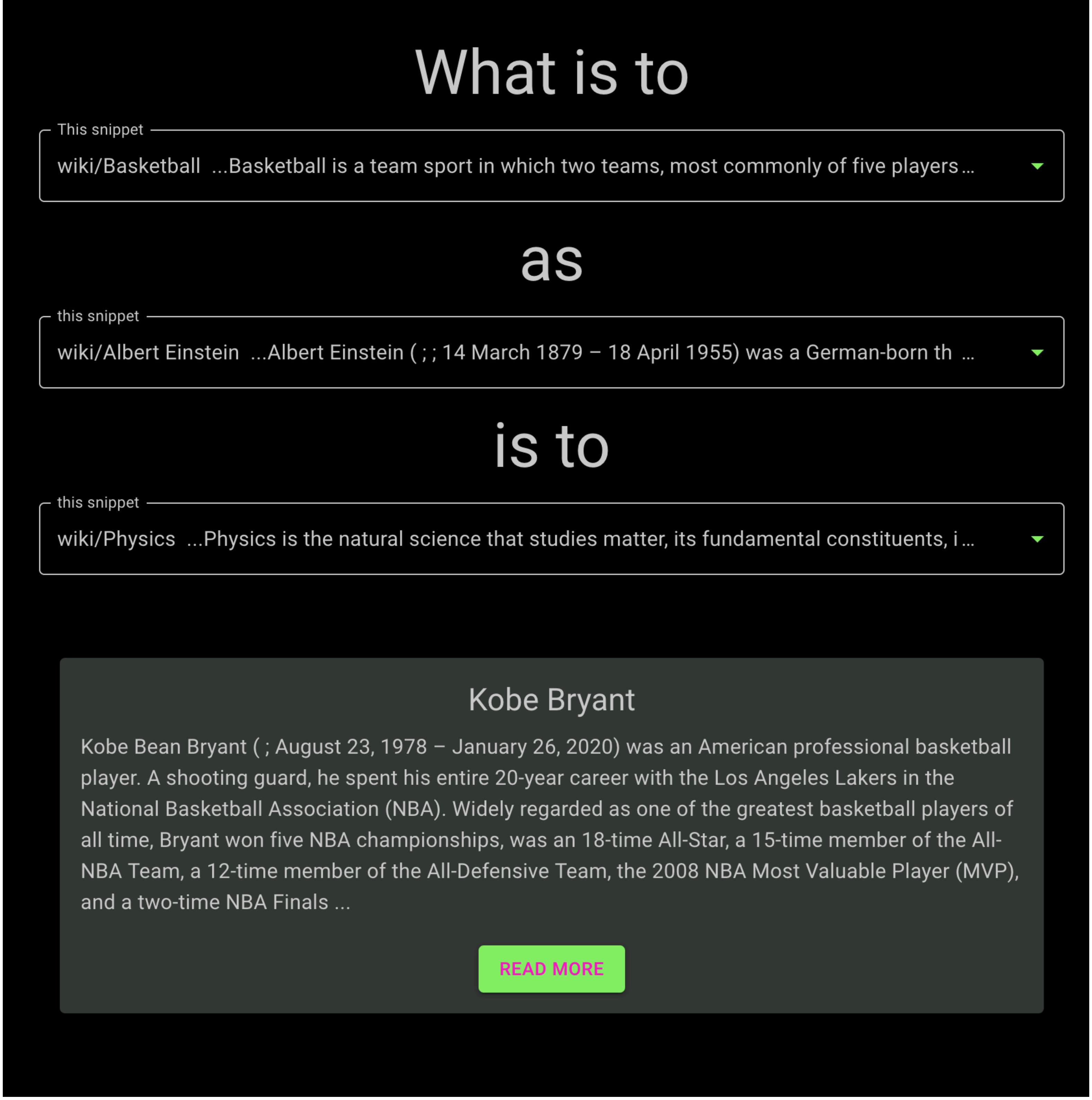
Embed Wikipedia with Modal and search it with Weaviate


What is to

This snippet

wiki/Basketball ...Basketball is a team sport in which two teams, most commonly of five players ...

as

this snippet

wiki/Albert Einstein ...Albert Einstein (; ; 14 March 1879 – 18 April 1955) was a German-born th ...

is to

this snippet

wiki/Physics ...Physics is the natural science that studies matter, its fundamental constituents, i ...

Kobe Bryant

Kobe Bean Bryant (; August 23, 1978 – January 26, 2020) was an American professional basketball player. A shooting guard, he spent his entire 20-year career with the Los Angeles Lakers in the National Basketball Association (NBA). Widely regarded as one of the greatest basketball players of all time, Bryant won five NBA championships, was an 18-time All-Star, a 15-time member of the All-NBA Team, a 12-time member of the All-Defensive Team, the 2008 NBA Most Valuable Player (MVP), and a two-time NBA Finals ...

This sample project demonstrates the powerful combo of serverless infrastructure from Modal and the search capabilities of Weaviate for projects that combine data-intensive Python compute, like neural network inference, with data-intensive search, like indexing all of Wikipedia.

You can find the code on GitHub here. It’s intended as a jumping off point for your own code that combines Modal with databases like Weaviate and with JavaScript frontends. It is also deployed as a live demo application.

Overview

The `frontend` of this project (written in React, hosted on Vercel) allows users to construct "vector analogies" of the form made famous by Word2Vec. For example, the approximation

```
Albert Einstein - Physics + Basketball ~= Kobe Bryant
```

expresses the analogy "Kobe Bryant is the Albert Einstein of basketball". We can compute it by applying those operations to embedding vectors of each concept, where `~=` is implemented using an approximate nearest-neighbor search index, the key method used for querying in vector databases.

Where Word2Vec used word embeddings to express concepts, we use snippets of Wikipedia articles. The dataset used was constructed from the March 2022 WikiMedia dump by Hugging Face.

Users can type into each search bar to find a snippet of interest, using Weaviate text search under the hood, and once they've selected the three components of their analogy, the frontend kicks off a vector search to complete it.

Both searches are coordinated by a `backend` Python service running on Modal.

Modal is also used to construct embeddings for snippets and then insert them into Weaviate. You can read more about the embedding process here. We also wrote a high level guide to this project here.

Run it yourself

The full, end-to-end version of this project involves a number of services and workflows:

1. A **Vite/React frontend**, to allow users to search for Wikipedia snippets and construct analogies via vector search
2. A **Weaviate database on Weaviate Cloud Services**, to store the Wikipedia snippets and their embeddings and to run both text and vector searches
3. A **serverless Weaviate database client on Modal**, to listen for requests from app clients, run the search logic, and communicate with the database
4. A **vector embedding service on Modal serverless GPUs**, to embed the Wikipedia snippets as vectors
5. An **ingestion workflow on Modal**, to download the Wikipedia dataset, embed it, and send the results to Weaviate

To make setup easier, we make it possible to run the search via a read-only client of our Weaviate database and run the app locally, which lets you skip the vector embedding and ingestion steps.

Set up a Python environment

Set up a Python environment however you like and then install `modal` with

```
pip install modal==0.62.140
```

Because Modal runs all of your code in cloud containers, you don't have to worry about any other dependencies!

If you don't already have a Modal account, get started with `modal setup`.

Deploy a serverless, read-only Weaviate client with Modal

Next, we set up a Weaviate client on Modal that reads from a Weaviate database that has already ingested and indexed the Wikipedia data.

Add `WCS_URL=https://gzimzbmdr6ycxyja715rsa.c0.us-west4.gcp.weaviate.cloud` and `WCS_RO_KEY=tUeQG12AkFLBY9SYOWVh2y00hZ25yu8va0UP` to a `modal.Secret` called `wiki-weaviate`.

Then, run the following command from the repo root to create a database client on Modal.

```
modal deploy backend.database
```

This client is *serverless*, meaning it scales automatically with load, including scaling down to zero instances when there is no load. That means you only need to pay for the compute resources you use!

You can run queries against the database from your local machine to test the client logic, for example

```
modal run backend.database::WeaviateClient.query --q='Albert Einstein'
```

or you can hit the API directly from your browser or with a tool like `curl` or Postman.

```
curl https://modal-labs--modal-weaviate-query.modal.run\?q\=Albert%20Einstein
```

Note that you should replace `modal-labs` in the URL with your Modal username!

This will return a large JSON object with a big vector of floating point numbers attached, so you might want to pipe it through `jq` or another JSON formatter:

```
curl https://modal-labs--modal-weaviate-query.modal.run\?q\=Albert%20Einstein \
 | jq . results\[0\].content
```

Optional: Embed and index Wikipedia yourself

If you’d like to run the entire pipeline yourself, there are several additional steps.

▶ Click here to reveal them.

Note that ingesting and indexing Wikipedia takes several hours! We **highly recommend you proceed with the read-only version first**.

Run the React frontend locally

Ensure you have a recent version of Node.js and `npm` installed. See the instructions here.

To set up the environment, run `npm install` in the `frontend` directory.

Create a file called `.env` in the `frontend` directory and set the value of `VITE_MODAL_WORKSPACE` to the name of your Modal workspace (by default, your GitHub username). See `.env.example` for the format.

Now, to run a hot-reloading, local version of the frontend, execute

```
npm run dev
```

and navigate a browser to the URL provided, which should be something like `http://localhost:5173`.

Optional: Serve a hot-reloading backend

You can also run the backend with hot reloading by using `modal serve` instead of `modal deploy`:

```
modal serve backend.database
```

This backend is still hosted on Modal, but it will automatically reload when you make changes to the code.

It uses a different URL than the deployed version, with a `-dev` appended just before `.modal.run`.

You can configure your frontend to use this backend by setting the `VITE_DEV_BACKEND` environment variable in `.env` to `true`.

Optional: Deploy the React frontend

If you’d like to share your own version of this app, you’ll need to host it somewhere.

We took advantage of Vercel’s excellent support for React apps to deploy directly from the GitHub repository.