Search Medium Write

Member-only story

GETTING STARTED

Text analysis basics in Python

Bigram/trigram, sentiment analysis, and topic modeling

Sophia Yang, Ph.D. · Following
Published in Towards Data Science · 5 min read · Oct 20, 2020

248

Source: https://unsplash.com/photos/uGP_6CAD-14

This article talks about the most basic text analysis tools in Python. We are not going into the fancy NLP models. Just the basics. Sometimes all you need is the basics :)

Let's first get some text data. Here we have a list of course reviews that I made up. What can we do with this data? The first question that comes to mind is can we tell which reviews are positive and which are negative? Can we do some sentiment analysis on these reviews?

```
corpus = [
'Great course. Love the professor.',
'Great content. Textbook was great',
'This course has very hard assignments. Great content.',
'Love the professor.',
'Hard assignments though',
'Hard to understand.'
]
```

Sentiment analysis

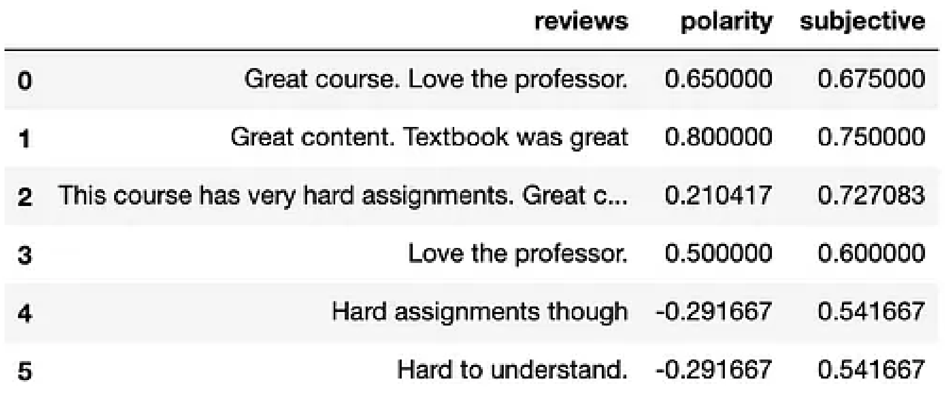
Great, let's look at the overall sentiment analysis. I like to work with a pandas data frame. So let's create a pandas data frame from the list.

```
import pandas as pd
df = pd.DataFrame(corpus)
df.columns = ['reviews']
```

Next, let's install the library *textblob* (`conda install textblob -c conda-forge`) and import the library.

```
from textblob import TextBlob
df['polarity'] = df['reviews'].apply(lambda x: TextBlob(x).polarity)
df['subjective'] = df['reviews'].apply(lambda x:
TextBlob(x).subjectivity)
```

We then can calculate the sentiment through the `polarity` function. `polarity` ranges from -1 to 1, with -1 being negative and 1 being positive. The TextBlob can also use the `subjectivity` function to calculate `subjectivity`, which ranges from 0 to 1, with 0 being objective and 1 being subjective.



	reviews	polarity	subjective
0	Great course. Love the professor.	0.650000	0.675000
1	Great content. Textbook was great	0.800000	0.750000
2	This course has very hard assignments. Great c...	0.210417	0.727083
3	Love the professor.	0.500000	0.600000
4	Hard assignments though	-0.291667	0.541667
5	Hard to understand.	-0.291667	0.541667

Sentiment analysis of Bigram/Trigram

Next, we can explore some word associations. N-grams analyses are often used to see which words often show up together. I often like to investigate combinations of two words or three words, i.e., Bigrams/Trigrams.

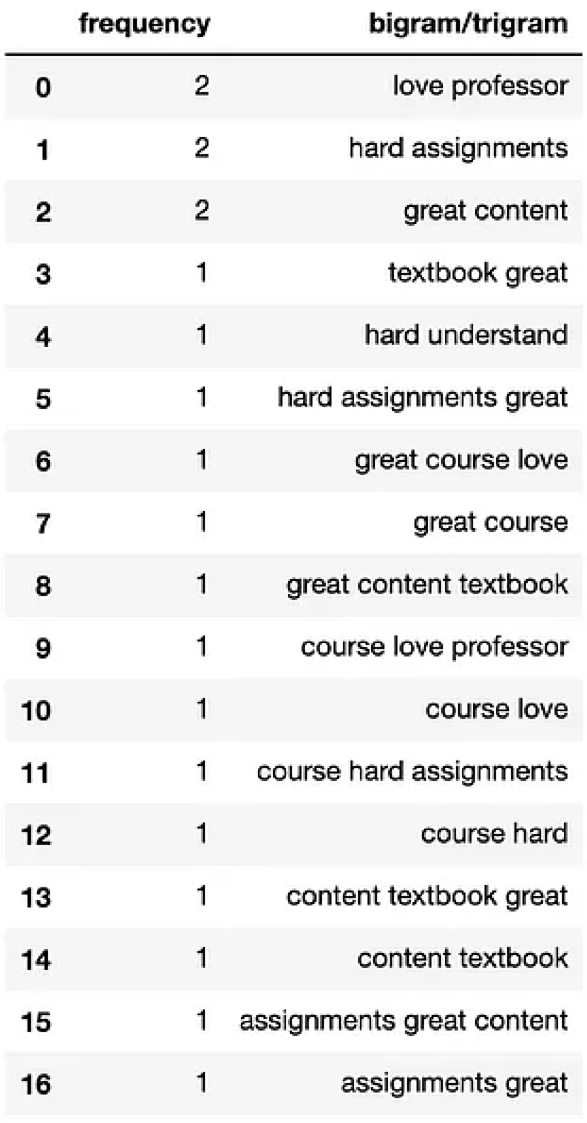
> *An* **n-gram** *is a contiguous sequence of* n *items from a given sample of text or speech.*

In the text analysis, it is often a good practice to filter out some stop words, which are the most common words but do not have significant contextual meaning in a sentence (e.g., “a”, “ the”, “and”, “but”, and so on). *nltk* provides us a list of such stopwords. We can also add customized stopwords to the list. For example, here we added the word "though".

```
from nltk.corpus import stopwords
stoplist = stopwords.words('english') + ['though']
```

Now we can remove the stop words and work with some bigrams/trigrams. The function `CountVectorizer` "convert a collection of text documents to a matrix of token counts". The `stop_words` parameter has a build-in option "english". But we can also use our user-defined stopwords like I am showing here. The `ngram_range` parameter defines which n-grams are we interested in — 2 means bigram and 3 means trigram. The other parameter worth mentioning is `lowercase` , which has a default value *True* and converts all characters to lowercase automatically for us. Now with the following code, we can get all the bigrams/trigrams and sort by frequencies.

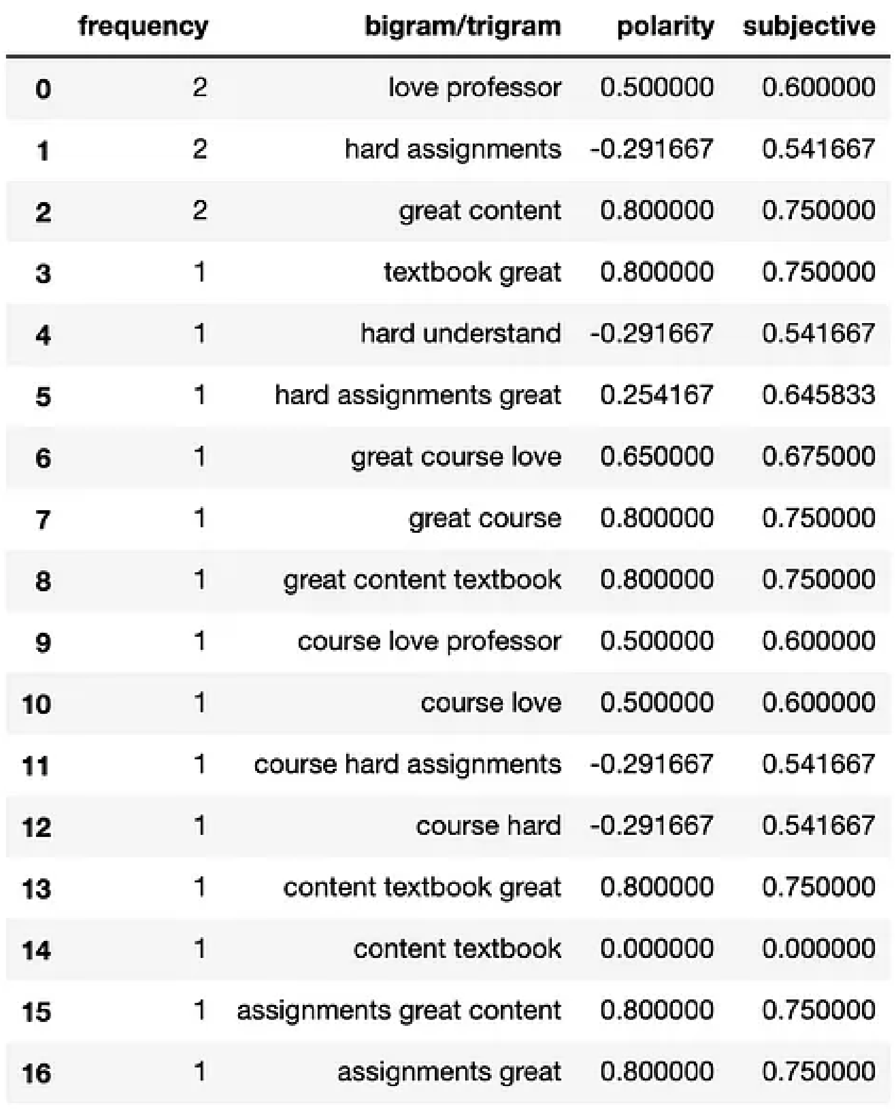
```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(stop_words=stoplist, ngram_range=(2,3))
# matrix of ngrams
ngrams = c_vec.fit_transform(df['reviews'])
# count frequency of ngrams
count_values = ngrams.toarray().sum(axis=0)
# list of ngrams
vocab = c_vec.vocabulary_

df_ngram = pd.DataFrame(sorted([(count_values[i],k) for k,i in
vocab.items()], reverse=True)
            ).rename(columns={0: 'frequency', 1:'bigram/trigram'})
```



	frequency	bigram/trigram
0	2	love professor
1	2	hard assignments
2	2	great content
3	1	textbook great
4	1	hard understand
5	1	hard assignments great
6	1	great course love
7	1	great course
8	1	great content textbook
9	1	course love professor
10	1	course love
11	1	course hard assignments
12	1	course hard
13	1	content textbook great
14	1	content textbook
15	1	assignments great content
16	1	assignments great

Similar to the sentiment analysis before, we can calculate the polarity and subjectivity for each bigram/trigram.

```
df_ngram['polarity'] = df_ngram['bigram/trigram'].apply(lambda x:
TextBlob(x).polarity)
df_ngram['subjective'] = df_ngram['bigram/trigram'].apply(lambda x:
TextBlob(x).subjectivity)
```



	frequency	bigram/trigram	polarity	subjective
0	2	love professor	0.500000	0.600000
1	2	hard assignments	-0.291667	0.541667
2	2	great content	0.800000	0.750000
3	1	textbook great	0.800000	0.750000
4	1	hard understand	-0.291667	0.541667
5	1	hard assignments great	0.254167	0.645833
6	1	great course love	0.650000	0.675000
7	1	great course	0.800000	0.750000
8	1	great content textbook	0.800000	0.750000
9	1	course love professor	0.500000	0.600000
10	1	course love	0.500000	0.600000
11	1	course hard assignments	-0.291667	0.541667
12	1	course hard	-0.291667	0.541667
13	1	content textbook great	0.800000	0.750000
14	1	content textbook	0.000000	0.000000
15	1	assignments great content	0.800000	0.750000
16	1	assignments great	0.800000	0.750000

Topic modeling

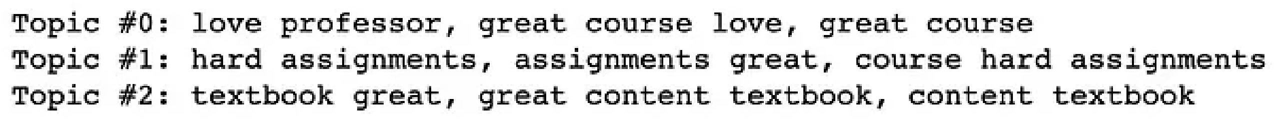
We can also do some topic modeling with text data. There are two ways to do this: NMF models and LDA models. We will show examples using both methods next.

NMF models

Non-Negative Matrix Factorization (NMF) is a matrix decomposition method, which decomposes a matrix into the product of W and H of non-negative elements. The default method optimizes the distance between the original matrix and WH, i.e., the Frobenius norm. Below is an example where we use NMF to produce 3 topics and we showed 3 bigrams/trigrams in each topic.

```
Source: https://scikit-
learn.org/stable/auto_examples/applications/plot_topics_extraction_wi
th_nmf_lda.html

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF
from sklearn.pipeline import make_pipeline

tfidf_vectorizer = TfidfVectorizer(stop_words=stoplist, ngram_range=
(2,3))
nmf = NMF(n_components=3)
pipe = make_pipeline(tfidf_vectorizer, nmf)
pipe.fit(df['reviews'])

def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += ", ".join([feature_names[i]
                              for i in topic.argsort()[:-n_top_words -
1:-1]])
        print(message)
    print()

print_top_words(nmf, tfidf_vectorizer.get_feature_names_out(),
n_top_words=3)
```

Here is the result. Looks like topic 0 is about the professor and courses; topic 1 is about the assignment, and topic 3 is about the textbook. Note that we do not know what is the best number of topics here. We used 3 just because our sample size is very small. In practice, you might need to do a grid search to find the optimal number of topics.



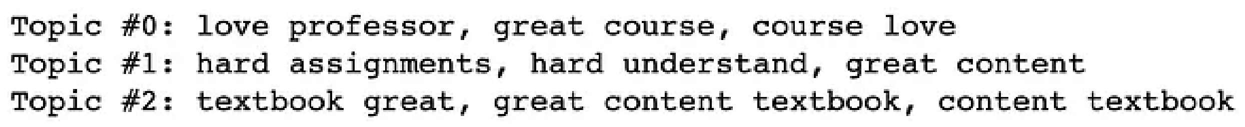
```
Topic #0: love professor, great course love, great course
Topic #1: hard assignments, assignments great, course hard assignments
Topic #2: textbook great, great content textbook, content textbook
```

LDA models

> *Latent Dirichlet Allocation is a generative probabilistic model for collections of discrete dataset such as text corpora. It is also a topic model that is used for discovering abstract topics from a collection of documents.*

Here in our example, we use the function ***`LatentDirichletAllocation`***, which "implements the online variational Bayes algorithm and supports both online and batch update methods". Here we show an example where the learning method is set to the default value "online".

```
Source: https://scikit-
learn.org/stable/auto_examples/applications/plot_topics_extraction_wi
th_nmf_lda.html

from sklearn.decomposition import LatentDirichletAllocation
tfidf_vectorizer = TfidfVectorizer(stop_words=stoplist, ngram_range=
(2,3))
lda = LatentDirichletAllocation(n_components=3)
pipe = make_pipeline(tfidf_vectorizer, lda)
pipe.fit(df['reviews'])

def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += ", ".join([feature_names[i]
                              for i in topic.argsort()[:-n_top_words -
1:-1]])
        print(message)
    print()

print_top_words(lda, tfidf_vectorizer.get_feature_names_out(),
n_top_words=3)
```



```
Topic #0: love professor, great course, course love
Topic #1: hard assignments, hard understand, great content
Topic #2: textbook great, great content textbook, content textbook
```

Now you know how to do some basic text analysis in Python. Our example has very limited data sizes for demonstration purposes. The text analysis in real-world will be a lot more challenging and fun. Hope you enjoy this article. Thanks!

References

https://scikit-learn.org/stable/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://stackoverflow.com/questions/11763613/python-list-of-ngrams-with-frequencies/11834518

Text Analysis Python Topic Modeling Sentiment Analysis Getting Started

Characteristics of Word Embeddings

✦ · 11 min read · Sep 4, 2021

The Word2vec Hyperparameters

✦ · 6 min read · Sep 3, 2021

The Word2ve

✦ · 15 min rea

View list

Written by Sophia Yang, Ph.D.

Following

4.92K Followers · Writer for Towards Data Science

💛 LinkedIn: www.linkedin.com/in/sophiamyang 🐦 Twitter: twitter.com/sophiamyang ✨✨✨ ✨ 📽️ YouTube: youtube.com/SophiaYangDS ✨✨✨📚Book Club: dsbookclub.github.io

More from Sophia Yang, Ph.D. and Towards Data Science

Sophia Yang, Ph.D. in Towards Data Science

4 Ways to Do Question Answering in LangChain

Antonis Makropoulos in Towards Data Science

How to Build a Multi-GPU System for Deep Learning in 2023

Chat with your long PDF docs: load_qa_chain, RetrievalQA, VectorstoreIndexCreator,...

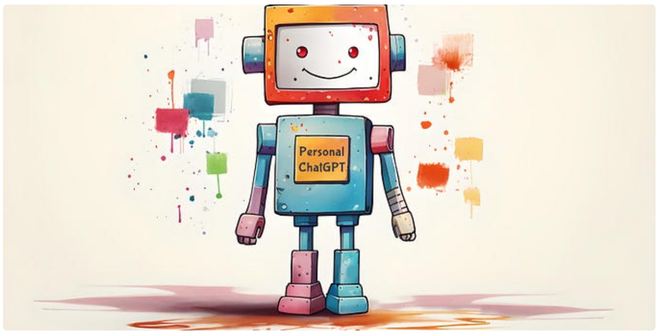
✦ · 6 min read · Apr 8

723 18

This story provides a guide on how to build a multi-GPU system for deep learning and...

10 min read · Sep 17

549 11



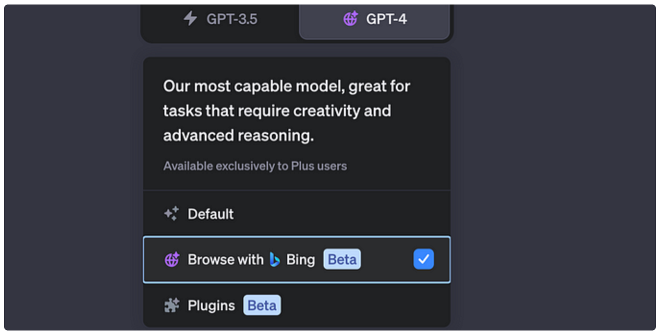
Robert A. Gonsalves in Towards Data Science

Your Own Personal ChatGPT

How you can fine-tune OpenAI's GPT-3.5 Turbo model to perform new tasks using you...

✦ · 15 min read · Sep 8

595 7


GPT-3.5
GPT-4
Our most capable model, great for tasks that require creativity and advanced reasoning.
Available exclusively to Plus users
Default
Browse with Bing Beta
Plugins Beta

Sophia Yang, Ph.D. in Towards Data Science

The Best ChatGPT Plugins

Web browsing, summarizing, coding, and more

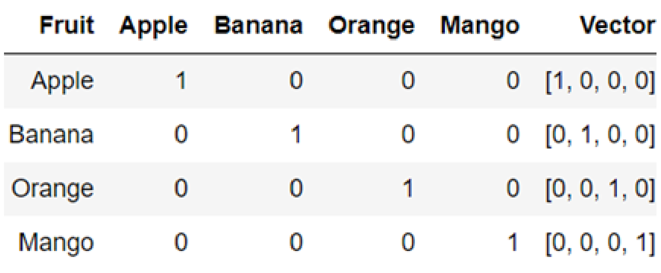
✦ · 8 min read · May 25

196 5

See all from Sophia Yang, Ph.D.

See all from Towards Data Science

Recommended from Medium



Fruit	Apple	Banana	Orange	Mango	Vector
Apple	1	0	0	0	[1, 0, 0, 0]
Banana	0	1	0	0	[0, 1, 0, 0]
Orange	0	0	1	0	[0, 0, 1, 0]
Mango	0	0	0	1	[0, 0, 0, 1]

Nimrita Koul

Natural Langauge Processing with Python Part 4: Text Representation

This article is the fourth in the series of my articles covering the sessions I delivered for…

9 min read · Apr 28

Rashini Liyanarachchi in Text Mining Basics

RegEx for Text Mining — I

RegEx or Regular Expressions are used in Text Mining mainly for Basic Patterns or…

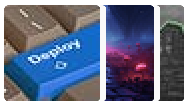
7 min read · Jul 25

1

Lists

Coding & Development

11 stories · 200 saves

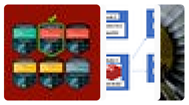


Predictive Modeling w/ Python

20 stories · 452 saves

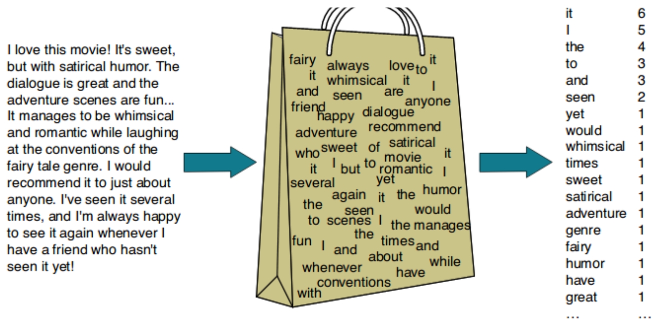
Practical Guides to Machine Learning

10 stories · 519 saves



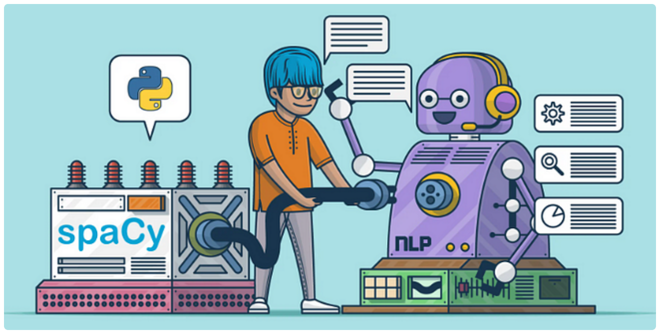
New_Reading_List

174 stories · 133 saves



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

```
it          6
I           5
the         4
to          3
and         3
seen        2
yet         1
would       1
whimsical   1
times       1
sweet       1
satirical   1
adventure   1
genre       1
fairy       1
humor       1
have        1
great       1
...         ...
```



Vamshi Prakash

An Introduction to Bag of Words (BoW)

Topic :-An Introduction to Bag of Words (BoW)

10 min read · Jun 27

2

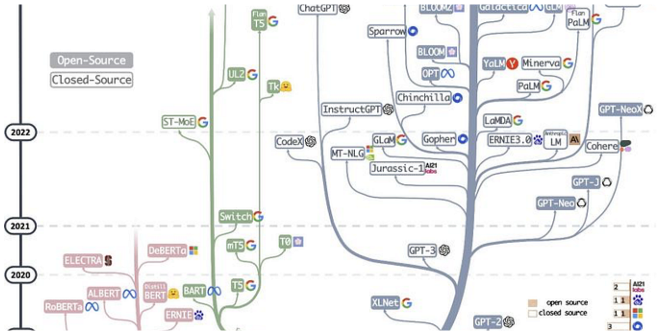
HasancanÇakıcıoğlu

Comprehensive Text Preprocessing NLP (Natural Language...

Text preprocessing plays a crucial role in Natural Language Processing (NLP) by...

12 min read · Jul 9

12



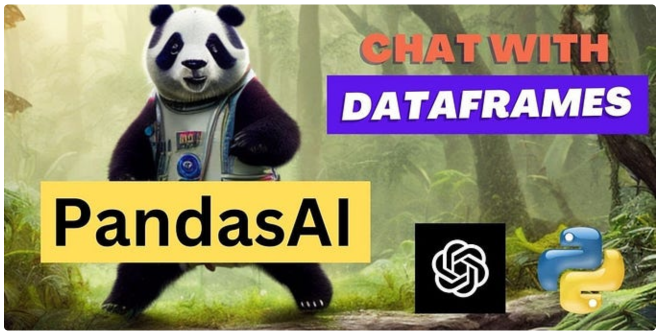
Haifeng Li

A Tutorial on LLM

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

15 min read · Sep 14

372



Avinash A

Introducing Pandas AI: Supercharging Data Analysis with...

Introducing Pandas AI: Supercharging Data Analysis with OpenAI

4 min read · Aug 16

6 1

See more recommendations