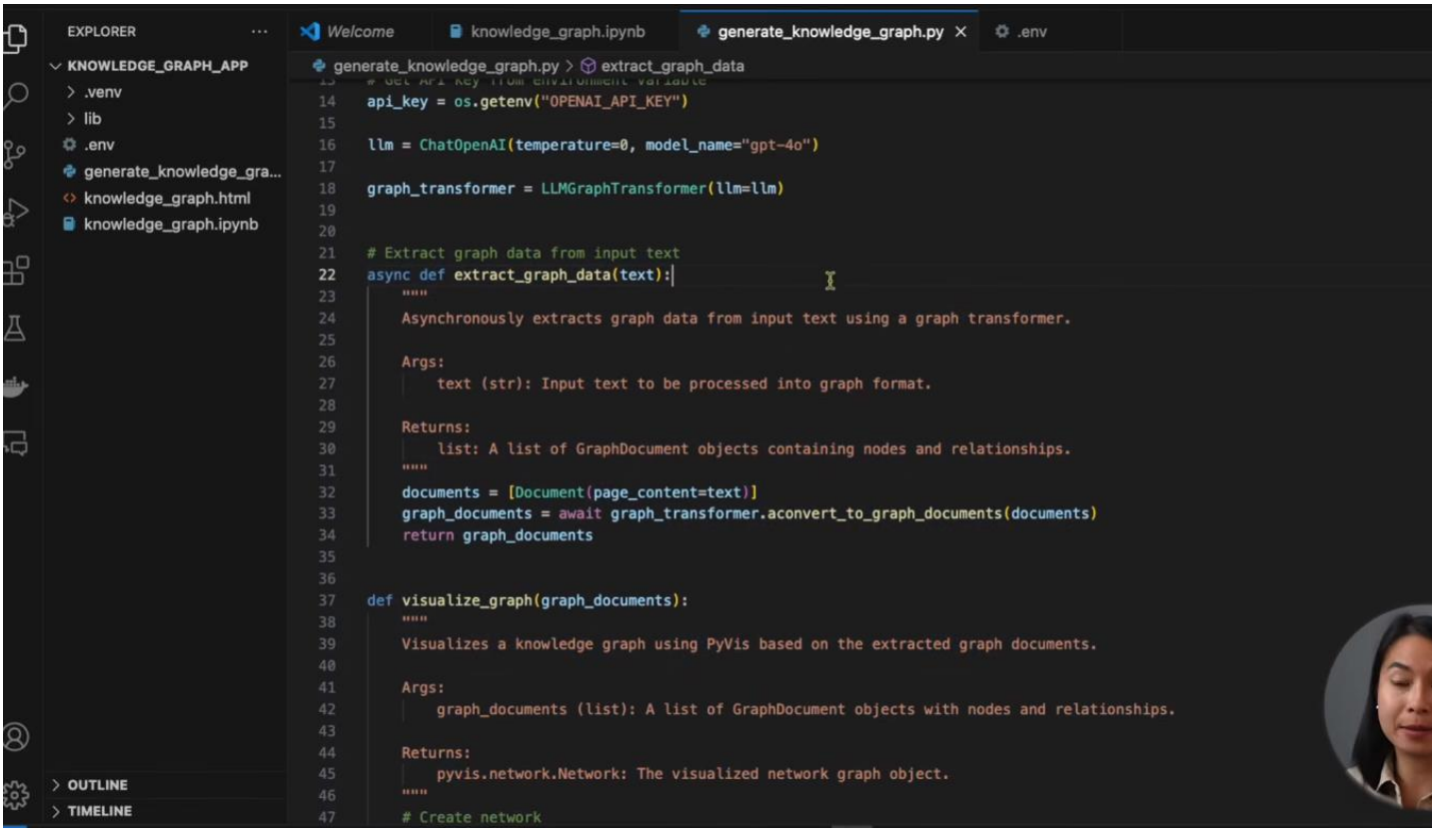

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > llm
},
"minVelocity": 0.75,
"solver": "forceAtlas2Based"
}
}
"""
output_file = "knowledge_graph.html"
try:
net.save_graph(output_file)
print(f"Graph saved to {os.path.abspath(output_file)}")
return net
except Exception as e:
print(f"Error saving graph: {e}")
return None
def generate_knowledge_graph(text):
"""
Generates and visualizes a knowledge graph from input text.
This function runs the graph extraction asynchronously and then visualizes
the resulting graph using PyVis.
Args:
text (str): Input text to convert into a knowledge graph.
Returns:
pyvis.network.Network: The visualized network graph object.
"""
graph_documents = asyncio.run(extract_graph_data(text))
net = visualize_graph(graph_documents)
return net
OUTLINE
TIMELINE

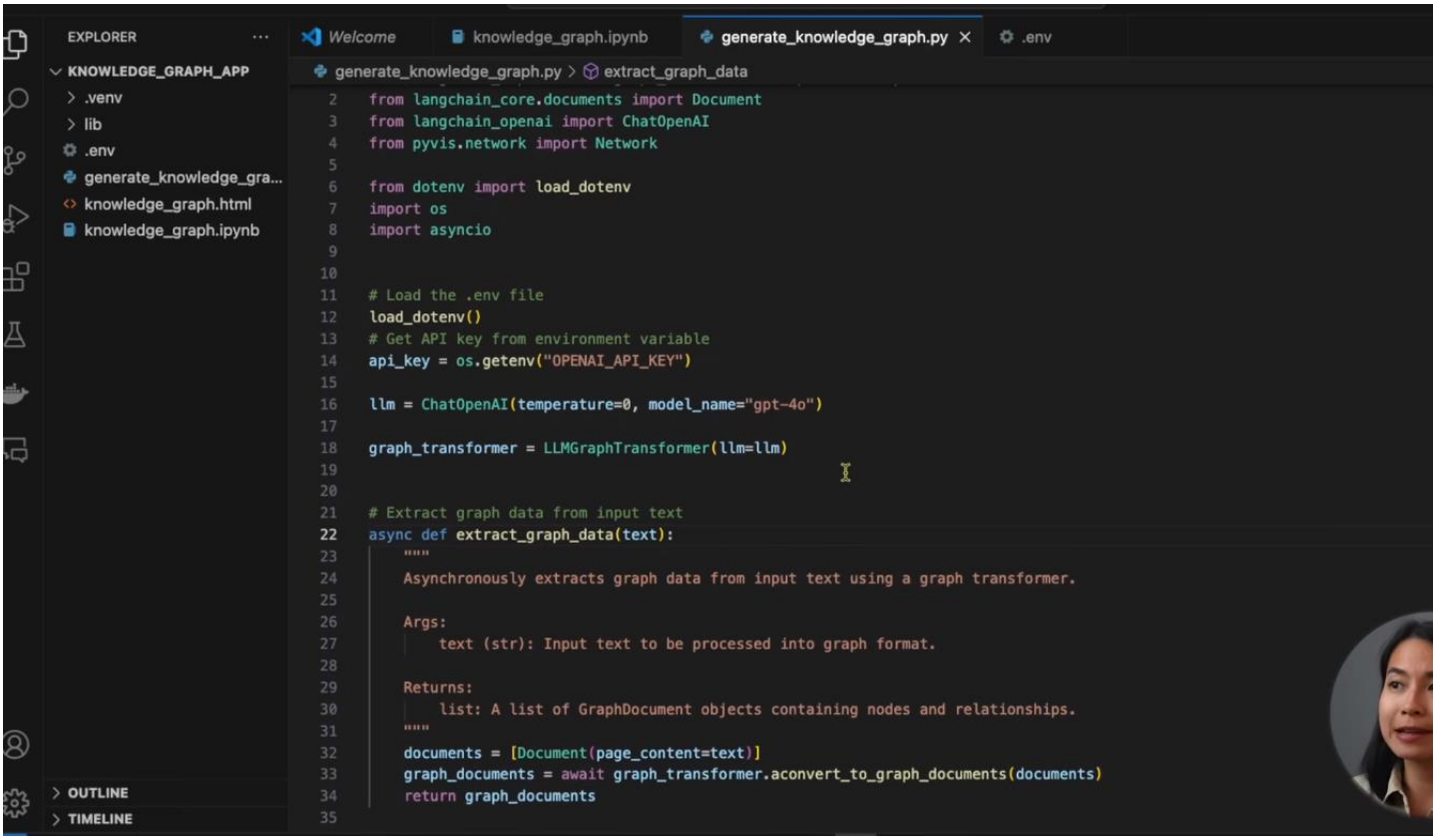

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > llm
def visualize_graph(graph_documents):
net.add_node(node.id, label=node.id, title=node.type, group=node.type)
except:
continue # Skip node if error occurs
# Add valid edges to the graph
for rel in valid_edges:
try:
net.add_edge(rel.source.id, rel.target.id, label=rel.type.lower())
except:
continue # Skip edge if error occurs
Configure graph layout and physics
net.set_options("""
{
"physics": {
"forceAtlas2Based": {
"gravitationalConstant": -100,
"centralGravity": 0.01,
"springLength": 200,
"springConstant": 0.08
},
"minVelocity": 0.75,
"solver": "forceAtlas2Based"
}
}
""")
output_file = "knowledge_graph.html"
try:
net.save_graph(output_file)
print(f"Graph saved to {os.path.abspath(output_file)}")
return net
except Exception as e:
OUTLINE
TIMELINE

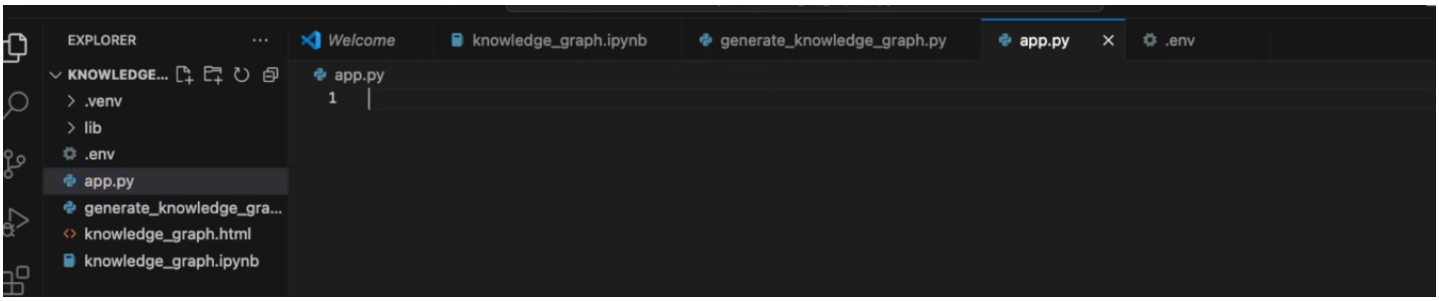

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > llm
def visualize_graph(graph_documents):
if rel.source.id in node_dict and rel.target.id in node_dict:
valid_edges.append(rel)
valid_node_ids.update([rel.source.id, rel.target.id])
# Track which nodes are part of any relationship
connected_node_ids = set()
for rel in relationships:
connected_node_ids.add(rel.source.id)
connected_node_ids.add(rel.target.id)
# Add valid nodes to the graph
for node_id in valid_node_ids:
node = node_dict[node_id]
try:
net.add_node(node.id, label=node.id, title=node.type, group=node.type)
except:
continue # Skip node if error occurs
# Add valid edges to the graph
for rel in valid_edges:
try:
net.add_edge(rel.source.id, rel.target.id, label=rel.type.lower())
except:
continue # Skip edge if error occurs
Configure graph layout and physics
net.set_options("""
{
"physics": {
"forceAtlas2Based": {
"gravitationalConstant": -100,
"centralGravity": 0.01,
"springLength": 200,


EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > llm
def visualize_graph(graph_documents):
"""
Visualizes a knowledge graph using PyVis based on the extracted graph documents.
Args:
graph_documents (list): A list of GraphDocument objects with nodes and relationships.
Returns:
pyvis.network.Network: The visualized network graph object.
"""
# Create network
net = Network(height="1200px", width="100%", directed=True,
notebook=False, bgcolor="#222222", font_color="white")
nodes = graph_documents[0].nodes
relationships = graph_documents[0].relationships
# Build lookup for valid nodes
node_dict = {node.id: node for node in nodes}
# Filter out invalid edges and collect valid node IDs
valid_edges = []
valid_node_ids = set()
for rel in relationships:
if rel.source.id in node_dict and rel.target.id in node_dict:
valid_edges.append(rel)
valid_node_ids.update([rel.source.id, rel.target.id])
# Track which nodes are part of any relationship
connected_node_ids = set()
for rel in relationships:
connected_node_ids.add(rel.source.id)
connected_node_ids.add(rel.target.id)

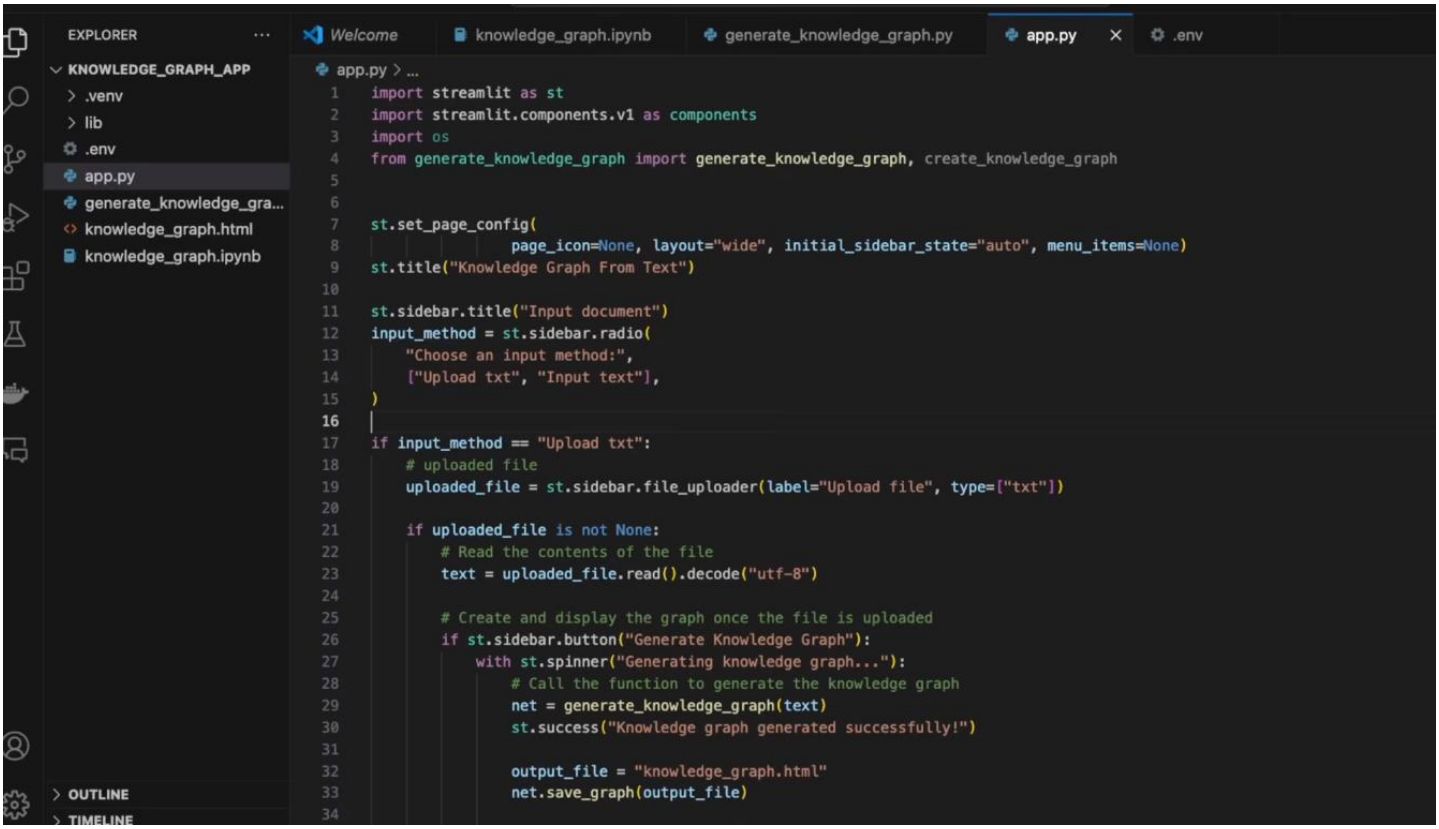

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > generate_knowledge_graph
api_key = os.getenv("OPENAI_API_KEY")
llm = ChatOpenAI(temperature=0, model_name="gpt-4o")
graph_transformer = LLMGraphTransformer(llm=llm)
# Extract graph data from input text
async def extract_graph_data(text):
"""
Asynchronously extracts graph data from input text using a graph transformer.
Args:
text (str): Input text to be processed into graph format.
Returns:
list: A list of GraphDocument objects containing nodes and relationships.
"""
documents = [Document(page_content=text)]
graph_documents = await graph_transformer.aconvert_to_graph_documents(documents)
return graph_documents
def visualize_graph(graph_documents):
"""
Visualizes a knowledge graph using PyVis based on the extracted graph documents.
Args:
graph_documents (list): A list of GraphDocument objects with nodes and relationships.
Returns:
pyvis.network.Network: The visualized network graph object.
"""
Create network

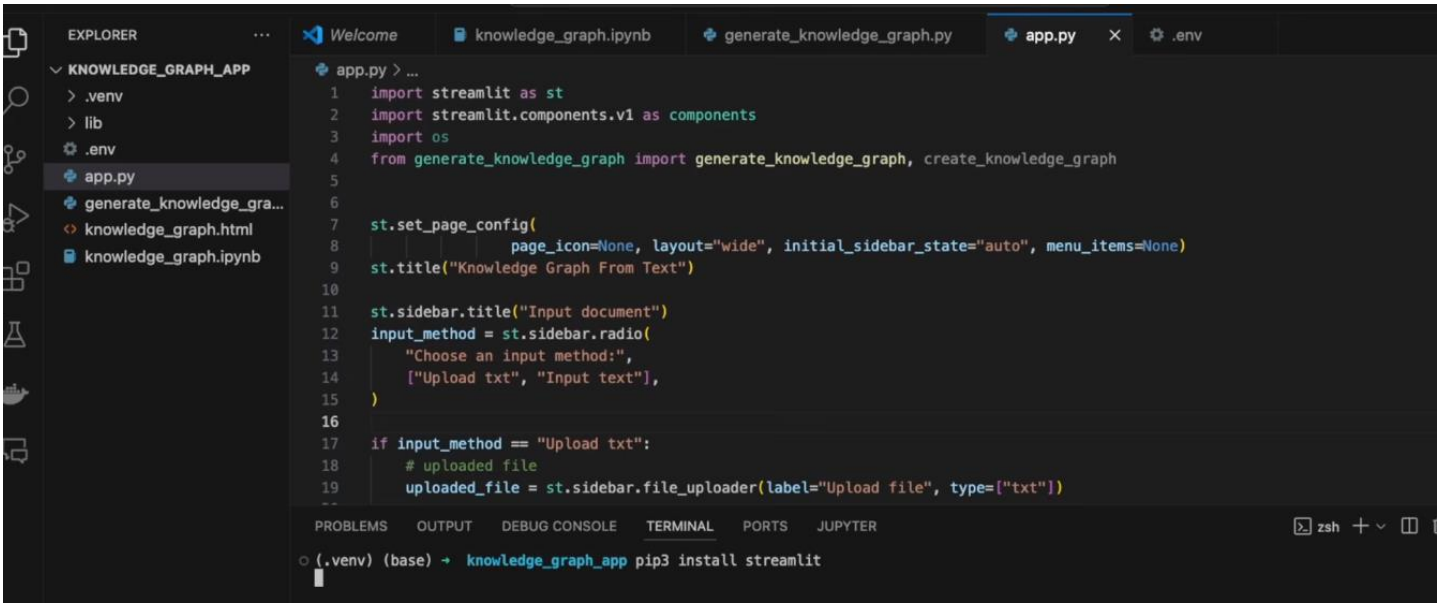

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
.env
generate_knowledge_graph.py > extract_graph_data
api_key = os.getenv("OPENAI_API_KEY")
llm = ChatOpenAI(temperature=0, model_name="gpt-4o")
graph_transformer = LLMGraphTransformer(llm=llm)
# Extract graph data from input text
async def extract_graph_data(text):
"""
Asynchronously extracts graph data from input text using a graph transformer.
Args:
text (str): Input text to be processed into graph format.
Returns:
list: A list of GraphDocument objects containing nodes and relationships.
"""
documents = [Document(page_content=text)]
graph_documents = await graph_transformer.aconvert_to_graph_documents(documents)
return graph_documents
def visualize_graph(graph_documents):
"""
Visualizes a knowledge graph using PyVis based on the extracted graph documents.
Args:
graph_documents (list): A list of GraphDocument objects with nodes and relationships.
Returns:
pyvis.network.Network: The visualized network graph object.
"""
Create network

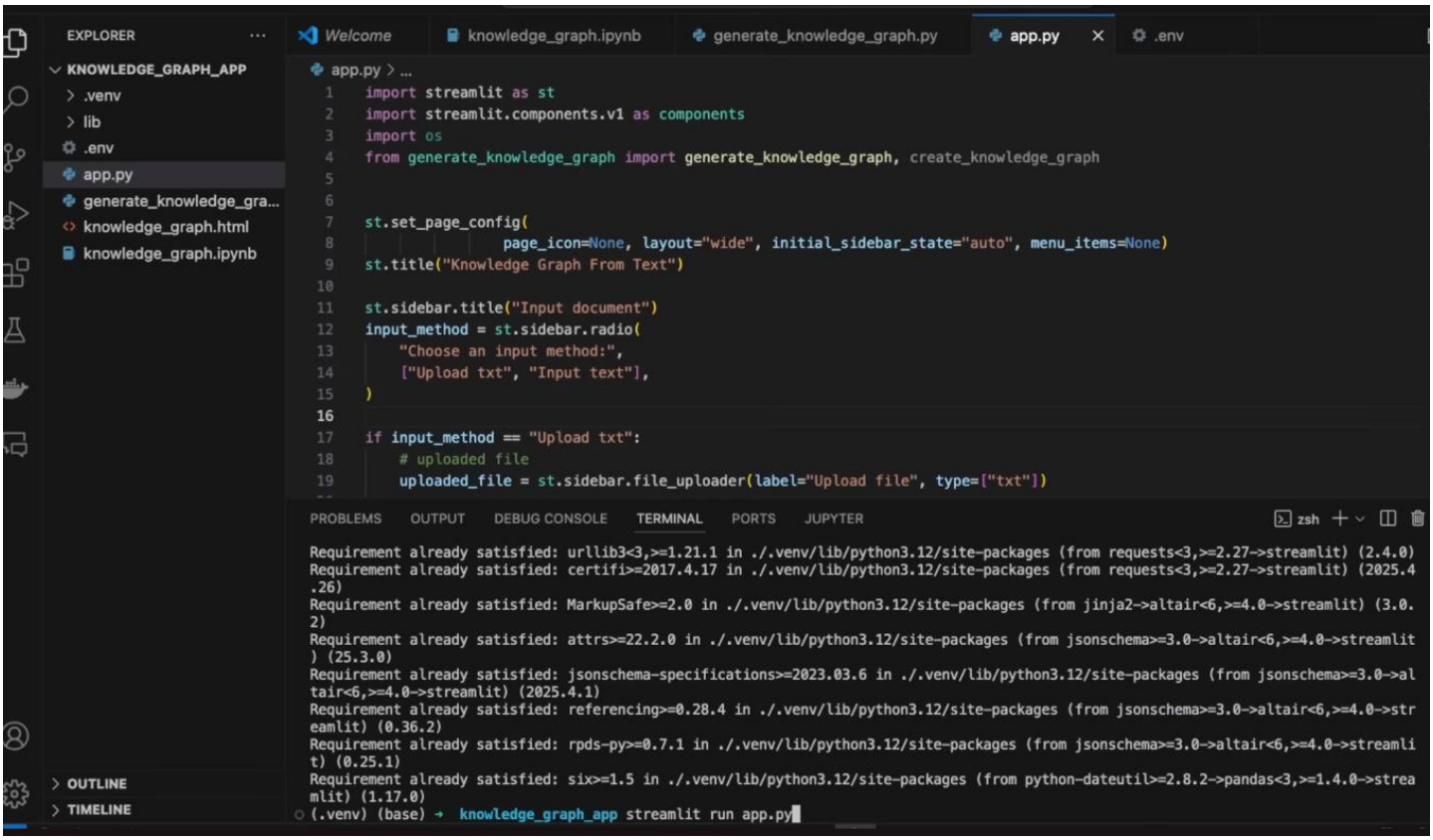


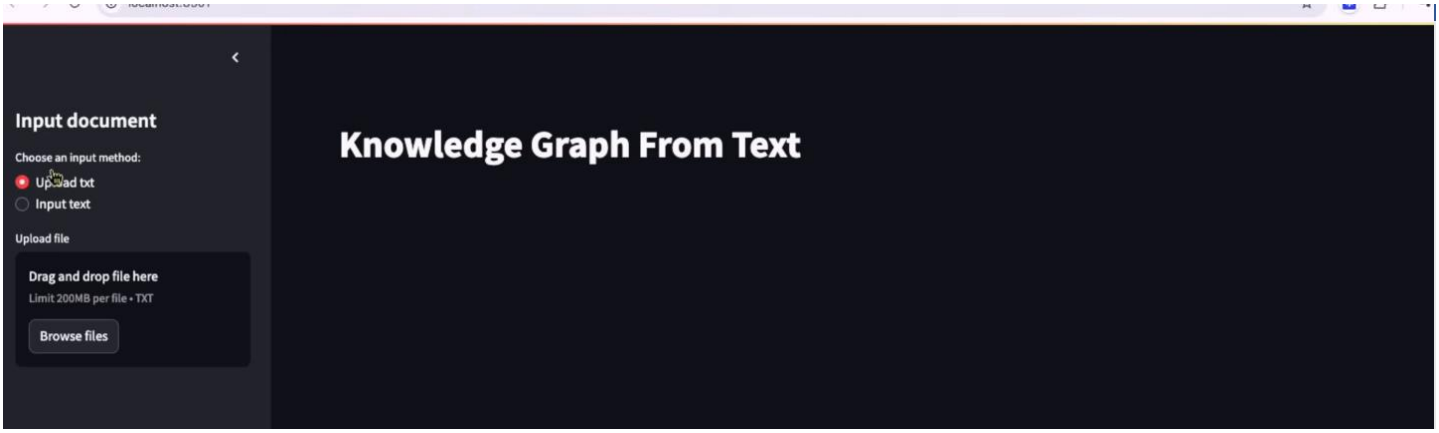


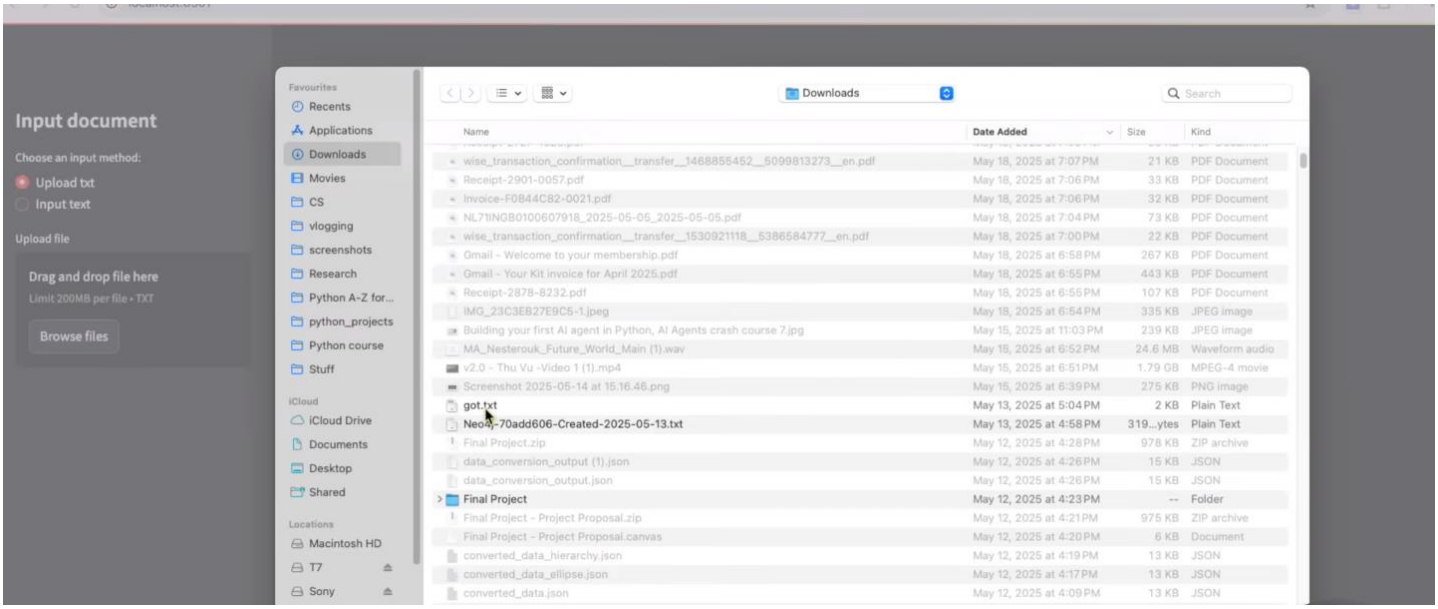
We also create an app.py file for our Streamlit application

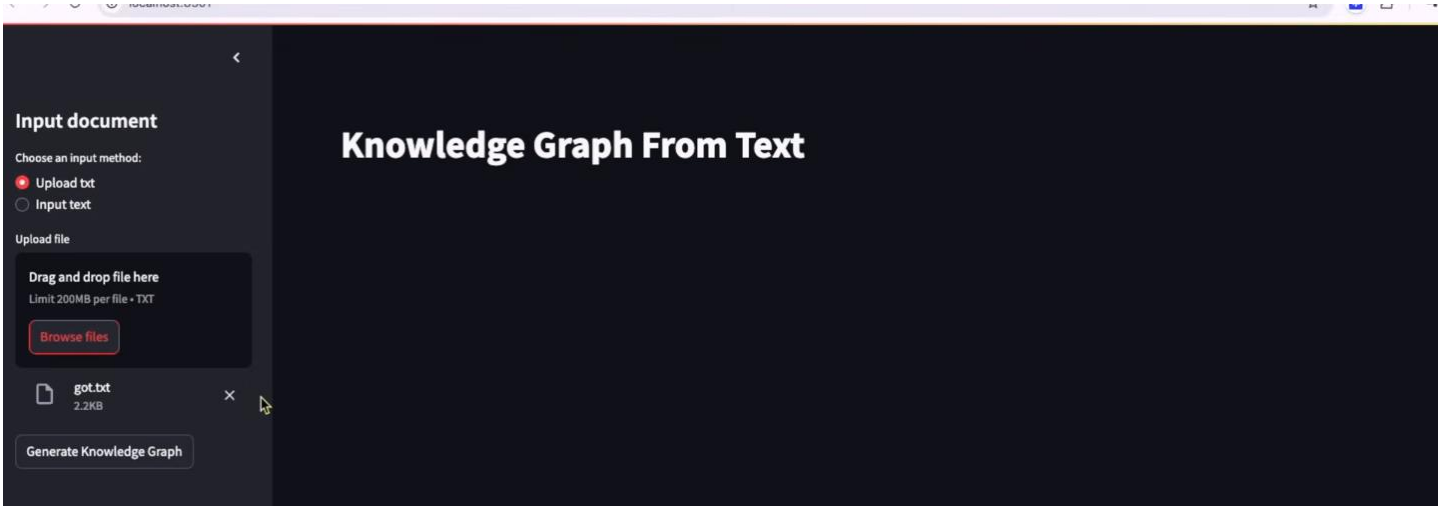

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
app.py
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
OUTLINE
TIMELINE
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
app.py
.env
app.py > ...
import streamlit as st
import streamlit.components.v1 as components
import os
from generate_knowledge_graph import generate_knowledge_graph, create_knowledge_graph
st.set_page_config(
page_icon=None, layout="wide", initial_sidebar_state="auto", menu_items=None)
st.title("Knowledge Graph From Text")
st.sidebar.title("Input document")
input_method = st.sidebar.radio(
"Choose an input method:",
["Upload txt", "Input text"],
)
if input_method == "Upload txt":
# uploaded file
uploaded_file = st.sidebar.file_uploader(label="Upload file", type=["txt"])
if uploaded_file is not None:
# Read the contents of the file
text = uploaded_file.read().decode("utf-8")
# Create and display the graph once the file is uploaded
if st.sidebar.button("Generate Knowledge Graph"):
with st.spinner("Generating knowledge graph..."):
# Call the function to generate the knowledge graph
net = generate_knowledge_graph(text)
st.success("Knowledge graph generated successfully!")
output_file = "knowledge_graph.html"
net.save_graph(output_file)

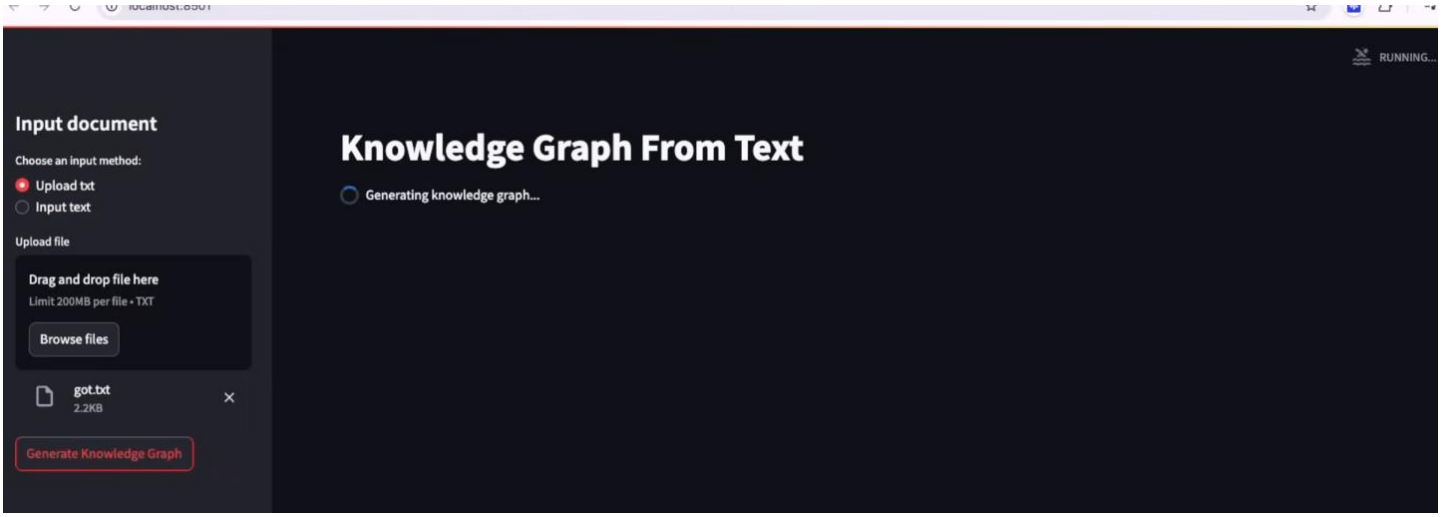

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
app.py
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
app.py
.env
app.py > ...
import streamlit as st
import streamlit.components.v1 as components
import os
from generate_knowledge_graph import generate_knowledge_graph, create_knowledge_graph
st.set_page_config(
page_icon=None, layout="wide", initial_sidebar_state="auto", menu_items=None)
st.title("Knowledge Graph From Text")
st.sidebar.title("Input document")
input_method = st.sidebar.radio(
"Choose an input method:",
["Upload txt", "Input text"],
)
if input_method == "Upload txt":
# uploaded file
uploaded_file = st.sidebar.file_uploader(label="Upload file", type=["txt"])
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
zsh
(.venv) (base) → knowledge_graph_app pip3 install streamlit

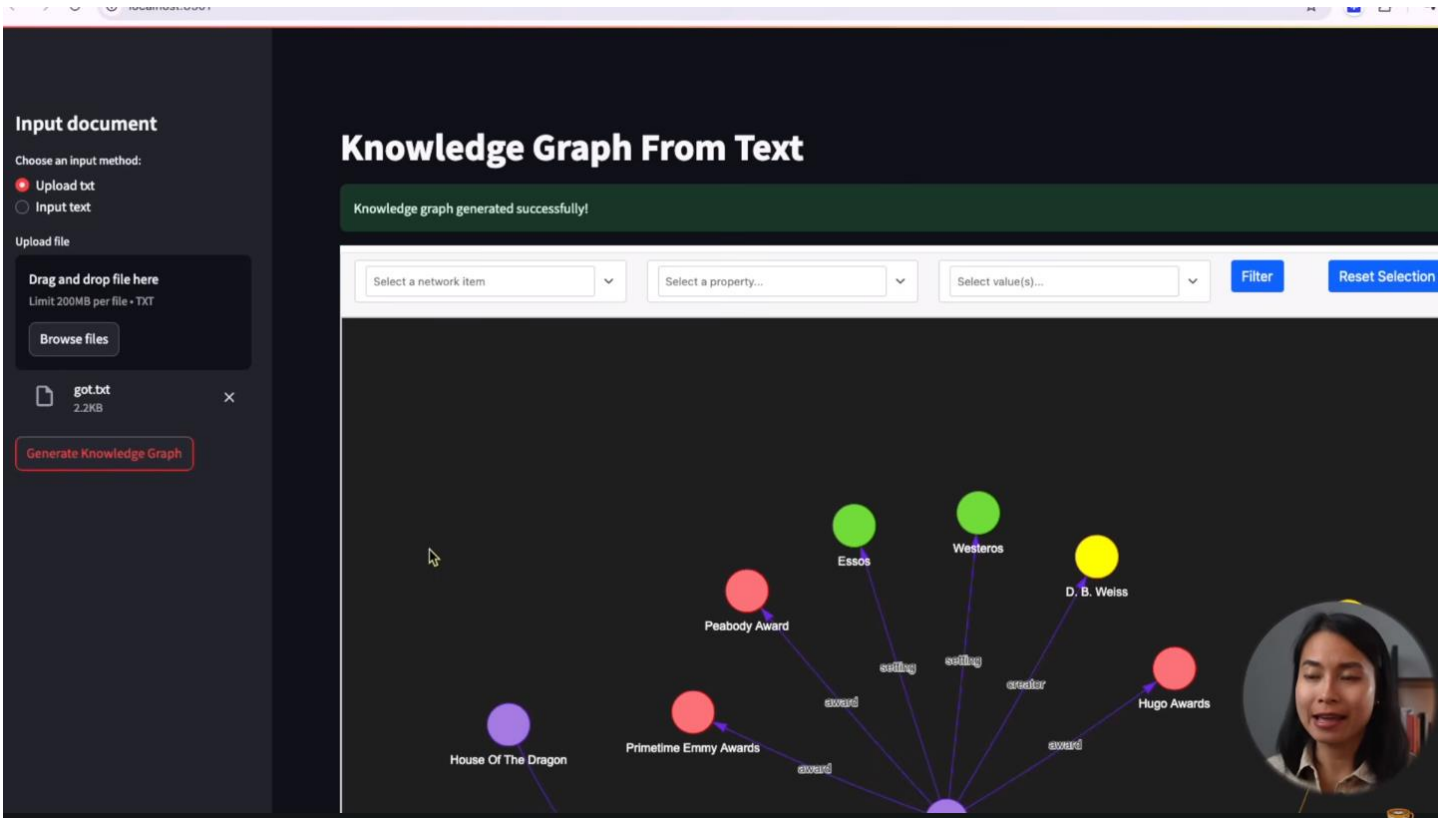

EXPLORER
KNOWLEDGE_GRAPH_APP
.venv
lib
.env
app.py
generate_knowledge_gra...
knowledge_graph.html
knowledge_graph.ipynb
Welcome
knowledge_graph.ipynb
generate_knowledge_graph.py
app.py
.env
app.py > ...
1 import streamlit as st
2 import streamlit.components.v1 as components
3 import os
4 from generate_knowledge_graph import generate_knowledge_graph, create_knowledge_graph
7 st.set_page_config(
8 page_icon=None, layout="wide", initial_sidebar_state="auto", menu_items=None)
9 st.title("Knowledge Graph From Text")
11 st.sidebar.title("Input document")
12 input_method = st.sidebar.radio(
13 "Choose an input method:",
14 ["Upload txt", "Input text"],
15)
17 if input_method == "Upload txt":
18 # uploaded file
19 uploaded_file = st.sidebar.file_uploader(label="Upload file", type=["txt"])
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
zsh
Requirement already satisfied: urllib3<3,>=1.21.1 in ./.venv/lib/python3.12/site-packages (from requests<3,>=2.27->streamlit) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in ./.venv/lib/python3.12/site-packages (from requests<3,>=2.27->streamlit) (2025.4.26)
Requirement already satisfied: MarkupSafe>=2.0 in ./.venv/lib/python3.12/site-packages (from jinja2->altair<6,>=4.0->streamlit) (3.0.2)
Requirement already satisfied: attrs>=22.2.0 in ./.venv/lib/python3.12/site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in ./.venv/lib/python3.12/site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2025.4.1)
Requirement already satisfied: referencing>=0.28.4 in ./.venv/lib/python3.12/site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in ./.venv/lib/python3.12/site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.25.1)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.12/site-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.17.0)
(.venv) (base) → knowledge_graph_app streamlit run app.py
OUTLINE
TIMELINE


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
Knowledge Graph From Text

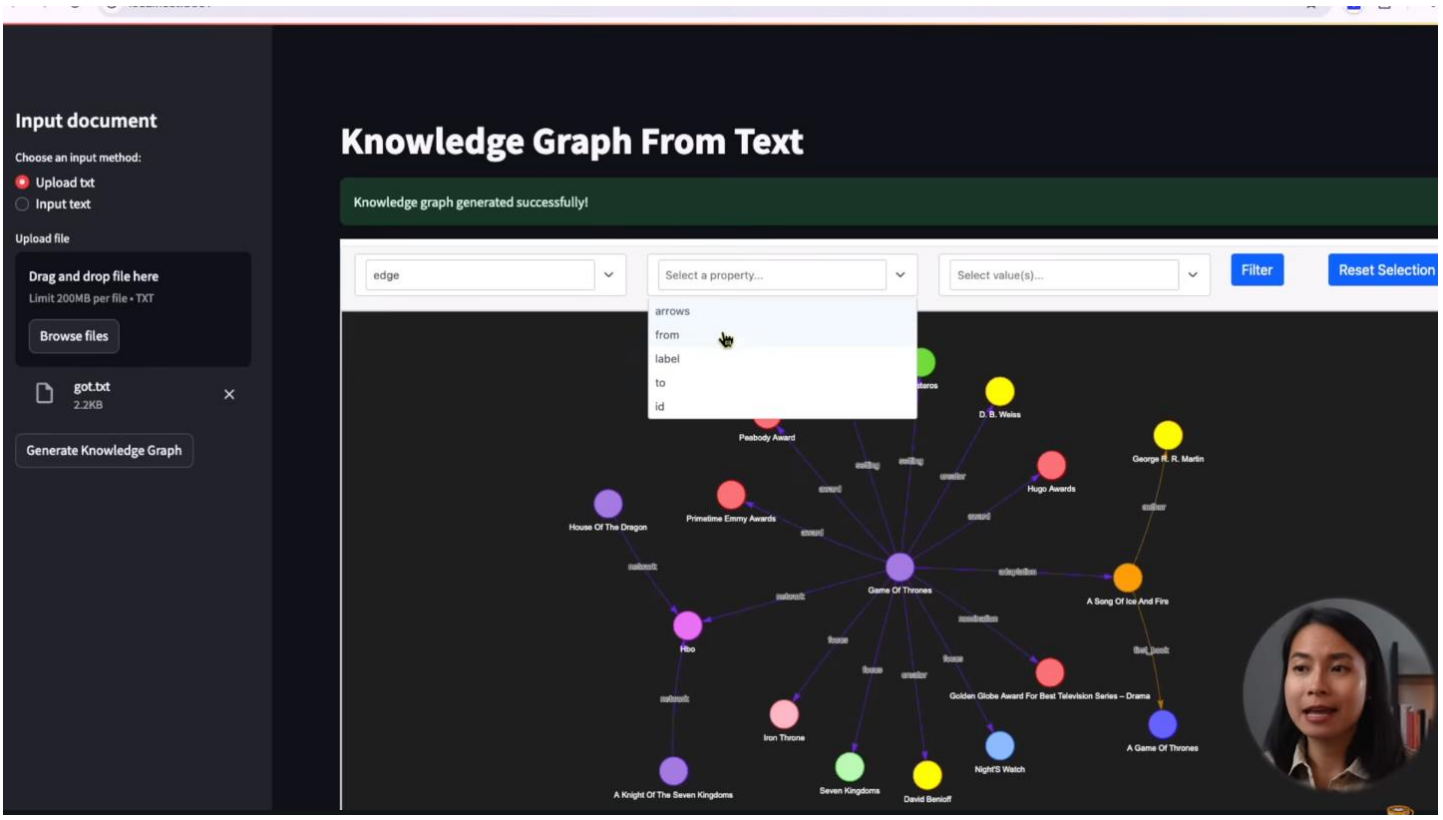

Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
Favourites
Recents
Applications
Downloads
Movies
CS
vlogging
screenshots
Research
Python A-Z for...
python_projects
Python course
Stuff
iCloud
iCloud Drive
Documents
Desktop
Shared
Locations
Macintosh HD
T7
Sony
Downloads
Search
Name
Date Added
Size
Kind
got.txt
May 13, 2025 at 5:04 PM
2 KB
Plain Text
Neo4j-70add606-Created-2025-05-13.txt
May 13, 2025 at 4:58 PM
Plain Text
Final Project
May 12, 2025 at 4:23 PM
Folder

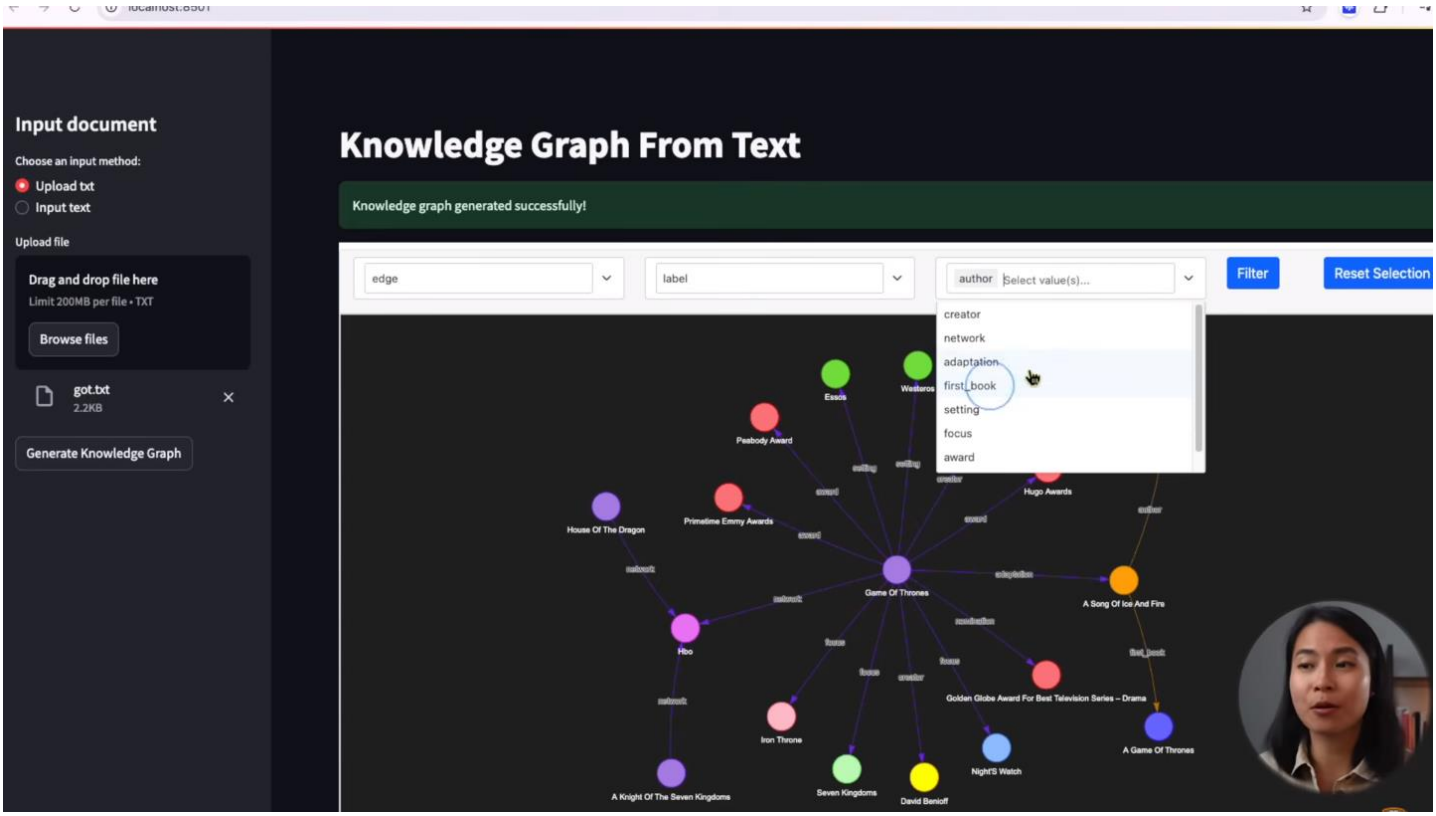

Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text

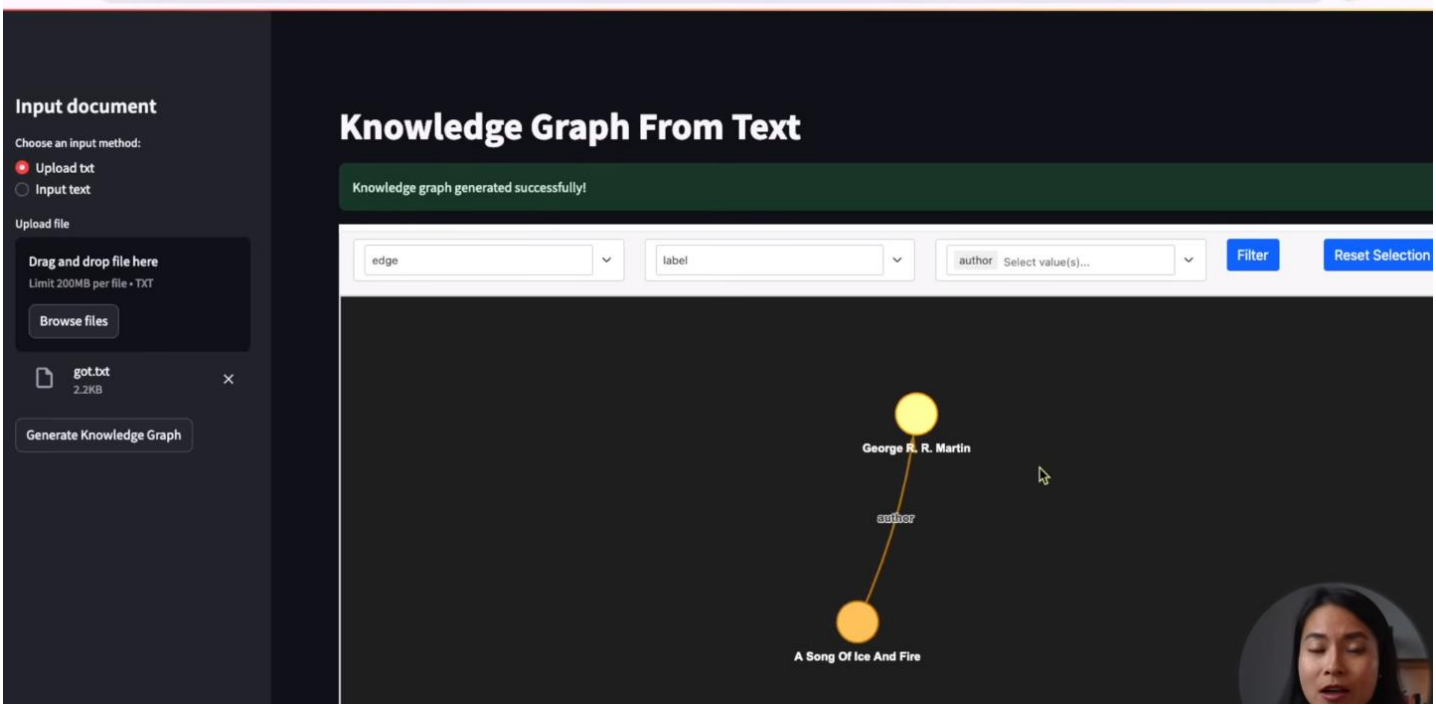

RUNNING...
Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text
Generating knowledge graph...


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text
Knowledge graph generated successfully!
Select a network item
Select a property...
Select value(s)...
Filter
Reset Selection
Essos
Westeros
D. B. Weiss
Peabody Award
Hugo Awards
House Of The Dragon
Primetime Emmy Awards
award
writing
creator


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Essos
Westeros
D. B. Weiss
Peabody Award
George R. R. Martin
Hugo Awards
Primetime Emmy Awards
House Of The Dragon
Game Of Thrones
A Song Of Ice And Fire
Hbo
Golden Globe Award For Best Television Series – Drama
Iron Throne
Night'S Watch
A Knight Of The Seven Kingdoms
Seven Kingdoms
David Benioff
selling
creator
award
author
network
adaptation
focus
first_book


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text
Knowledge graph generated successfully!
edge
Select a property...
Select value(s)...
Filter
Reset Selection
arrows
from
label
to
id
Peabody Award
D. B. Weiss
George R. R. Martin
Hugo Awards
Primetime Emmy Awards
House Of The Dragon
Game Of Thrones
A Song Of Ice And Fire
Hbo
Golden Globe Award For Best Television Series – Drama
Iron Throne
A Game Of Thrones
Night'S Watch
A Knight Of The Seven Kingdoms
Seven Kingdoms
David Benioff


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text
Knowledge graph generated successfully!
edge
label
author Select value(s)...
Filter
Reset Selection
creator
network
adaptation
first_book
setting
focus
award
Essos
Westeros
Peabody Award
Hugo Awards
Primetime Emmy Awards
House Of The Dragon
Game Of Throne
A Song Of Ice And Fire
Hbo
Golden Globe Award For Best Television Series – Drama
Iron Throne
A Game Of Thrones
Night'S Watch
A Knight Of The Seven Kingdoms
Seven Kingdoms
David Benioff


Input document
Choose an input method:
Upload txt
Input text
Upload file
Drag and drop file here
Limit 200MB per file • TXT
Browse files
got.txt
2.2KB
Generate Knowledge Graph
Knowledge Graph From Text
Knowledge graph generated successfully!
edge
label
author Select value(s)...
Filter
Reset Selection
George R. R. Martin
author
A Song Of Ice And Fire