

Hosting a Vector Database with Hugging Face

Here's a fun app idea: **You Know What They Say**.

The user gives the app a sentence, and the app returns a relevant proverb.

Example:

```
1 User > "My friend Tommy broke his computer while trying to install a better battery!"
2 App > "Well, you know what they say: If it ain't broke, don't fix it!"
3
4 User > "I went to the beach before sunrise but all the good spots were taken."
5 App > "Well, you know what they say: Early bird gets the worm."
```

We can implement this app by storing hundreds (or even thousands) of English proverbs in a vector database, running a semantic search with the user's input sentence, and then fetching and returning the most relevant proverb.

Now, what if we wanted to store the data for our app online and do semantic search via HTTP calls instead of locally?

Guess what - Hugging Face allows us to host and query vector datasets online! Let's see how below.

First, we need to export our vector database to a `.csv` file. We can do this by using `pandas.to_csv()` as follows:

```
1 # Convert only the embeddings to a DataFrame, without the texts
2 df_embeddings = pd.DataFrame(embeddings)
3
4 # Export the dataframe as a .csv file
5 df_embeddings.to_csv("embeddings.csv", index=False)
```

After doing so, you should see `embeddings.csv` in your IDE file explorer.

Next, create a new dataset on Hugging Face by doing the following:

1. Navigate to <https://huggingface.co/>
2. Click on your user picture in the right-top

3. Click "New Dataset"
4. Give your dataset a name (e.g. "my_test_dataset")
5. Make sure the "Public" box is checked
6. Hit "Create dataset"

The screenshot shows the Hugging Face website interface. At the top, there's a navigation bar with the Hugging Face logo, a search bar, and links for Models, Datasets, Spaces, Docs, Solutions, and Pricing. The user profile 'efkansg' is logged in, and a dropdown menu is open, showing options like Profile, Notifications, New Model, New Dataset, New Space, Create organization, Settings, and Sign Out. A red arrow points to the 'New Dataset' option. The main content area shows 'Efkan S. Goktepe's Activity' with tabs for All, Models, Datasets, Spaces, Community, and Likes. Below this, there's a 'Trending' section with a list of models, including 'meta-llama/Llama-2-7b-chat-hf' and 'meta-llama/Llama-2-70b-hf'. A red arrow also points to the 'New Dataset' option in the dropdown menu.

Hugging Face

Search models, datasets, users...

Models Datasets Spaces Docs Solutions Pricing

+ New

efkansg

- Profile
- Inbox (2)
- Settings
- Get Pro

Organizations

- IBM
- Create New

Resources

- Hub guide
- Transformers doc
- Forum
- Tasks
- Learn

Light theme

Efkan S. Goktepe's Activity

All Models Datasets Spaces Community Likes

No recent user activity to display

Trending last 7 d

All Models Datasets

meta-llama/Llama-2-7b-chat-hf

Text Generation

meta-llama/Llama-2-70b-hf

Text Generation

Running on CPU UPGRADING

Explore

meta-llama/Llama-2-7b-chat-hf

Text Generation Updated about 1 day ago 87.2k 355

meta-llama/Llama-2-70b-hf

Text Generation Updated about 1 day ago 158k 330

stabilityai/FreeWilly2

Text Generation Updated 3 days ago 592 300

Open LLM Leaderboard 3.62k

fka/awesome-chatgpt-prompts

Viewer Updated Mar 7 2.13k 2.66k

Open-Orca/OpenOrca

Viewer Updated 10 days ago 6k 451

Salesforce/dialogstudio

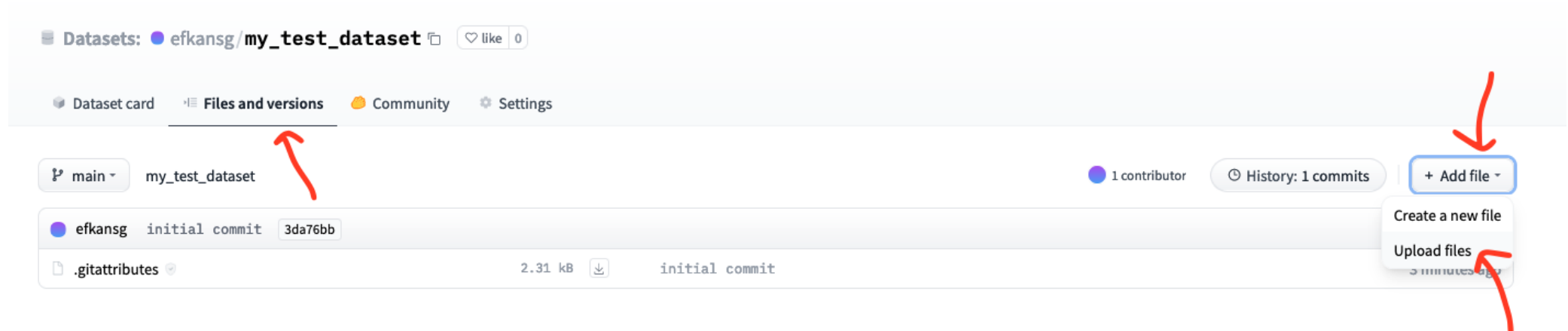
Updated about 7 hours ago 171 57

lmsys/chatbot_arena_conversations

Viewer Updated 4 days ago 184 53

Then, you should be redirected to your dataset. Add your `.csv` file to the dataset by doing:

1. Navigate to the “Files and Versions” section of your dataset
2. Click “Add File -> Upload files”
3. Drag and drop your files into the box, and hit “Commit changes to `main`”



my_test_dataset/

The screenshot shows the Hugging Face file upload interface. At the top, there's a header "Upload file(s)". Below it is a large dashed box with the text "Drag files/folders here or click to browse from your computer." A red arrow points to this text. Below the dashed box is a file list showing "embeddings.csv" with a red arrow pointing to it. Underneath the file list are two radio buttons: "Commit directly to the main branch" (selected) and "Open as a pull request to the main branch". Below these is a section titled "Commit changes" with a text input field containing "Upload embeddings.csv". Underneath this is a tabbed interface with "Edit" and "Preview" tabs. The "Edit" tab is active, showing a text area with the placeholder "Add an extended description...". A red arrow points to this text area. Below the text area is a small icon and the text "Upload images, audio, and videos by dragging in the text input, pasting, or clicking here." At the bottom of the form are two buttons: "Commit changes to main" and "Cancel".

After uploading our dataset, we can load the embedded dataset from Hugging Face using the `datasets` library and convert it to a PyTorch FloatTensor, which is one way to operate on the data. Make sure to replace `namespace/repo_name` with your user and repo name.

```
1 import torch
2 from datasets import load_dataset
```

```

3
4 articles_embeddings = load_dataset('namespace/repo_name')
5 dataset_embeddings = torch.from_numpy(articles_embeddings["train"].to_pandas().to_numpy()).to(torch.float)

```

Given any article name, suppose we would like to conduct semantic search to generate relevant suggestions. Similar to the previous section, we first get the embeddings for that article.

```

1 cat_article = ['Cats have very fast reflexes']
2 response = requests.post(api_url, headers=headers, json={"inputs": cat_article, "options":{"wait_for_model":True}})
3
4 query_embeddings = torch.FloatTensor(response.json())

```

Next, we can use the `sentence_transformers` library to query our data. Let's use it to find the most relevant 2 articles.

```

1 from sentence_transformers.util import semantic_search
2
3 # Find top 2 similar vectors using semantic_search
4 hits = semantic_search(query_embeddings, dataset_embeddings, top_k=2)
5
6 # Print result
7 print(hits)

```

From doing so, you should get the following list:

```

1 [
2   [
3     {'corpus_id': 1, 'score': 0.7399995923042297},
4     {'corpus_id': 5, 'score': 0.6190879344940186}
5   ]
6 ]

```

The `corpus_id` field in each row refers to the index at which the 'similar' vector was found. We can use this value to index the text from our starting data.

```
1 print([examples[hits[0][i]['corpus_id']] for i in range(len(hits[0]))])
```

The output should be:

```
1 [  
2     'There are no animals with faster reflexes than cats.',  
3     'Felines are very quick to react to sudden events.'  
4 ]
```