✨ Member-only story

HANDS-ON TUTORIALS, PYTHON LIBRARY

Scikit-LLM: Power Up Your Text Analysis in Python Using LLMs within scikit-learn Framework

Use advanced language models like ChatGPT to perform text classification, like sentiment analysis, text summarization, and other text analysis tasks

Essi Alizadeh · Follow

Published in Towards Data Science

10 min read · Jun 6

Listen Share

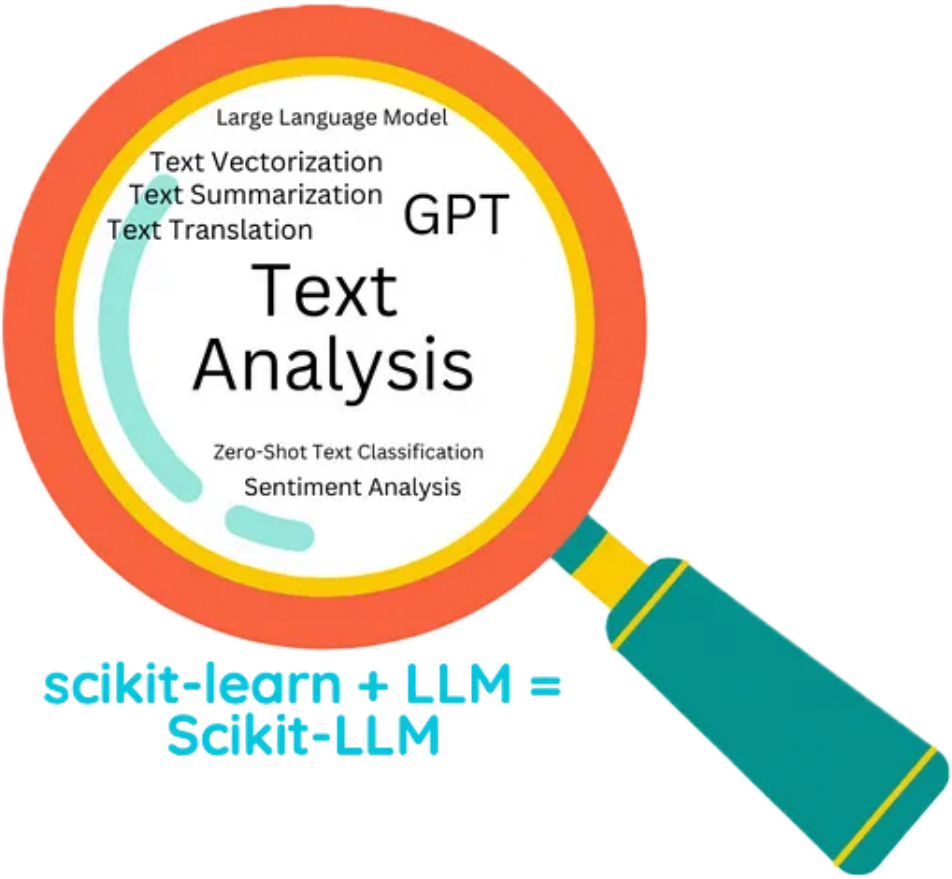
Photo by Patrick Tomasso on Unsplash

Introduction

Scikit-LLM is a Python package that integrates large language models (LLMs) like OpenAI's GPT-3 into the scikit-learn framework for text analysis tasks.

Scikit-LLM is designed to work within the scikit-learn framework. Hence, if you're familiar with scikit-learn, you'll feel right at home with scikit-llm. The library offers a range of features, out of which we will cover the following:

- Zero-shot text classification
- Multi-label zero-shot text classification
- Text vectorization
- Text translation
- Text summarization



Concepts covered in this post (Image by Author)

> *Please note that all datasets used in this post were exclusively created and compiled by the author.*

Why this library?

The main benefit of this library is it's familiar scikit-learn API, in particular,

- You can use similar APIs as scikit-learn, like `.fit()`, `.fit_transform()`, and `.predict()`.
- You can combine estimators from the scikit-llm library in a Sklearn pipeline (check out my last example in this post).

Installation

You can install the library via pip:

```
pip install scikit-llm
```

Configuration

Before you start using Scikit-LLM, you need to pass your OpenAI API key to Scikit-LLM. You can check out this post to set up your OpenAI API key.

```
from skllm.config import SKLLMConfig

OPENAI_SECRET_KEY = "sk-***"
OPENAI_ORG_ID = "org-***"

SKLLMConfig.set_openai_key(OPENAI_SECRET_KEY)
SKLLMConfig.set_openai_org(OPENAI_ORG_ID)
```

> *Please note that Scikit-LLM provides a convenient interface to access OpenAI's GPT-3 models. Use of these models is not free and requires an API key. While the API cost is relatively cheap, depending on the volume of your data and frequency of calls, these costs can add up. Therefore, it's important to plan and manage your usage carefully to control costs. Always remember to review OpenAI's pricing details and terms of use before getting started with Scikit-LLM.*
>
> *To give you a rough idea, I ran this notebook at least five times to make this tutorial, and the total cost was US $0.02. I have to say, I thought this would be higher!*

Zero-Shot Text Classification

One of the features of Scikit-LLM is the ability to perform zero-shot text classification. Scikit-LLM provides two classes for this purpose:

- **ZeroShotGPTClassifier**: used for single label classification (e.g. sentiment analysis),
- **MultiLabelZeroShotGPTClassifier**: used for a multi-label classification task.

Single label ZeroShotGPTClassifier

Let's do a sentiment analysis of a few movie reviews. For training purposes, we define the sentiment for each review (defined by a variable `movie_review_labels`). We train the model with these reviews and labels, so that we can predict new movie reviews using the trained model.

The sample dataset for the movie reviews is given below:

```
movie_reviews = [
    "This movie was absolutely wonderful. The storyline was compelling and the
    "I really loved the film! The plot had a few unexpected twists which kept m
    "The movie was alright. Not great, but not bad either. A decent one-time wa
    "I didn't enjoy the film that much. The plot was quite predictable and the
    "This movie was not to my taste. It felt too slow and the storyline wasn't
    "The film was okay. It was neither impressive nor disappointing. It was jus
    "I was blown away by the movie! The cinematography was excellent and the pe
    "I didn't like the movie at all. The story was uninteresting and the acting
    "The movie was decent. It had its moments but was not consistently engaging
]

movie_review_labels = [
    "positive",
    "positive",
    "neutral",
    "negative",
    "negative",
    "neutral",
    "positive",
    "negative",
    "neutral"
]

new_movie_reviews = [
    # A positive review
    "The movie was fantastic! I was captivated by the storyline from beginning

    # A negative review
    "I found the film to be quite boring. The plot moved too slowly and the act

    # A neutral review
    "The movie was okay. Not the best I've seen, but certainly not the worst."
]
```

Let’s train the model and then check what the model predicts for each new review.

```
from skllm import ZeroShotGPTClassifier

# Initialize the classifier with the OpenAI model
clf = ZeroShotGPTClassifier(openai_model="gpt-3.5-turbo")

# Train the model
```

```
clf.fit(X=movie_reviews, y=movie_review_labels)

# Use the trained classifier to predict the sentiment of the new reviews
predicted_movie_review_labels = clf.predict(X=new_movie_reviews)

for review, sentiment in zip(new_movie_reviews, predicted_movie_review_labels):
    print(f"Review: {review}\nPredicted Sentiment: {sentiment}\n\n")
```

```
Review: The movie was fantastic! I was captivated by the storyline from beginni
Predicted Sentiment: positive


Review: I found the film to be quite boring. The plot moved too slowly and the
Predicted Sentiment: negative


Review: The movie was okay. Not the best I've seen, but certainly not the worst
Predicted Sentiment: neutral
```

As can be seen above, the model predicted the sentiment of each movie review correctly.

Multi-Labels ZeroShotGPTClassifier

In the previous section, we had a single-label classifier ([“positive”, “negative”, “neutral”]). Here, we are going to use the `MultiLabelZeroShotGPTClassifier` estimator to assign multiple labels to a list of restaurant reviews.

```
restaurant_reviews = [
    "The food was delicious and the service was excellent. A wonderful dining e
    "The restaurant was in a great location, but the food was just average.",
    "The service was very slow and the food was cold when it arrived. Not a goo
    "The restaurant has a beautiful ambiance, and the food was superb.",
    "The food was great, but I found it to be a bit overpriced.",
    "The restaurant was conveniently located, but the service was poor.",
    "The food was not as expected, but the restaurant ambiance was really nice.
    "Great food and quick service. The location was also very convenient.",
    "The prices were a bit high, but the food quality and the service were exce
    "The restaurant offered a wide variety of dishes. The service was also very
]

restaurant_review_labels = [
    ["Food", "Service"],
```

```
    ["Location", "Food"],
    ["Service", "Food"],
    ["Atmosphere", "Food"],
    ["Food", "Price"],
    ["Location", "Service"],
    ["Food", "Atmosphere"],
    ["Food", "Service", "Location"],
    ["Price", "Food", "Service"],
    ["Food Variety", "Service"]
]

new_restaurant_reviews = [
    "The food was excellent and the restaurant was located in the heart of the
    "The service was slow and the food was not worth the price.",
    "The restaurant had a wonderful ambiance, but the variety of dishes was lim
]
```

Let's train the model and then predict the labels for new reviews.

```
from skllm import MultiLabelZeroShotGPTClassifier

# Initialize the classifier with the OpenAI model
clf = MultiLabelZeroShotGPTClassifier(max_labels=3)

# Train the model
clf.fit(X=restaurant_reviews, y=restaurant_review_labels)

# Use the trained classifier to predict the labels of the new reviews
predicted_restaurant_review_labels = clf.predict(X=new_restaurant_reviews)

for review, labels in zip(new_restaurant_reviews, predicted_restaurant_review_l
    print(f"Review: {review}\nPredicted Labels: {labels}\n\n")
```

```
Review: The food was excellent and the restaurant was located in the heart of t
Predicted Labels: ['Food', 'Location']


Review: The service was slow and the food was not worth the price.
Predicted Labels: ['Service', 'Price']
```

```
Review: The restaurant had a wonderful ambiance, but the variety of dishes was
Predicted Labels: ['Atmosphere', 'Food Variety']
```

The predicted labels for each review are spot-on.

Text Vectorization

Scikit-LLM provides the `GPTVectorizer` class to convert the input text into a fixed-dimensional vector representation. Each resulting vector is an array of floating numbers, which is a representation of the corresponding sentence.

Let's get a vectorized representation of the following sentences.

```
from skllm.preprocessing import GPTVectorizer

X = [
    "AI can revolutionize industries.",
    "Robotics creates automated solutions.",
    "IoT connects devices for data exchange."
]

vectorizer = GPTVectorizer()

vectors = vectorizer.fit_transform(X)

print(vectors)
```

```
[[-0.00818074 -0.02555227 -0.00994665 ... -0.00266894 -0.02135153
   0.00325925]
 [-0.00944166 -0.00884305 -0.01260475 ... -0.00351341 -0.01211498
  -0.00738735]
 [-0.01084771 -0.00133671  0.01582962 ...  0.01247486 -0.00829649
  -0.01012453]]
```

In practice, these vectors are inputs to other machine learning models for tasks like classification, clustering, or regression, rather than examining the vectors directly.

Text Translation

GPT models can be used to translate by making accurate readings from one language to another. We can translate a text into a language of interest using the `GPTTranslator` module.

```
from skllm.preprocessing import GPTTranslator
from skllm.datasets import get_translation_dataset

translator = GPTTranslator(openai_model="gpt-3.5-turbo", output_language="Engli

text_to_translate = ["Je suis content que vous lisiez ce post."]
# "I am happy that you are reading this post."

translated_text = translator.fit_transform(text_to_translate)

print(
    f"Text in French: \n{text_to_translate[0]}\n\nTranslated text in English: {
)
```

```
Text in French:
Je suis content que vous lisiez ce post.

Translated text in English: I am glad that you are reading this post.
```

Text Summarization

GPT models are very useful for summarizing texts. The Scikit-LLM library provides the `GPTSummarizer` estimator for text summarization. Let's see that in action by summarizing the long reviews given below.

```
reviews = [
    (
        "I dined at The Gourmet Kitchen last night and had a wonderful experien
```

```
            "The service was impeccable, the food was exquisite, and the ambiance w
            "I had the seafood pasta, which was cooked to perfection. "
            "The wine list was also quite impressive. "
            "I would highly recommend this restaurant to anyone looking for a fine
        ),
        (
            "I visited The Burger Spot for lunch today and was pleasantly surprised
            "Despite being a fast food joint, the quality of the food was excellent
            "I ordered the classic cheeseburger and it was juicy and flavorful. "
            "The fries were crispy and well-seasoned. "
            "The service was quick and the staff was friendly. "
            "It's a great place for a quick and satisfying meal."
        ),
        (
            "The Coffee Corner is my favorite spot to work and enjoy a good cup of
            "The atmosphere is relaxed and the coffee is always top-notch. "
            "They also offer a variety of pastries and sandwiches. "
            "The staff is always welcoming and the service is fast. "
            "I enjoy their latte and the blueberry muffin is a must-try."
        )
    ]

    # NOTE
    # string1 = "ABC"
    # string2 = ("A" "B" "C")
    # print(string1 == string2)
    # >>> True
```

Note that the `reviews` above is a list of three items, and that each item in the list is written out in a way that makes it easy to read.

```
from skllm.preprocessing import GPTSummarizer

gpt_summarizer = GPTSummarizer(openai_model = "gpt-3.5-turbo", max_words = 15)


summaries = gpt_summarizer.fit_transform(reviews)

print(summaries)
```

A short summary of each review is generated. The `max_words` parameter sets a rough upper bound on the summary's length; in practice, it may be a slightly longer.

Use a scikit-llm within a scikit-learn pipeline

So far, all examples above have only used estimators from the Scikit-LLM library. As mentioned before, the main benefit of this library is its integration with the scikit-learn platform.

The following example uses a scikit-llm estimator in a scikit-learn pipeline and runs an XGBoost classifier on the movie review examples illustrated earlier.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from skllm.preprocessing import GPTVectorizer
from xgboost import XGBClassifier

# Define variables with conventional names to reduce confusion
X, y = movie_reviews, movie_review_labels
X_test, y_test = new_movie_reviews, new_movie_review_labels

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)
y_test_encoded = le.transform(y_test)

# Use a scikit-learn pipeline
steps = [("GPT", GPTVectorizer()), ("Clf", XGBClassifier())]
clf = Pipeline(steps)

clf.fit(X, y_encoded)

y_pred_encoded = clf.predict(X_test)

# Revert the encoded labels to actual labels
y_pred = le.inverse_transform(y_pred_encoded)

print(f"\nEncoded labels (train set): {y_encoded}\n")
print(f"Actual Labels (train set): {y}")

print(f"Predicted labels (encoded): {y_test_encoded}\n")

print("------------------\nEvaluate the performance of XGBoost Classifier:\n")
for test_review, actual_label, predicted_label in zip(X_test, y_test, y_pred):
    print(f"Review: {test_review}\nActual Label: {actual_label}\nPredicted Labe
```

```
Encoded labels (train set): [2 2 1 0 0 1 2 0 1]

Actual Labels (train set): ['positive', 'positive', 'neutral', 'negative', 'neg
Predicted labels (encoded): [2 0 1]

------------------
Evaluate the performance of XGBoost Classifier:

Review:The movie was fantastic! I was captivated by the storyline from beginnin
Actual Label: positive
Predicted Label: positive

Review:I found the film to be quite boring. The plot moved too slowly and the a
Actual Label: neutral
Predicted Label: neutral
```


Please note that the above is only a use case to illustrate the possibility of integrating Scikit-LLM estimators in a SKlearn pipeline.

If you enjoyed this post, you're going to like my other article on String-to-String algorithms for NLP tasks.

Taming Text with string2string: A Powerful Python Library for String-to-String Algorithms

Leverage string2string for Natural Language Processing Tasks

towardsdatascience.com

Conclusion

Scikit-LLM is a powerful tool that adds the power of advanced language models like GPT-3 to the well-known scikit-learn framework. In this tutorial, we looked at some of Scikit-LLM's most important features: 1) zero-shot text classification, 2) multi-label zero-shot text classification, 3) text vectorization, 4) text summary, 5) language translation, and 6) integration with scikit-learn pipeline

As I mentioned earlier, the main benefit of this library is its similar APIs to the scikit-learn and its integration into scikit-learn pipelines. However, the main limitation of Scikit-LLM is its heavy reliance on OpenAI. Although integrating open-source models is on Scikit-LLM's roadmap, it's not yet available in the library. Hence, if you use Scikit-LLM, you should know that:

- Since Scikit-LLM uses OpenAI models, there is an API cost or a rate limit on the number of requests made to OpenAI.
- The Scikit-LLM is bound to have the same limitations as the OpenAI API, like not having access to the internet or the maximum tokens allowed for a given OpenAI model (in this post, the "gpt-3-turbo" model is mainly used, which has a maximum of 4096 tokens).

Scikit-LLM is a promising tool that opens up new possibilities in the realm of text analysis using large language models. This library might be a useful addition to your toolbox; therefore, I suggest giving it a try.

> 📓 *You can find the notebook for this post on GitHub.*

Thanks for reading! 📚

I'm a senior data scientist 📊 and engineer, writing about statistics, machine learning, Python, and more.

> 🌱 *I also curate a weekly newsletter called **AI Sprout** where I provide hands-on reviews and analysis of the latest AI tools and innovations. Subscribe to explore emerging AI with me!*

- *Follow me on Medium* 👋 *to get my latest post*

- *Subscribe to my mailing list ✉️ for updates right to your inbox*
- *Let's connect on LinkedIn and Twitter* 🤝

Join Medium with my referral link - Esmaeil Alizadeh

📖 Read every story from Esmaeil Alizadeh (and thousands of other writers on Medium). Subscribe to Medium to get full...

medium.ealizadeh.com

Reference

GitHub - iryna-kondr/scikit-llm: Seamlessly integrate powerful language models like ChatGPT into...

Seamlessly integrate powerful language models like ChatGPT into scikit-learn for enhanced text analysis tasks. You can...

github.com

Useful Links

scikit-learn

"We use scikit-learn to support leading-edge basic research [...]" "I think it's the most well-designed ML package I've...

scikit-learn.org

How to Get an OpenAI API Key

Lots of applications and AI tools now require you bring your own OpenAI API key. You can generate one on OpenAI's...

www.howtogeek.com

Originally published at https://ealizadeh.com.

NLP ChatGPT Deep Learning Machine Learning Python

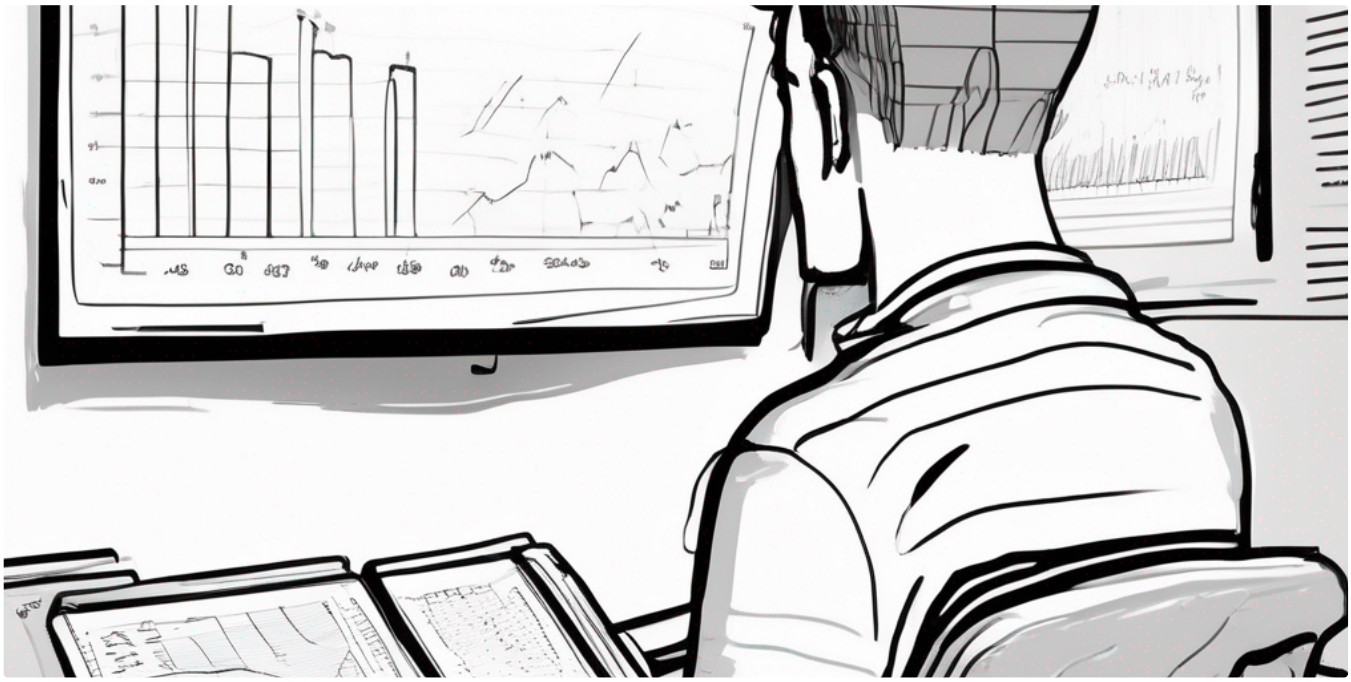
Follow

Written by Essi Alizadeh

302 Followers · Writer for Towards Data Science

Data Scientist | 200k+ Med Views | AI Sprout Newsletter: aisprout.xyz | 📝 Ghostwriting Email Courses for AI Startups | 🆓 5-Day 🎁 Course 👉 ChatGPTin5Days.com

More from Essi Alizadeh and Towards Data Science



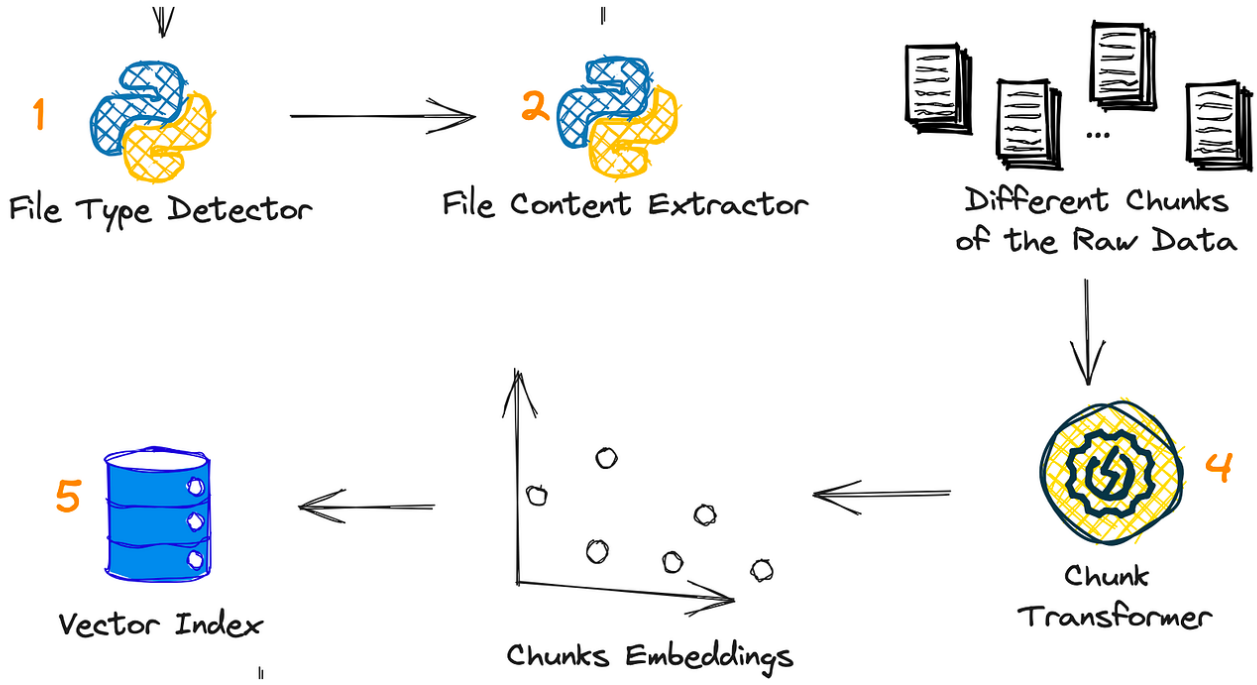
Essi Alizadeh in Towards Data Science

The Two Metrics That Reveal True Data Dispersion Beyond Standard Deviation

A guide to computing and interpreting Coefficient of Variation and Quantile Coefficient of Dispersion

✦ · 8 min read · Jul 30

-- 4



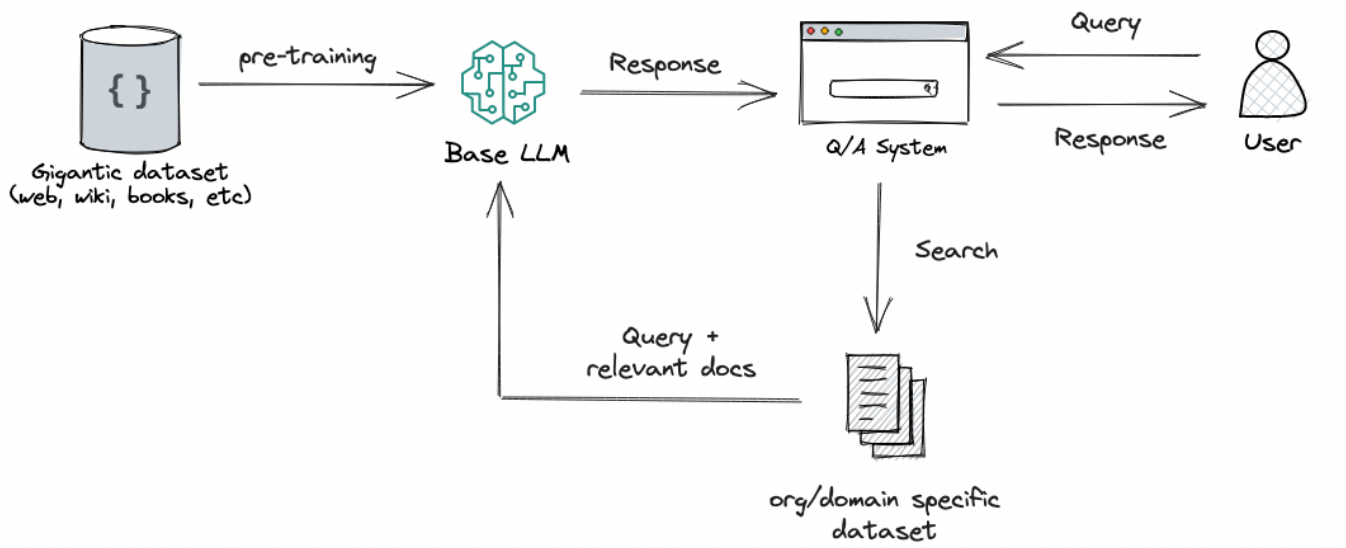
Zoumana Keita in Towards Data Science

How to Chat With Any File from PDFs to Images Using Large Language Models—With Code

Complete guide to building an AI assistant that can answer questions about any file

✦ · 9 min read · Aug 5

-- 12



Heiko Hotz in Towards Data Science

RAG vs Finetuning—Which Is the Best Tool to Boost Your LLM Application?

The definitive guide for choosing the right method for your use case

✦ · 19 min read · Aug 24

-- 14

Essi Alizadeh in Towards Data Science

Taming Text with string2string: A Powerful Python Library for String-to-String Algorithms

Leverage string2string for Natural Language Processing Tasks

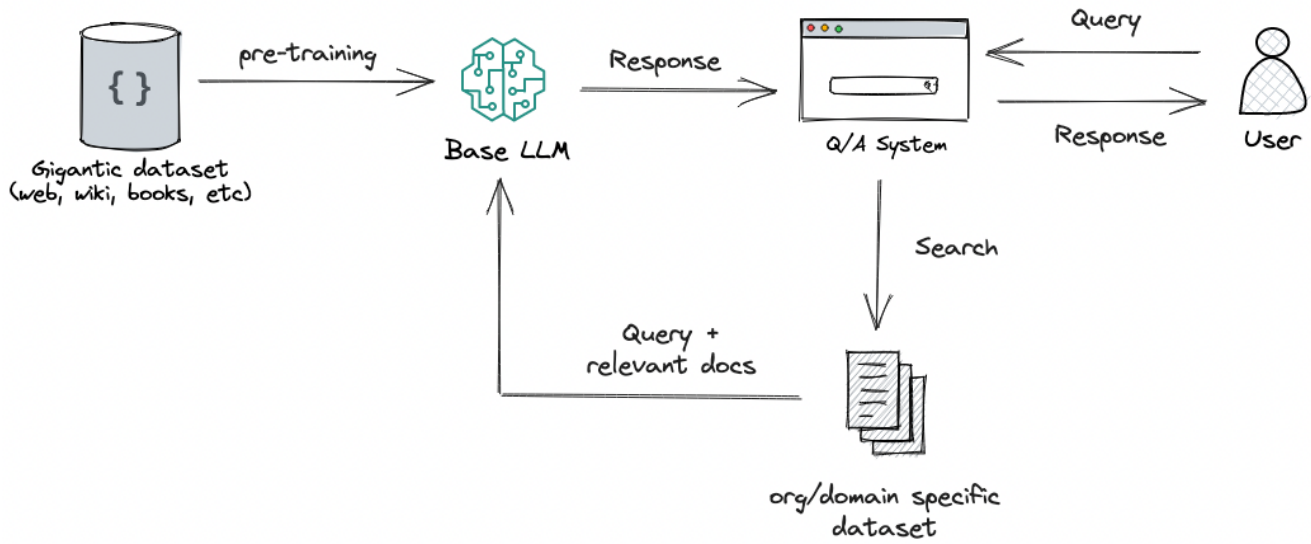
✦ · 8 min read · May 11

--

See all from Essi Alizadeh

See all from Towards Data Science

Recommended from Medium



Heiko Hotz in Towards Data Science

RAG vs Finetuning—Which Is the Best Tool to Boost Your LLM Application?

The definitive guide for choosing the right method for your use case

✦ · 19 min read · Aug 24

-- 14

TeeTracker

Fine-tuning LLMs

Tasks to finetune

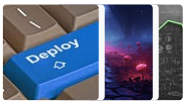
6 min read · Jul 22

-- 1

Lists

Natural Language Processing

574 stories · 193 saves



Predictive Modeling w/ Python

20 stories · 351 saves

Practical Guides to Machine Learning

10 stories · 388 saves

The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 107 saves

five stars+"	General Feedback
"This pop-up \"Get the best Spotify experience on Android 12\" is too annoying. Please let's get rid of this."	Feature Request
"Really buggy and terrible to use as of recently"	Bug
"Dear Spotify why do I get songs that I didn't put on my playlist??? And why do we have shuffle play?"	General Feedback
"The player controls sometimes disappear for no reason. App restart forgets what I was playing but fixes the issue."	Bug
"I love the selection and the lyrics are provided with the song you're listening to!"	General Feedback
"Still extremely slow when changing storage to external sd card.. I'm convinced this is done on purpose, spotify knows of this issue and has done NOTHING to solve it! Over time I have changed sd cards, each being faster in read, write speeds(all samsung brand). And please add \"don't like song\" so it will never appear again in my searches or playlists."	Bug
"It's a great app and the best mp3 music app I have ever used but there is	

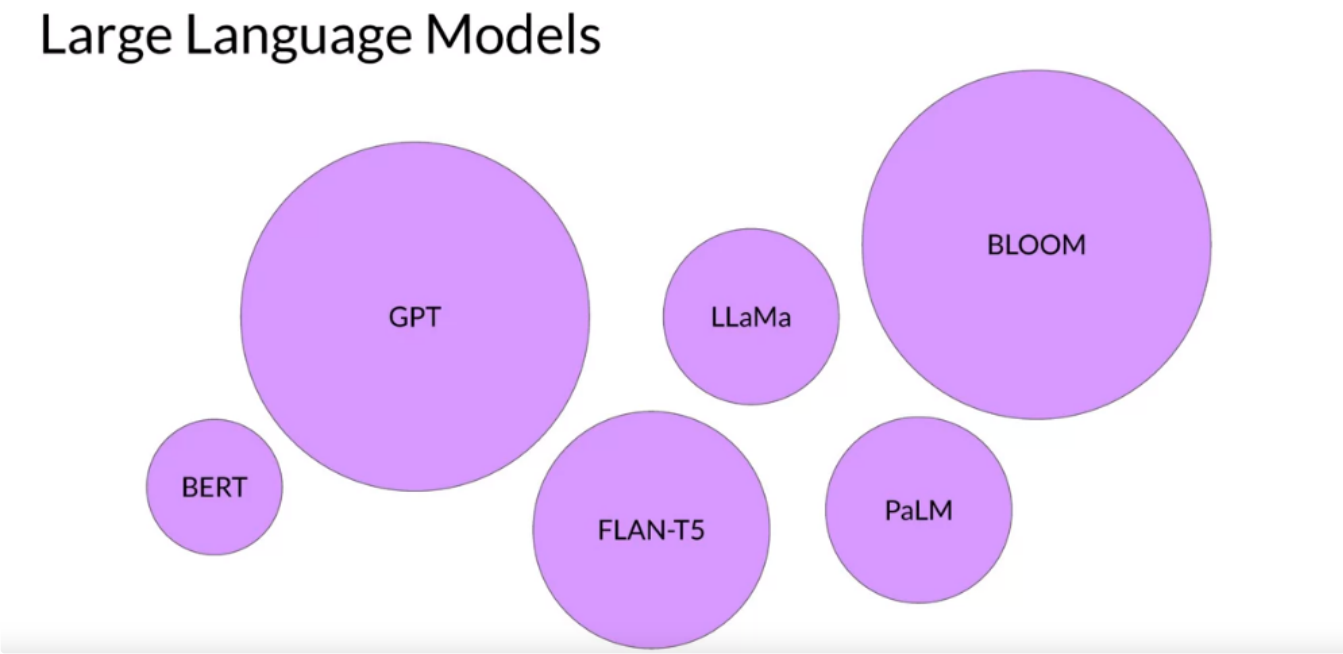
Santhosh Kumar Setty in Bootcamp

Perform Sentiment Analysis with ChatGPT APIs for Customer Insights

With GPT-3.5-turbo from OpenAI, you can now turbocharge your sentiment analysis effortlessly, without the need for extensive coding skills!

7 min read · Jun 19

-- 1



Yash Bhaskar

Introduction to LLMs and the generative AI : Part 1- LLM Architecture, Prompt Engineering and LLM...

Large language models (LLMs) have revolutionized the field of artificial intelligence (AI) development, offering developers unprecedented...

11 min read · Jul 16

-- 2

Wei-Meng Lee in Level Up Coding

Training Your Own LLM using privateGPT

Learn how to train your own language model without exposing your private data to the provider

· 8 min read · May 19

-- 12

 Sachin Kulkarni

Generative AI with Enterprise Data

Create business value add Enterprise knowledge to Large Language Models

6 min read · Jul 25

-- 3

See more recommendations