README
Yoga MAIT Project for COSI 132a
Justin Lewman, Timothy Obiso, Ben Soli, Anastasiia Tatlubaeva

API Key

The use of this project requires an API key for OpenAI. There will be an empty file in the submission folder entitled `open_api_key.txt` that needs to contain the API key at runtime.

Project Set Up

Along with the code files, we provided a `requirements.txt` that lists all packages necessary to run our code. To install these packages, navigate to the directory that `requirements.txt` is in and run `pip install -r requirements.txt`

First, create the index in Elasticsearch. Make sure you have Elasticsearch version 7.10.2, [found here (https://www.elastic.co/downloads/elasticsearch)](https://www.elastic.co/downloads/elasticsearch) installed. Then, navigate to the location of the Elasticsearch folder and run `./bin/elasticsearch`
This starts the Elasticsearch server on your machine locally.

Next, we will start the embedding server. In another terminal window, run `python -m embedding_service.server --embedding sbert --model all-mpnet-base-v2`
This must be done before creating the index because embeddings are used in our Elasticsearch index.

Next, we will load the index into Elasticsearch. In another terminal window, run `python load_index.py --index_name poses --poses_folder_path data`
This command uses the JSON file (provided in the code portion) and loads it into Elasticsearch. This process may take a few minutes.

These steps are also explained in `scripts.sh`

Running Yoga MAIT

Make sure that (1) the Elasticsearch server is running, (2) the Embedding server is running, (3) the index has been loaded into ES via the steps above, and (4) that an API Key has been given in `open_api_key.txt`.
Once these steps are done, you can start Yoga MAIT by running `python3.11 app.py` and going to 127.0.0.1:5000 in your browser.

Corpus creation

All the code used for creating our initial data - `yoga_pose_data.json` - and scraping the images can be found in the folder `corpus_creation`. All data will be provided alongside with the code so there is no need to run this code.

Corpus preprocessing

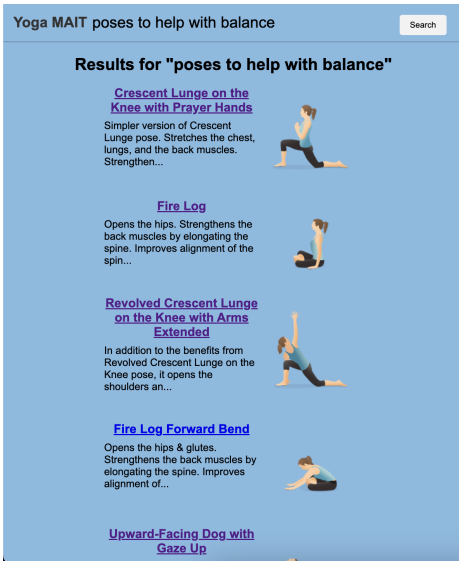
We did some preprocessing to the original data to make sure it included all information we needed for Elasticsearch index. The code used for preprocessing can be found in `preprocessing.py` but does not need to be run because `data_updated.json` - data after preprocessing - will be provided in the folder `data`.

If you encounter any difficulties running the code or downloading any packages, please feel free to email any member of our group.

Supplementary material:

All queries are run through the following pipeline:
Search (From User) -> Classifier -> GPT -> Elasticsearch -> Flask/UI (To User)

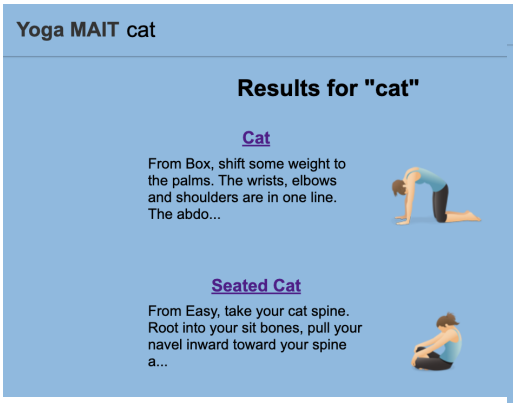
Query 1 - poses to help with balance



This is classified as “benefits” and fed into ChatGPT. Elasticsearch is then used to compare the cosine similarity between the embeddings of the “benefits” section of each pose. The results are then displayed on the page. All of these results are great for helping with balance. Our search pipeline was also able to discern that we did not want to “show off” our balance ability (such as “Sage Visvamitra's”) and provided poses related to *improving* balance.

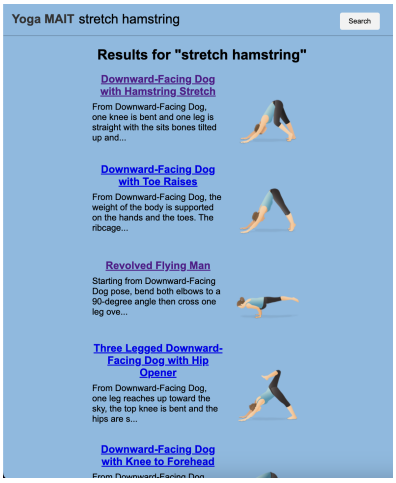
Queries 2, 3 and, 4 - cat; feline; sitting cat

Our classifier classifies each query as “name”, “benefits”, or “description”. If a query is classified as “name”, a direct keyword search based on the query is tried. If that fails, the query is embedded and cosine similarity is used to compare to other embedded query names. The query “feline” is classified as “name” but there is no pose called “feline”. Next, the embedding of “feline” is used. As we expect, the poses “cat” and “seated cat” appear near the top with other similar poses below. The first two results of “sitting cat” are “seated cat” and “cat”. This shows our system’s use of embeddings allows for fuzzy search. On the product-side, this means our app is beginner-friendly.



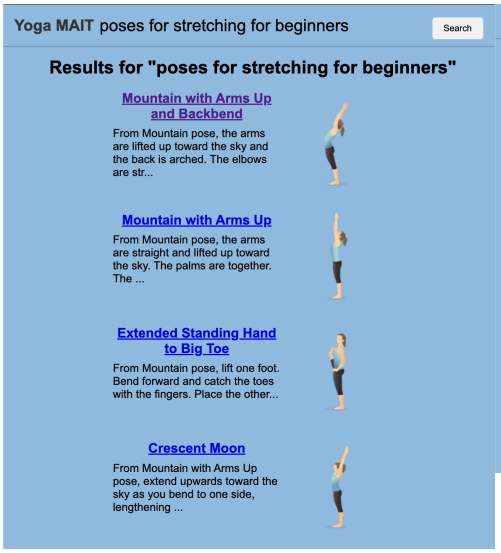
Query 5 - stretch hamstring

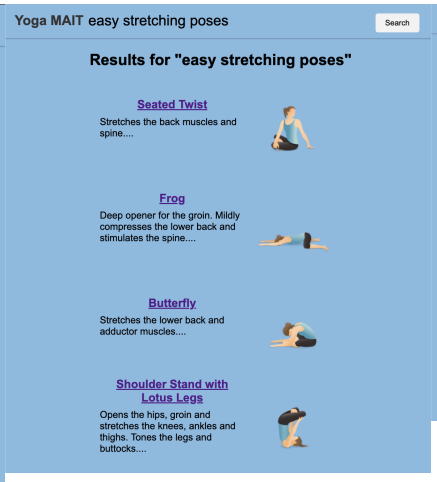
One area our model is very good at is identifying poses which stretch certain body parts. For example, searching “stretch hamstring” gives a lot of good hamstring (and leg) poses.

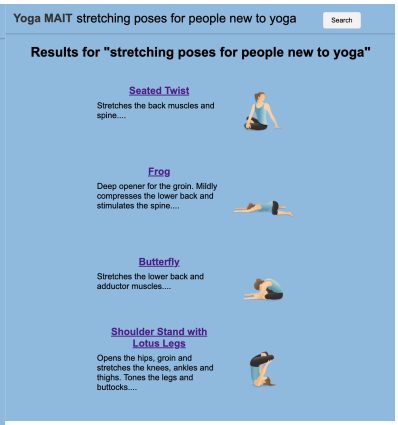


Queries 6 and 7 - poses to help with balance for beginners; easy stretching poses; stretching poses for people new to yoga

One feature that could be added in a future version of Yoga MAIT is the ability to sort by difficulty or rule out certain difficulty levels. Looking at the results of this test search, we can see that most of these poses are great for beginners. However, these results can only be obtained with specific searches. For example, these three results do not al lreturn the same results (when they should).

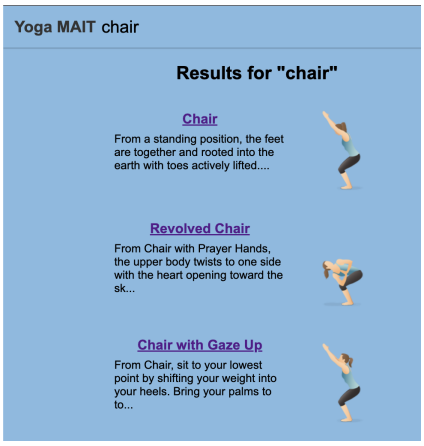






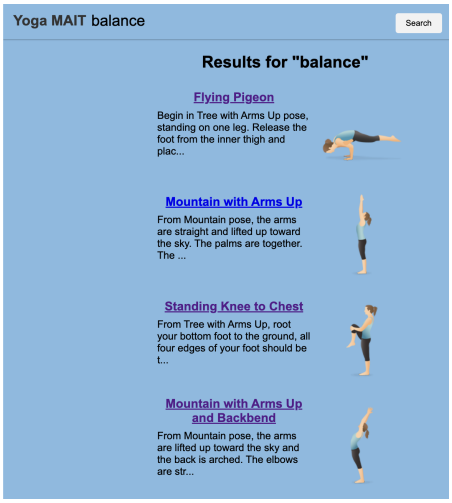
Query 8 - chair

Using the query “Chair” we can test the keyword matching and classifier detection of a pose name. We can see this example correctly identifies it as a name and includes all poses with “chair” in the title. This is a common and well known pose so it's important that these are accurate. There are many more relevant results returned that are not shown in the image.



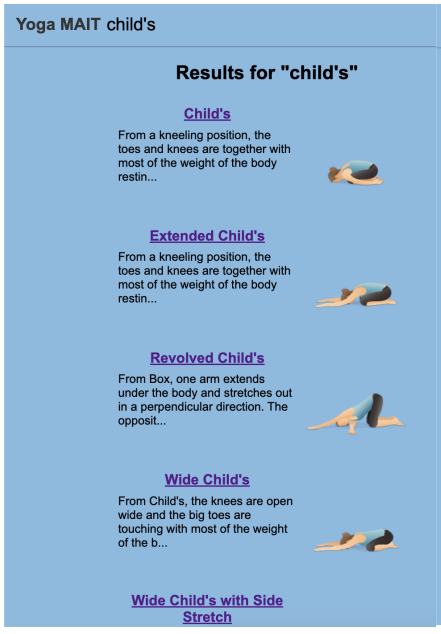
Query 9 - balance

Additionally, despite using ChatGPT our search engine is still much more accurate with short non-natural queries like: “Balance.” As you can see on the left, using a singular word gave us much more relevant results. These poses all focus heavily on leaning and balancing which are more beneficial for helping balance than just strengthening your core.



Queries 10-11 - child’s pose; child’s

We ensured that these searches would return the same results. We ran into an issue where “child’s” would be classified as a name but “child’s pose” would be classified as “description”. To fix this, we classify the query with the string “ pose” removed from the end of it. This can be seen by the identical results for each search. (While it does still say “child’s” on the page, that is because we edit the query itself



when we remove the string of characters.)

If the user already has a pose in mind, the embeddings do a great job helping a person find that pose; however, a longer query often results in less accurate poses. For example, searching “pose where you sit and reach for your legs” is a simple description of “fire log forward bed.” We can see that the pose does appear in the front page of results as expected. Unfortunately, it's also at the end of the page under a few unrelated poses. These are likely a consequence of the embeddings. (They have short descriptions and lots of keywords like legs and reach, but don’t quite fit the information request.)

Overall the model provided satisfactory results most of the time. While improvements can still be made with regard to longer queries, searching for a specific pose, searching by body part, and searching by name all return relevant results. If time allowed, we would have liked to return full routines as results. While this is not built into our system, that functionality still somewhat exists by clicking through the transition pages available for most poses.