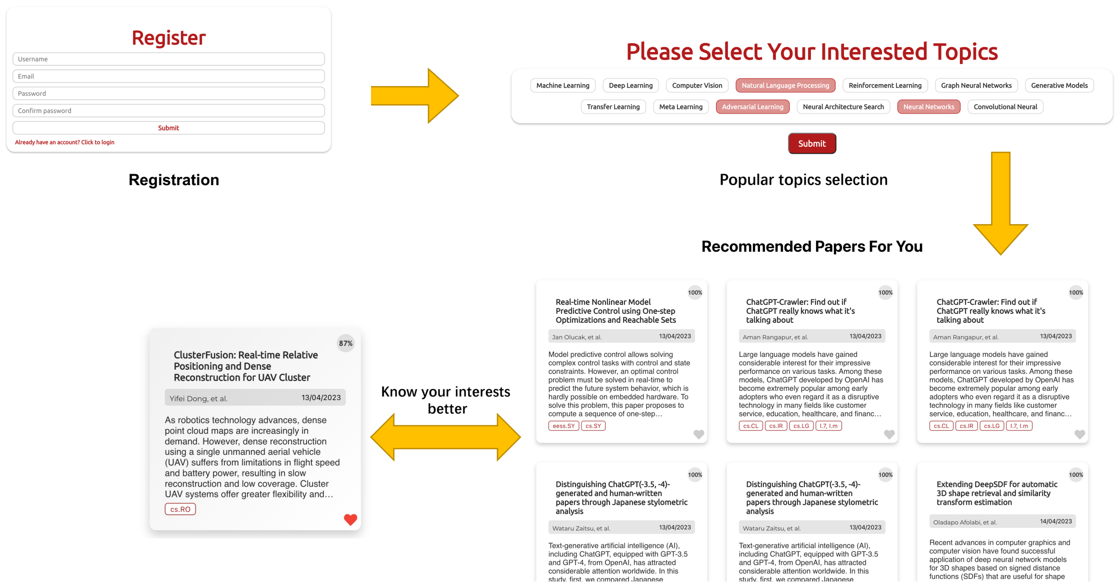
YourArXiv.com

April 2023

1 Features



87%
ClusterFusion: Real-time Relative Positioning and Dense Reconstruction for UAV Cluster
Yifei Dong, et al. 13/04/2023
As robotics technology advances, dense point cloud maps are increasingly in demand. However, dense reconstruction using a single unmanned aerial vehicle (UAV) suffers from limitations in flight speed and battery power, resulting in slow reconstruction and low coverage. Cluster UAV systems offer greater flexibility and…
cs.RO

100%
Real-time Nonlinear Model Predictive Control using One-step Optimizations and Reachable Sets
Jan Olucak, et al. 13/04/2023
Model predictive control allows solving complex control tasks with control and state constraints. However, an optimal control problem must be solved in real-time to predict the future system behavior, which is hardly possible on embedded hardware. To solve this problem, this paper proposes to compute a sequence of one-step…
eess.SY cs.SY

100%
ChatGPT-Crawler: Find out if ChatGPT really knows what it's talking about
Aman Rangapur, et al. 13/04/2023
Large language models have gained considerable interest for their impressive performance on various tasks. Among these models, ChatGPT developed by OpenAI has become extremely popular among early adopters who even regard it as a disruptive technology in many fields like customer service, education, healthcare, and financ…
cs.CL cs.IR cs.LG I.7, I.m

100%
ChatGPT-Crawler: Find out if ChatGPT really knows what it's talking about
Aman Rangapur, et al. 13/04/2023
Large language models have gained considerable interest for their impressive performance on various tasks. Among these models, ChatGPT developed by OpenAI has become extremely popular among early adopters who even regard it as a disruptive technology in many fields like customer service, education, healthcare, and financ…
cs.CL cs.IR cs.LG I.7, I.m

100%
Distinguishing ChatGPT(-3.5, -4)-generated and human-written papers through Japanese stylometric analysis
Wataru Zaitsu, et al. 13/04/2023
Text-generative artificial intelligence (AI), including ChatGPT, equipped with GPT-3.5 and GPT-4, from OpenAI, has attracted considerable attention worldwide. In this study, first, we compared Japanese

100%
Distinguishing ChatGPT(-3.5, -4)-generated and human-written papers through Japanese stylometric analysis
Wataru Zaitsu, et al. 13/04/2023
Text-generative artificial intelligence (AI), including ChatGPT, equipped with GPT-3.5 and GPT-4, from OpenAI, has attracted considerable attention worldwide. In this study, first, we compared Japanese

100%
Extending DeepSDF for automatic 3D shape retrieval and similarity transform estimation
Oladapo Afolabi, et al. 14/04/2023
Recent advances in computer graphics and computer vision have found successful application of deep neural network models for 3D shapes based on signed distance functions (SDFs) that are useful for shape

- A paper recommendation system based on arXiv database
- Natural language search functionality using keyword matching and semantic analysis
- Bookmarking feature to save interesting papers for future reference
- Personalized recommendations based on user's bookmarked papers
- Natural language processing techniques to extract topics and keywords from bookmarked papers
- More accurate and relevant paper recommendations based on user's interests

2 Application Architecture

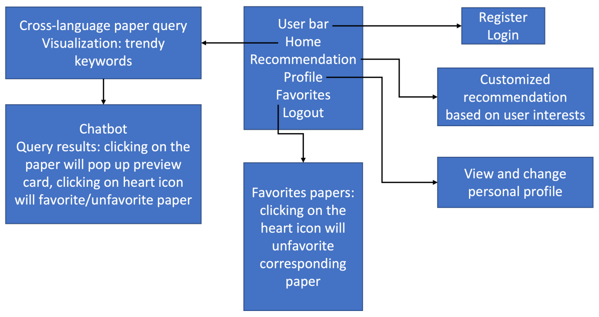

Cross-language paper query
Visualization: trendy keywords
Chatbot
Query results: clicking on the paper will pop up preview card, clicking on heart icon will favorite/unfavorite paper
User bar
Home
Recommendation
Profile
Favorites
Logout
Favorites papers: clicking on the heart icon will unfavorite corresponding paper
Register
Login
Customized recommendation based on user interests
View and change personal profile

Figure 1: User Interaction

The web application consists of five primary components, which interact with each other to provide the necessary functionality, including background processes:

- A Flask-based backend for API hosting
- A React-based front-end that interacts with users and the backend server
- Lambda function-based background processes triggered by time to collect the latest papers and generate embeddings
- An S3-based file system for storing the embedding dataset used for similarity calculations
- A DynamoDB-based database for managing user information and paper metadata

2.1 Paper Database schema and interaction pattern

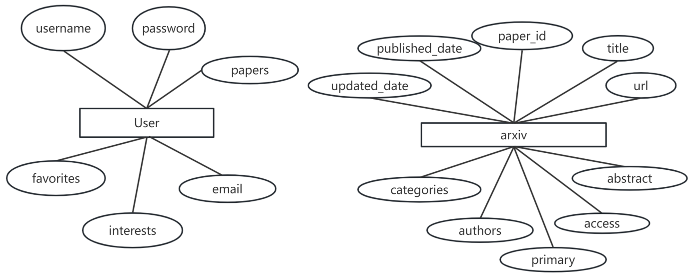


Figure 2: Database schema

To efficiently and scalably store our paper metadata and embedding information, we utilized two AWS services, DynamoDB and S3, in our project. While DynamoDB's table format structure enables easy querying of the paper metadata using *scan* or *query* functions, we opted to store the embedding information in S3 due to its scalability and cost-effectiveness. Retrieving the embedding information from DynamoDB would become slow as the number of papers grows. With this approach, DynamoDB is responsible for serving data lookup requests, while S3 serves as the foundation of our recommendation system. By leveraging both services, we ensure that our recommendation function is supported, enabling our users to receive personalized suggestions based on their interests.

2.2 User Database schema and interaction

New users choose their topics of interest upon registration and are then redirected to the login page. The user interface includes a personal recommendation page with a list of arXiv papers tailored to the user's interests. Users can add or remove papers to their favorites list, which will be used to extract interests and recommend corresponding latest arXiv papers. The user bar in the upper right corner allows users to navigate to various pages, such as the search page, recommendation page, favorite papers list, personal profile page, or log out. Flask framework is used for the backend and provides endpoints for both frontend and backend interactions, with API routes handle registration, login, interest selection, recommendations, adding/removing favorites, and updating user profiles.

2.3 Background process and Lambda functions

We have implemented the data collection process using AWS Lambda service. We rely on the public arXiv paper database to query the latest papers from the internet. To ensure our database is updated daily, we set up a trigger using AWS CloudWatch Events through Amazon EventBridge. This trigger is designed to activate our Lambda function every day, keeping the database up-to-date with the latest papers.

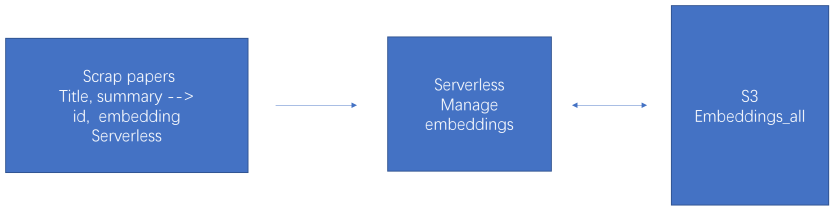
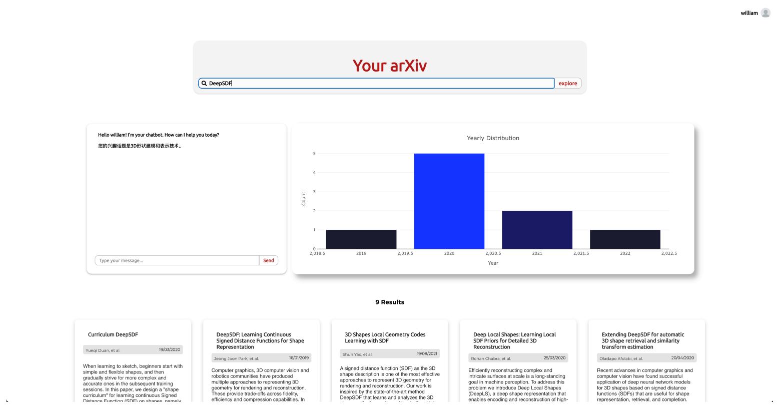


Figure 3: Lambda-based paper collector

We do not have updating database functionality in our web application. So the user won't trigger this function. This function will only be triggered by CloudWatch and will only run on the background which won't consume resources from our web application.

2.4 Search Engine backend and frontend design

We developed the front-end page using the widely-used React framework, which offers a solid basis for building responsive and dynamic user interfaces. To improve our data visualization capabilities, we integrated the powerful Three.js and Plotly libraries, which enable us to create informative and interactive visualizations with ease. Additionally, to enhance the user experience and personalize it, we use OpenAI's GPT-4 API for conversational AI and paper abstract understanding.



Hello william! I'm your chatbot. How can I help you today?

您的兴趣话题是3D形状建模和表示技术。

Curriculum DeepSDF
Yueqi Duan, et al. 19/03/2020
When learning to sketch, beginners start with simple and flexible shapes, and then gradually strive for more complex and accurate ones in the subsequent training sessions. In this paper, we design a "shape curriculum" for learning continuous Signed Distance Function (SDF) on shapes, namely

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation
Jeong Joon Park, et al. 16/01/2019
Computer graphics, 3D computer vision and robotics communities have produced multiple approaches to representing 3D geometry for rendering and reconstruction. These provide trade-offs across fidelity, efficiency and compression capabilities. In

3D Shapes Local Geometry Codes Learning with SDF
Shun Yao, et al. 19/08/2021
A signed distance function (SDF) as the 3D shape description is one of the most effective approaches to represent 3D geometry for rendering and reconstruction. Our work is inspired by the state-of-the-art method DeepSDF that learns and analyzes the 3D

Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction
Rohan Chabra, et al. 25/03/2020
Efficiently reconstructing complex and intricate surfaces at scale is a long-standing goal in machine perception. To address this problem we introduce Deep Local Shapes (DeepLS), a deep shape representation that enables encoding and reconstruction of high-

Extending DeepSDF for automatic 3D shape retrieval and similarity transform estimation
Oladapo Afolabi, et al. 20/04/2020
Recent advances in computer graphics and computer vision have found successful application of deep neural network models for 3D shapes based on signed distance functions (SDFs) that are useful for shape representation, retrieval, and completion.

Figure 4: Search engine with chat bot

2.5 Embedding caching strategy

The semantic embedding database is presently stored in the cloud-based S3 storage solution, with updates to the local copy at the backend server occurring once every 2-hour interval. This approach has been adopted to mitigate the time spent on each download process.

3 Cost Model

The cost for the domain rental fee and Cloudflare DNS service is approximately $20 per year. So there will be $10 for 6 months.

There are several AWS services that we need to take into account. They are S3, Lambda, CloudWatch and DynamoDB.

Based on our research we need $0.023 per GB-month for the first 50 TB in the US East (N. Virginia) region. For now, our database has 8300 papers in only 108.3 MB. Thus after 6 months updating our database. The database will only reach 480.65MB. So the price of S3 within 6 months should be less than $0.06.

For the Lambda function which will only be executed once for each day, the maximum memory usage for each runtime is restricted to 500MB. The maximum time usage for each runtime is set to 150 seconds. So for 6 months usage the total cost for lambda should be around $0.06.

The Cloudwatch service charges for storing the logs. We set the reserve period of logs to 30 days. Then in each month there should be around 0.01GB of log data. The log for 6 months lambda usage should be around 0.01GB so the cost for $0.03.

We assume that all of our users are students or researchers who want to go through different kinds of papers. For each day they want to check 3 to 5 papers. Then before they find their papers, they may need to send multiple requests to our application to query the papers from our database. They may also check the abstraction of one paper to see if it matches their goals. We assume they will need to query 20 times before finding one paper. So the total number of requests that one user can have in a day should be 100 maximum. In each request we will provide 30 results, therefore, 30 read operations will be counted. We take each paper's information as 1KB, so each time we will transfer a data with size 30KB.

When we have 10 users, the total number of requests will be 1000 per day. Then within a month the total cost of usage in transferring data will be 0.03GB per day and 0.9GB per month. In this case, we transfer data from the US to CA the price will be $0.02 per GB for the first 10TB. Thus, the price will be $0.018 and the price will be $0.108 for 6 months. When it comes to 1,000 users, the usage will be 90GB per month and the total price for 6 months will be $10.8. When we luckily have 1,000,000 users, we will have to transfer 90TB for the data. In this case we may need to upgrade our DynamoDB for higher ability and the transfer price after the first 10TB will be $0.01 per GB. Thus the price for transferring the data will be about $1000 per month and the total cost for 6 months will be more than $6000.

4 Latency and throughput test

4.1 Latency performance

The web application's latency is tested with the following configuration:

- database size: 1200 papers / 8300 papers
- query same title and abstract during the test
- 50 iterations, 10 requests in one iteration

During our testing the latency of our application, we choose to measure our performance over 5000 requests. When we have a query abstract request at the beginning, the latency is quite high. This is because every time we restart our application, we need to download the embedding information from S3 to our system cache. Thus after downloading the file, the latency drops greatly. We also notice that the latency for querying the title and querying the abstract do not increase as the number of requests increases. This may be due to the optimization that we apply using the Numpy library to quickly calculate the matrix result. When querying the abstract, the latency is higher than querying the title. This is due to the different measures that we treat these requests. When we want to query the title, we just need to compare the title which takes less effort. However when we want to compare the abstract, we need a longer time to calculate the similarities. Lastly, as we increase the number of items which need to be considered when calculating the similarities, our performance in latency remains stable and just increases less than 0.1 ms.

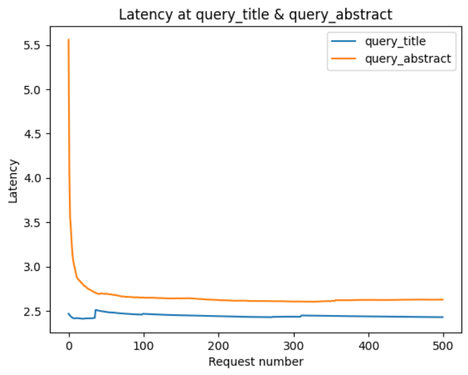


Figure 5: Latency with small database

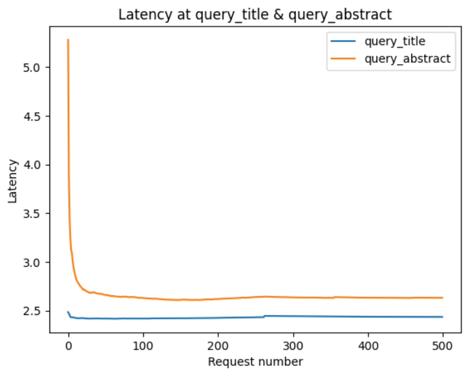


Figure 6: Latency with large database

4.2 Throughput performance

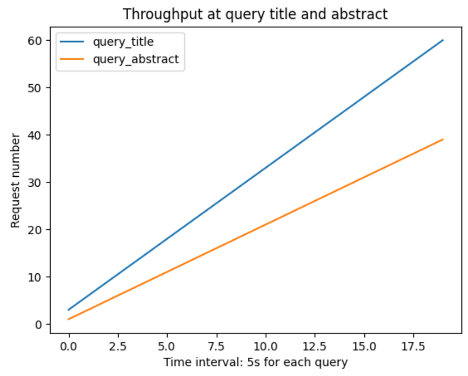


Figure 7: Throughput with small database

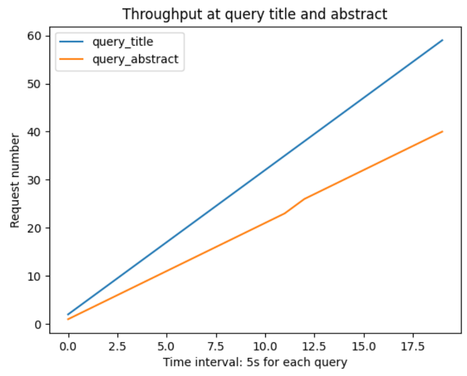


Figure 8: Throughput with large database

The web application's throughput is tested with the following configuration:

- database size: 1200 papers / 8300 papers
- query same title and abstract during the test
- 20 iterations, 10s requests in one iteration

We have 20 iterations for testing the throughput, in each iteration, we count the number of success responses for both query requests within 5 seconds. During our testing, we find that our performance in throughput remains stable as we increase the size of our database. The total number of successful responses does not change too much based on the graphs. In both databases, the query title requests have a higher throughput than querying the abstract. This may be due to the different measures that we used to query them. As we need longer time finding the best abstract, the slope of querying the abstract is smaller than querying the title.