Cluster chatter: HDBSCAN + LLM

Sirsh Amarteifio · Following
5 min read · Jun 13

HDBSCAN *is a density based (hierarchical) clustering algorithm. Clustering algorithms are generally thought of as an unsupervised machine learning method in the sense that you can get something out of your data without a target or even a clear question. The idea is intuitive i.e. finding similarities in N dimensional vector spaces and the results often visualize quite well. Here I take a look at how text summaries of clustering results from an LLM look, and extend a previous article on a similar theme with another small experiment.*

Lets consider some data. I generated sample data about ecommerce products, failures, returns etc. but the data can be anything you care about. Lets assume the data are normalized e.g. use “rank” columns or other normalized columns in the range 0,1 say. Lets maybe add some boolean columns too. Machine learning models certainly do like normalized data. By the way, normalizing plus rounding cell values will be kind to our LLM token limit too. Similarly, removing columns (dimensional reduction) via PCA or some such would help the clustering and the LLM.

We will first cluster and then sample randomly from our data up to some limit e.g. 40–50 rows depending on how many columns we have. We will then ask the LLM for a summary. If the clustering algorithm does what it is supposed to do (which we will assume for now) then the samples should all be representative and we can get a good idea of what they are about from sub-sampling and describing the set.

```
import hdbscan
from tqdm import tqdm
from langchain.llms import OpenAI

def explain_data(df, min_cluster_size=100, sample_size=50 ):
    """
    df is a dataframe properly prepared
     e.g. with a key and normalized and binary columns

    min_cluster_size: hdbascan see: https://hdbscan.readthedocs.io/en/latest/how
    sample_size: sub sample the class. value is small enough to fit in a token l
    """
    clusterer = hdbscan.HDBSCAN(min_cluster_size=min_cluster_size,
                                gen_min_span_tree=True)
    m = clusterer.fit(df)

    df['cid'] = m.labels_
    llm = OpenAI(model_name=model_name, temperature=0.0)
    answers = []
    for i in tqdm(df['cid'].unique()):
        sample = df[df['cid']==i].sample(sample_size)
        print(f'\n<<<Asking the LLM for a summary for cluster indexed {i}. its s
        ans = llm(f"""You will be given a dataset with boolean and rank columns
                  List the interesting patterns in the data referring to the
                  If columns are constant say so but emphasize any other low
                  At the end, given an overall summary of what you can say ab
                  Data: {sample.to_dict('records')}
                  """)
        print(ans)
        answers.append(ans)

    return answers
```

```
summaries = explain_data(df)
summaries
```

Example output

```
1. The columns 'has_product_recently_failed', '_sku_hsum', 'has_order_recently_f
2. The 'has_delivery_recently_failed' column has low entropy, with most of the v
3. The 'item_failure_probability' column has a constant value of 0.5, indicating
4. The 'error_probability' column has a range of values, but the majority of the
5. The 'product_category_failure_probabability', 'order_failure_probability', 'c
6. The 'component_failure_probability' column also has a wide range of values, s

In summary, the products in class "cid" 8 have varying failure probabilities acr
 However, there have been some recent delivery failures. The error probability s
```

Then we can re-summarize

```
#from IPython.display import HTML

r = llm(f"""The data provide summaries of different data classes.
        Please give an overall summary of the different classes,
        describing any differences and similarities between them.
        Mention which classes have the most problems recently and overall.
        Comment specifically on the low entropy features which may characterize
        Provide a structured answer in HTML format.
        Summaries:
        {summaries}
    """)

#HTML(r)
```

Overall Summary

The data consists of several classes (cid) with different patterns and characteristics. The classes are: 2, 3, 7, 5, 4, 6, -1, 0, 8, and 1. In general, most classes have constant boolean columns and failure.

Classes with the most problems recently are class 0 and class 8, which have recent delivery failures. Class 6 has consistent recent component size failures.

Low entropy features are present in several classes, such as error_probability and item_failure_probability columns, which have low variability. Some classes also have low entropy in the boc

Class-specific summaries:

- **Class 2:** No recent failures, low variability in error_probability and item_failure_probability, diverse set of products with different failure probabilities.
- **Class 3:** Constant error_probability and item_failure_probability, no recent failures, wide range of values for product category, order, component size, and delivery failure probabilities.
- **Class 7:** Constant component size and item failure probabilities, low error_probability, no recent failures, low entropy in _sku_hsum column.
- **Class 5:** No recent failures, constant item_failure_probability, wide range of failure probabilities for components, product categories, orders, and deliveries.
- **Class 4:** Higher probability of failure in product category, order, and delivery, no recent failures, constant component size and item failure probabilities.
- **Class 6:** Consistent recent component size failures, constant item_failure_probability, wide range of values for other failure probabilities.
- **Class -1:** Constant item_failure_probability, low entropy in recent failure columns, some variation in other failure probabilities.
- **Class 0:** No recent failures except for delivery, constant item_failure_probability, high probability of delivery failure, varying values for other failure probabilities.
- **Class 8:** No recent failures except for delivery, constant item_failure_probability, two distinct groups in error_probability, varying values for other failure probabilities.
- **Class 1:** No recent failures, constant item_failure_probability, varying values for other failure probabilities, some variation in _sku_hsum column.

If we are lazy, we might get excited about this as a sort of auto-EDA to save ourselves the trouble of exploring our own data. This is not my angle. Instead, think of this as part of an automated pipeline to report on patterns of interest in tabular datasets observed in some backend system. For example with this approach we can summarize a large amount of data by clustering, sampling and then sending summaries. We assume the sampled rows in each cluster will be representative of intersting classes in the data and, taken together, the combined summaries make for a good overall narrative. Probably, some classes will be much more relevant than others.

So for instance if you take the running example from my previous article, we could summarize our "data and stats" tool data, extract an overall summary and then highlight specific entities retrieved from the entity tool and summarize those as cases that are interesting for whatever reasons the cluster description says they are interesting. Maybe some of them are problem cases or anomalies or whatever. The exemplars also provide more colour and context to enrich the cluster descriptions.

Maybe you are thinking we should just take all the data and send it to some LLM with an obnoxiously large token limit and let it do all the work. Sure, we could try that too I guess.

Links

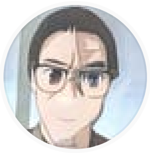
GitHub - scikit-learn-contrib/hdbscan: A high performance implementation of HDBSCAN clustering.

A high performance implementation of HDBSCAN clustering. - GitHub - scikit-learn-contrib/hdbscan: A high performance...

github.com

Llm Clustering Hdbscan

View list

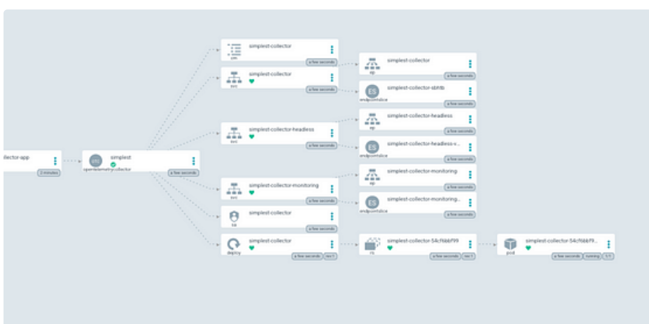


Written by Sirsh Amarteifio

11 Followers

Following

More from Sirsh Amarteifio



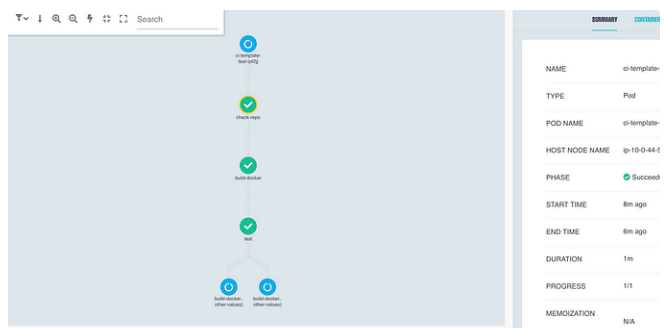
Sirsh Amarteifio

Some thoughts about Argo GitOps for K8s cluster bootstrapping

Following a theme in other articles, today I'm going to iterate on finding various ways to...

15 min read · Jun 18

1



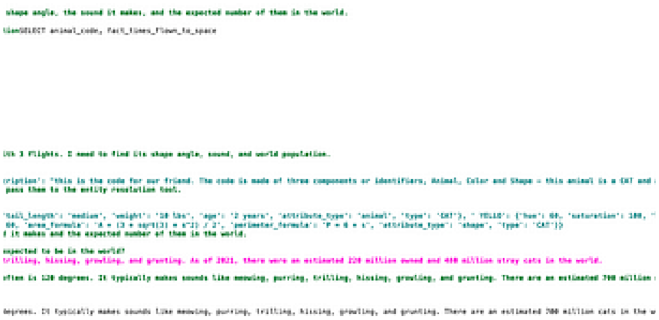
Sirsh Amarteifio

A simple Argo Workflow to build and push (ECR) Docker images in...

Lets build a workflow that could be used as part of a CI/CD pipeline. This starts by clonin...

10 min read · Jun 19

20



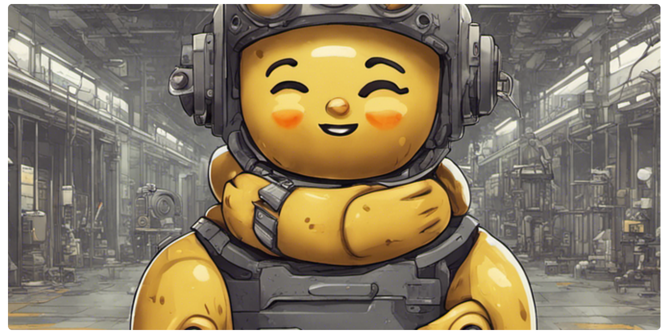
Sirsh Amarteifio

Does your LLM model understand your entities?

For applying LLMs to your own data, one interesting problem is merging context…

14 min read · Jun 3

3



Sirsh Amarteifio

0–1: Training Hugging Face pipelines on a simple Argo…

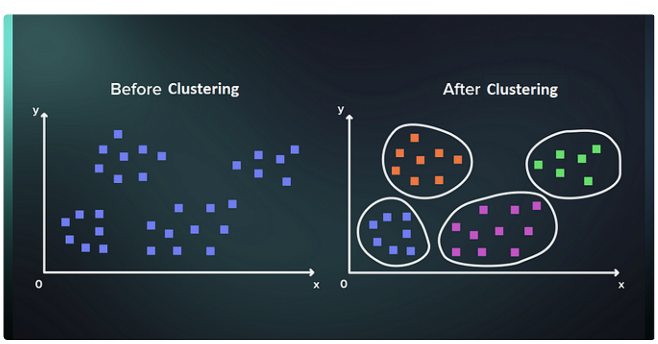
Illustration of a simple Argo workflow to train a Hugging Face model/pipeline on a K8s GP…

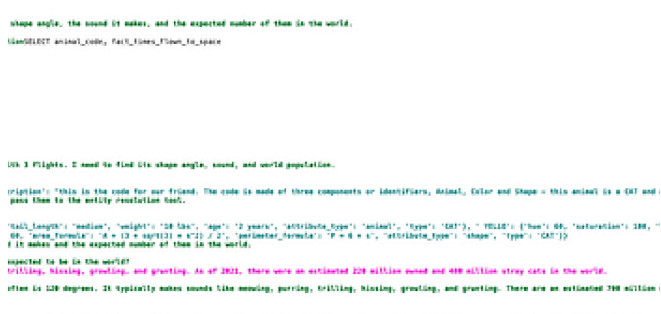
11 min read · Aug 27

2

See all from Sirsh Amarteifio

Recommended from Medium





Deniz Gunay

Clustering

Clustering

20 min read · Sep 18

26

Sirsh Amarteifio

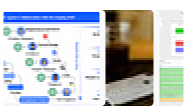
Does your LLM model understand your entities?

For applying LLMs to your own data, one interesting problem is merging context…

14 min read · Jun 3

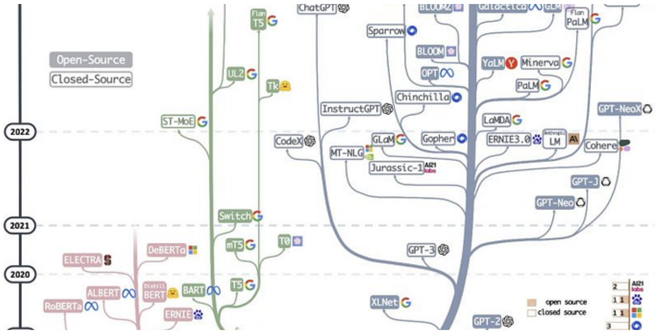
3

Lists



Natural Language Processing

669 stories · 283 saves



Haifeng Li

A Tutorial on LLM

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

15 min read · Sep 14

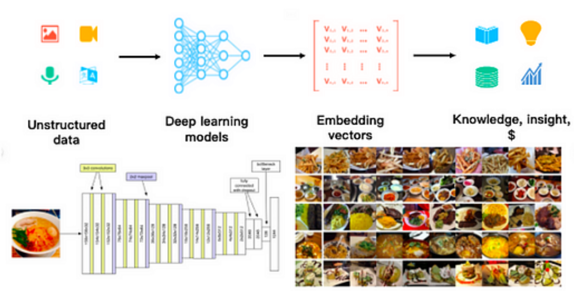
372

Guillaume Michel in Kensho Blog

Kensho Classify: The Solution to Common Challenges of Text…

Text classification is widely used in many industries and often serves as a pillar for mo…

4 min read · Sep 7



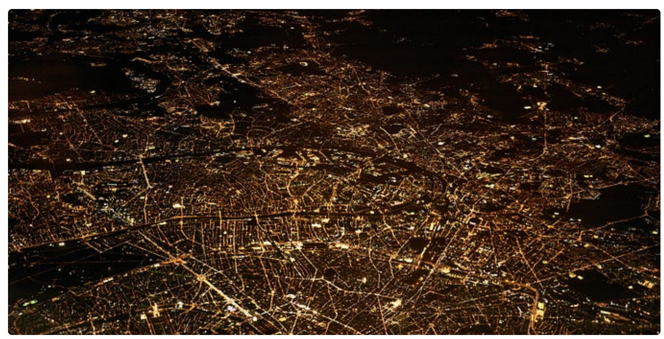
Jayita Bhattacharyya in GoPenAI

Primer on Vector Databases and Retrieval-Augmented Generation…

Vector Databases Generation (RAG) Langchain Pinecone HuggingFace Large…

9 min read · Aug 16

228 1



Yannis Poulakis

Is Silhouette the Right Clustering Evaluation Metric for You?

While popular, simple experiments prove that Silhouette index may not be the index to…

4 min read · Jul 4

17

See more recommendations