Member-only story

Graph Similar Sentences With NetworkX

See how sentences relate to each other

Mary Paskhaver · Following

Published in Better Programming · 4 min read · Mar 22

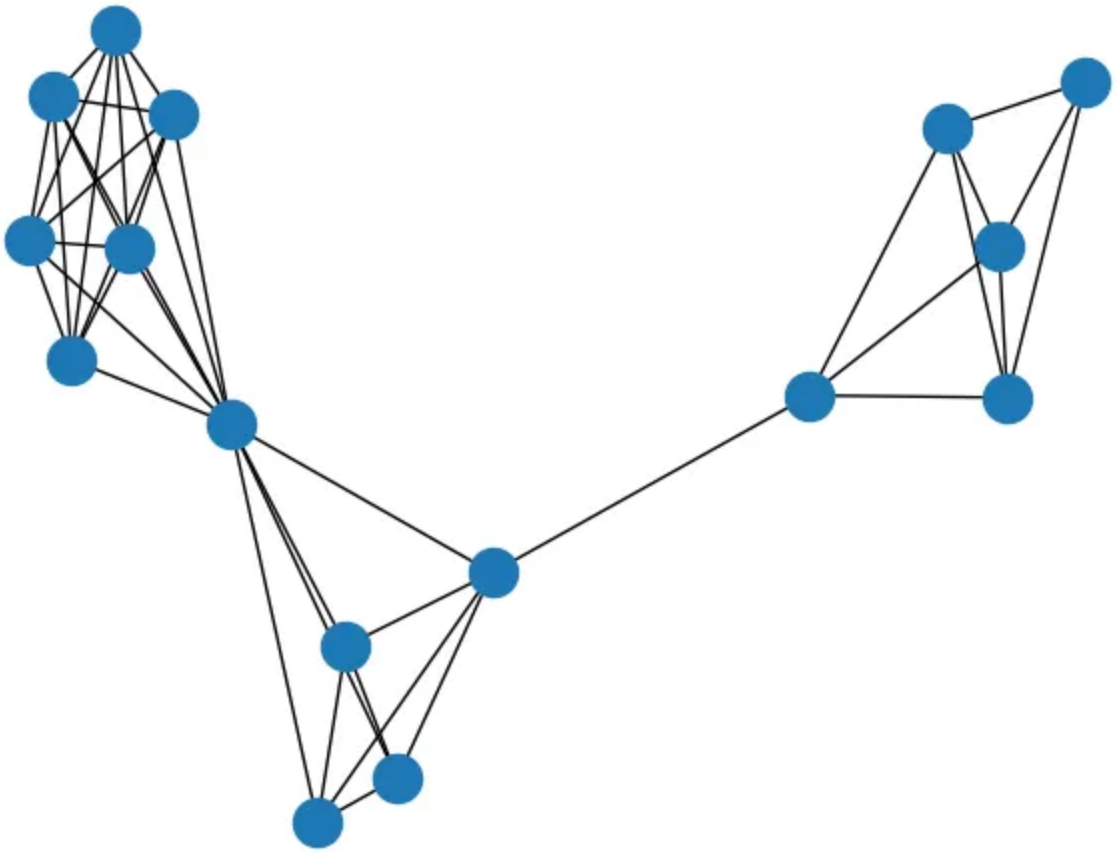


Image by author using pyplot.

For my research, I calculated how similar sentences were to each other.

This process involves transforming each sentence into a vector: a construct with direction and magnitude. Then, we take the cosine of the angle between the vectors. The closer the cosine is to one, the more similar the sentences are.

I created a 2D array showing each sentence's cosine similarity to another. But I had trouble seeing something meaningful in a sea of numbers.

I converted this 2D array to a graph and plotted it with NetworkX, a Python library. That way, I could easily see clusters of similar sentences.

Here's how you can do that.

Install Dependencies

If you don't have Python on your computer, install the latest version from Python's website. If you have trouble with that, try this tutorial.

For this project, you will need to install these libraries:

- matplotlib
- NetworkX
- sklearn

To do this, activate your virtual environment and run this command:

```
pip install matplotlib networkx scikit-learn
```

If you get a ModuleNotFoundError, your modules are probably installed in the wrong place. You may have many versions of Python and pip installed, so the wrong ones were used.

To fix this, identify where you need to install dependencies. Run this code:

```
import sys
print(sys.executable)
```

Copy the output. It might look something like `/Users/your_name/your_folder/your_virtual_environment/bin/python`.

Then, run this command. Replace the first part with what you just copied and install the necessary modules.

```
/Users/your_name/your_folder/your_virtual_environment/bin/python -m pip install
```

Set Up To Build the Graph

Here's the gist of it:

- Each node in this graph will represent a sentence.

- Each edge in this graph will connect two sentences with a high cosine similarity.

To build this graph, you'll need an array of sentences. Let's declare them in `main.py`:

```
sentences = [
    "My dog is my best friend.",
    "My dog is my best pal.",
    "My dog is the best.",
    "And now, for something completely different.",
]
```

We want to vectorize these sentences before taking their cosine similarities. To do this, we'll use sklearn's `TfidfVectorizer` class. This will assign our words TF-IDF values.

A word's term frequency-inverse document frequency, or TF-IDF, shows its importance in a bunch of text. Common words have low TF-IDF values because they tell us less about a sentence's meaning.

Let's get a matrix with our sentences' TF-IDF features (important words). At the top of `main.py`, import `TfidfVectorizer`:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

Now, let's define a function, `get_tfidf_matrix`, to get that matrix from an array of sentences:

```
def get_tfidf_matrix(sentences):
    tfidf_vectorizer = TfidfVectorizer(stop_words="english")
    return tfidf_vectorizer.fit_transform(sentences)
```

Stop words are unimportant words, like "um," "a," or "the." Notice we created a `TfidfVectorizer` with the argument `stop_words="english"`. This ensures we don't consider those words as features.

We'll use our TF-IDF matrix to get each sentence's cosine similarity to each other sentence. To do that, we'll use sklearn's `cosine_similarity` function. Add this to the top of `main.py`:

```
from sklearn.metrics.pairwise import cosine_similarity
```

Then, define this method:

```
def get_cosine_similarity_matrix():
    tfidf_matrix = get_tfidf_matrix(sentences)
    return cosine_similarity(tfidf_matrix)
```

Now we're ready to build our graph.

Build the Graph

Add these two imports to the top of `main.py`:

```
import matplotlib.pyplot as plt
import networkx as nx
```

Let's put all our graph logic into a new function. We'll call it `graph_sentences_by_similarity`. This function will do the following:

1. Take in an array of sentences as a parameter.
2. Create a graph with a node for each sentence.
3. Add an edge between two nodes with a cosine similarity greater than 0.5.
4. Remove nodes without edges.
5. Plot the graph.

```
def graph_sentences_by_similarity(sentences):
    # Create a graph. Each node represents a sentence.
    G = nx.Graph()
    G.add_nodes_from(sentences)

    cosine_similarity_matrix = get_cosine_similarity_matrix()

    # Add an edge between two sentences whose similarity is > 0.5.
    for row in range(0, len(cosine_similarity_matrix)):
        for col in range(0, len(cosine_similarity_matrix[0])):
            if row != col and cosine_similarity_matrix[row][col] > 0.5:
                G.add_edge(sentences[row], sentences[col])

    # Remove nodes that don't have any edges (i.e. dissimilar sentences).
    G.remove_nodes_from(list(nx.isolates(G)))
```

```
    nx.draw(
        G,
        with_labels=True,
    )
    plt.show()
```

We can invoke the main method. Add these lines to `main.py`:

```
def main():
    graph_sentences_by_similarity(sentences)

if __name__ == "__main__":
    main()
```

Now, in your terminal, activate your virtual environment. Run this command:

```
python main.py
```

You should see something like this:

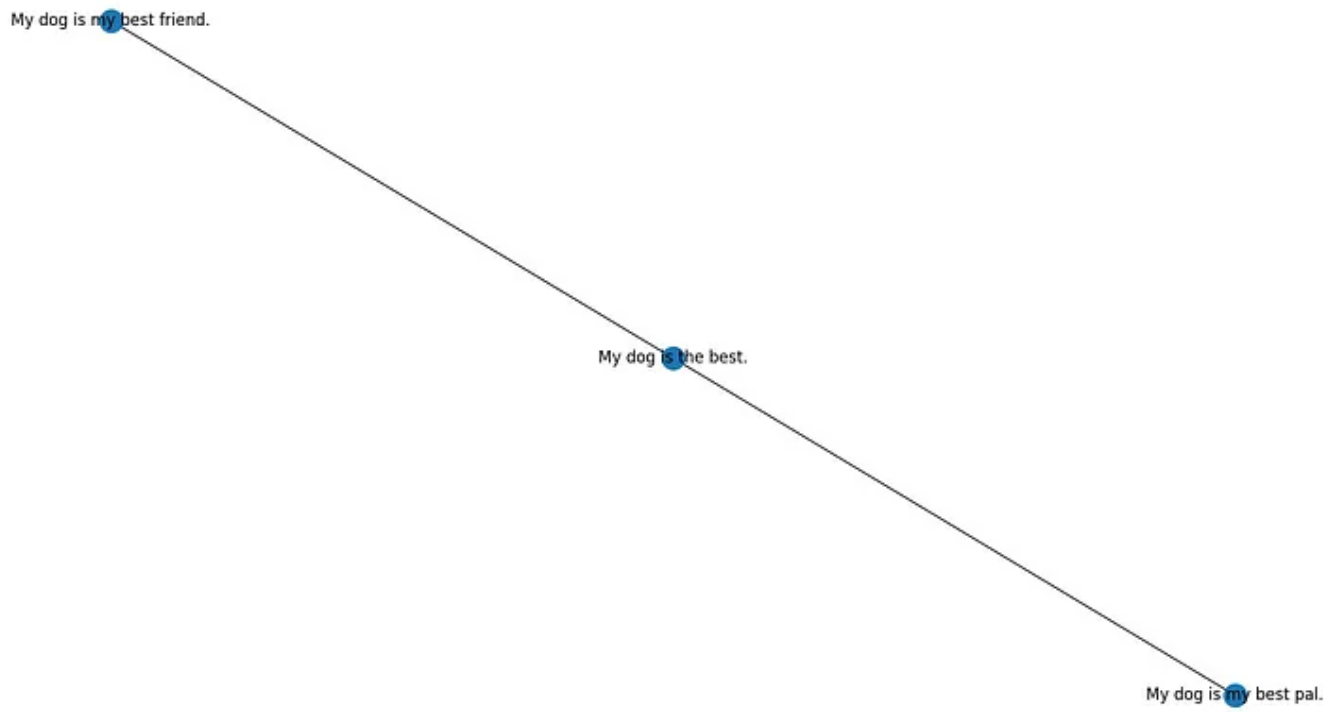


Image by author using pyplot.

Conclusion

Woo-hoo! We have a graph that shows us which sentences are similar. Notice that even though we had four sentences in our array, only three appear on the graph. The last sentence, “And now, for something completely different.” was not similar enough to the other three to be drawn.

There are a couple of ways we can customize our graph. For example, we can change our nodes’ color, labels, and sizes. Check out some examples here.

Thanks for reading. Have a great day!

Resources

Here is my main.py file.

Python NLP Research Programming Coding

More from the list: "NLP"

Curated by Himanshu Birla

Jon Gi... in Towards Data ...

Characteristics of Word Embeddings

· 11 min read · Sep 4, 2021

Jon Gi... in Towards Data ...

The Word2vec Hyperparameters

· 6 min read · Sep 3, 2021

Jon Gi... in

The Word2ve

· 15 min rea

View list

Written by Mary Paskhaver

Following

127 Followers · Writer for Better Programming

Student and mobile app developer. Follow me for tips on tech, work, and more.

More from Mary Paskhaver and Better Programming

Mary Paskhaver

Remove Unused Dependencies From a React Native Project

Learn how to clean up your code

· 3 min read · Jan 2

80

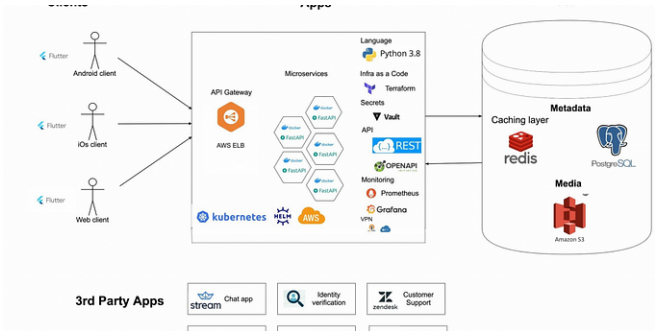
Vinita in Better Programming

How To Disagree With Someone More Powerful Than You

Disagreements can save time and energy by preventing critical mistakes and course...

· 9 min read · Sep 19

1.9K 30



Dmitry Kruglov in Better Programming

The Architecture of a Modern Startup

Hype wave, pragmatic evidence vs the need to move fast

16 min read · Nov 7, 2022

6.1K 59

Mary Paskhaver

Generate Subtitles with Python

Transcribe videos and audio with the free Autosub library

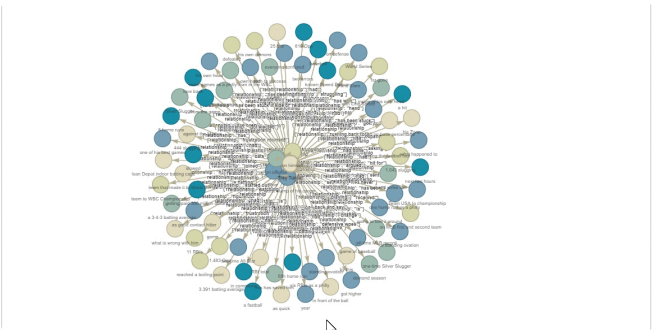
· 4 min read · Jan 18

6 1

See all from Mary Paskhaver

See all from Better Programming

Recommended from Medium

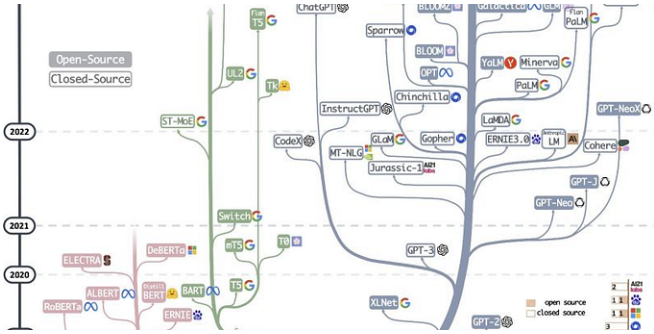


Wenqi Glantz in Better Programming

7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

· 17 min read · 4 days ago



Haifeng Li

A Tutorial on LLM

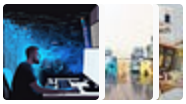
Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14

Lists

Coding & Development

11 stories · 200 saves

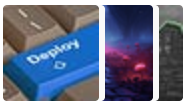


It's never too late or early to start something

15 stories · 145 saves

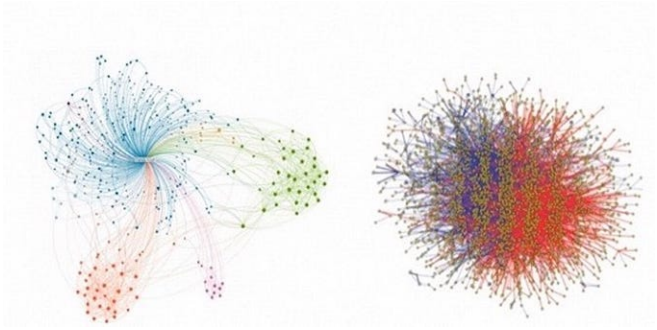
General Coding Knowledge

20 stories · 402 saves



Predictive Modeling w/ Python

20 stories · 452 saves



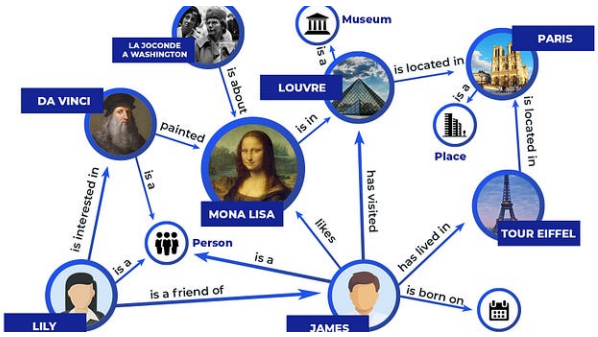
Lei Wang in graphscope

Graphs and Graph Applications

In this post, we will introduce basic concepts of graphs, and some typical applications of...

7 min read · May 22

7



Alla Chepurova in DeepPavlov

Improving Knowledge Graph Completion with Generative LM...

Combining both internal LM knowledge and external data from KG

13 min read · Sep 5

36

Ajay Verma

Knowledge Graph from Text Data

How to generate knowledge graph for any text data

Will Lockett in Predict

China May Have Just Condemned The Entire World

A crucial shift in rhetoric from the superpower is deeply worrying.

2 min read · Sep 18

63

✦ · 7 min read · Sep 26

3.3K 84

See more recommendations