A research publication recommendation web service using hexagonal architecture and RAKE algorithm

INHYUP KIM, UVU, USA
NEIL HARRISON, UVU, USA

A considerable number of research manuscripts are published every year detailing the new findings on various topics. Due to the large number of publications, researchers spend a notable amount of time and effort in searching through various journals and scientific database services to identify relevant publications. For lay persons interested in new information and topics, it may be particularly more challenging to stay up-to-date on the latest findings. To reduce these issues, I propose a search-based web service that recommends research publications to users to minimize the inconvenience of having to evaluate all possible relevant works. The main goal of this project is to implement the microservices architecture and recommendation algorithms to provide better quality research information to users.

In this paper, I describe to create a search-based recommendation web service using a hexagonal architecture that can increase efficiency and scalability of the software. I also illustrate the content-based filtering using RAKE algorithm, one of the natural language processing algorithms, to recommend research publications that are highly relevant to user's search history and saved research paper preference data. This service is deployed through AWS to provide users with an elastic service. It also uses DynamoDB, a NoSQL database service provided by AWS, to dynamically store the information and data.

The results of this project show better search results than existing research search websites (IEEE, ACM and ArXiv) and provide convenience to users that recommends research through a recommendation system. This project is meaningful because it utilizes the software architecture, machine learning, big data, algorithm and network that I have learned in graduate studies. The applications created for this project is a server and a client, with a total of 3500 lines of code.

Additional Key Words and Phrases: Hexagonal architecture, Recommendation Systems, Content-Based Filtering, NLP, RAKE

1 INTRODUCTION

In the past decade, the number of science and engineering journal articles and conference papers increased by 4% per year, with over 2.6 million publications[31]. While research search engines, such as Google Scholar, have functions to refine search preferences, researchers continue to spend a significant portion of their research time looking for the most appropriate publications that to set a framework for their projects. There are several explanations for such difficulties in conducting a literature review. For one, some search engines do not return satisfactory search results. Another reason may be that the filtering function is restrictive or inconvenient to use. Even when a search engine has a feature to save a user's preferred topics, it is difficult to expect utilities beyond the saved preferences. The proposed search-based service will provide search capabilities and improved user experience to replace these limitations of the search engines currently in service.

The current project uses a hexagonal architecture, a type of microservices architecture, to design a research publication search and recommendation service. Also, it uses RAKE algorithm to recommend more accurate search results and publications based on search keywords and each user's saved research articles. The hexagonal architecture was introduced by Alistair Cockburn and is also referred to as the ports and adapters architecture [8]. The hexagonal architecture, unlike the traditional layered architecture, is designed to follow the principles of object-oriented programming by separating the business logic from external technologies and following dependency inversions of each layer[8] [32]. The hexagonal architecture consists of the domain and the layers. The domain is the heart of the application and responsible

Authors' addresses: Inhyup Kim, UVU, Orem, UT, USA, 106280588@uvu.edu; Neil Harrison, UVU, Orem, UT, USA, Neil.Harrison@uvu.edu.

for business logic, and the layers surround the domain to handle the facilitation and management of objects in the domain model. Only ports and adapters are used to interact with the inside of the application[12]. This approach has the advantage of implementing only ports and adapters without modification of the business logic, even if technical details change.

Recommendation systems have become very common in recent years. They are already closely integrated into the everyday life in video content providers (e.g., Netflix and YouTube), as well as through social media and web-based recommendations on food, travel, shopping, and more. These recommendation systems provide convenience and satisfaction to customers by recommending products that they would be interested in based on big data rather than having to browse through all the options. Therefore, a recommendation system is applied to recommend research articles that might be of an interest to users based on the user's metadata (search history and saved data by a user) in this project. Filtering is used to increase the accuracy of the recommendation system. A content-based filtering in this project is a method that recommends other items that have similar content or have a special relationship to each other. I propose to use the rapid automatic keyword extraction (RAKE) algorithm as a method of content-based filtering. The RAKE algorithm is an unsupervised, domain-independent, and language-independent method for extracting keywords on individual documents that measures the relative importance of words that are considered important in documents or a specific sentence[27]. The algorithm calculates the importance of the user's metadata (the users' search keywords and the title of the research paper saved by the user) and finds a combination of the most important words to recommend the most relevant articles.

The program consists of a server and a client. The number of lines of code is 2,000 for the servers and 1,500 for the clients. This service operates on AWS EC2, and the database uses DynamoDB. In this project, the research publication external API used ArXiv API due to the ease of accessibility and availability.

The results of the project have established a recommendation system that is more effective than traditional search services. I have developed a service that is easy to maintain and highly scalable using the hexagonal architecture. This improves on accuracy of search results by analyzing key keywords and recommending papers that users may be interested in through RAKE algorithm. The paper is organized in the following order: discussion of related work in Section 2, description of the software architecture and implementation in more detail in Section 3, and discussion of the project results in Section 4. Finally, the conclusion is presented in Section 5.

2 RELATED WORKS

The hexagonal architecture is described in detail in the books [32] [25]. The hexagonal architecture is a type of microservices architecture. In [10] and [5], microservices are an alternative for developing complex and distributed applications, solving scalability and making online services easier to maintain. Software architecture can be broadly divided into monolithic and microservices architecture. Studies [11] and [3] compare monolithic and microservices architectures and describe the pros and cons.

Monolithic architecture refers to a traditional architecture, in which all components of software are integrated into one project [24]. All processes are tightly coupled and run as a single service. The advantage of monolithic is it is easy and simple to develop and test for small projects. However, as the project grows, the code becomes cluttered and difficult to modify because it is closely related to each other, and continuous integration and continuous deployment become difficult.

Amazon, Netflix, LinkedIn, Spotify and other companies have evolved their applications towards a microservice architecture [10][30]. Microservice architecture is an architectural pattern in which one large application is divided

into smaller units of service. Microservices architecture is becoming an alternative to traditional software development paradigms for developing complex and distributed applications, such as building mobile applications[33] as well as cloud platforms [13]. The advantage of microservices is that each service can be developed independently, and it is easy to modify and maintain. It is also independently deployable and scalable, and its internal configuration is not affected by external changes. However, the complexity of the system increases and a method of communication between services is required. Also, because it can be vulnerable to security threats. The researches[14], [28] are being conducted on this.
The hexagonal architecture was first introduced by Alistair Cockburn[8], and these books[32], [25] point out the limitations of the layered architecture and show how to use the hexagonal architecture. [2] and [32] show examples of successful applications of the hexagonal architecture in NodeJS and Java.
Recommendation systems have been successfully applied in several other areas, including news[6], movies[15], and audio[29]. In [9], the authors consider content-based filtering, collaborative filtering, demographic filtering, Knowledge-Based recommender system, and hybrid recommender system as a personalized recommendation system. Among them, collaborative filtering and content-based filtering, which are used most successfully in the recommendation system, were selected as candidates for this project.
Companies such as Amazon use collaborative filtering[19], a recommendation system that automatically predicts users' interests according to taste information obtained from many users. The fundamental assumption of the collaborative filtering approach is that the past trends of users will remain the same in the future[22]. The system is not limited to specific user information, but it uses information collected from many users to measure similarity and recommend items. It has the advantage of being able to give appropriate recommendations to users even if metadata is not provided. However, collaborative filtering is not suitable for use in this project because it makes recommendations based on the evaluation of many users, thus compromising on the individualized preferences of a user.
One of the chronic problems of collaborative filtering is the cold start[23]. Collaborative filtering can be used only when sufficient interaction metadata between users and items is provided. When a new product or a customer is added, it may not be possible to make a recommendation because there is not relevant information.
Therefore, this project uses content-based filtering. This method compares the features between items and recommends other items with similar features. In this approach, an item profile is created by extracting a user's preferences then the item profiles are compared using natural language processing to recommend the most similar items. Content-based filtering is a better choice when a new application is built since it can be applied using only the data available at all stages of the system. A common strategy used in content-based filtering is to rank the most important keywords using TF-IDF[4]. However, the RAKE algorithm is implemented to rank keywords in this project. This study[27] proves that RAKE is faster and efficient to extract the keywords. In [26], scientific article keywords extraction was conducted using RAKE.

3 SOFTWARE ARCHITECTURE AND IMPLEMENTATION

In this section, I discuss in depth how to implement Research Advisor, a research paper recommendation web service that I developed. This section is divided into two subsections, and the first subsection discusses the software architecture, and the second subsection discusses the filtering algorithm, the RAKE algorithm.

3.1 Architecture

The hexagonal architecture, also known as the port and adapter architecture, successfully separates the business logic from the external components.

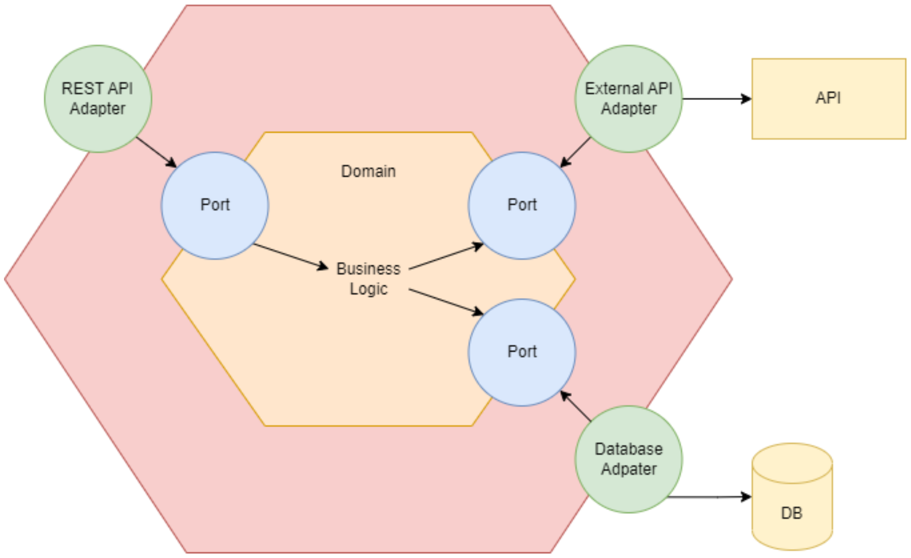


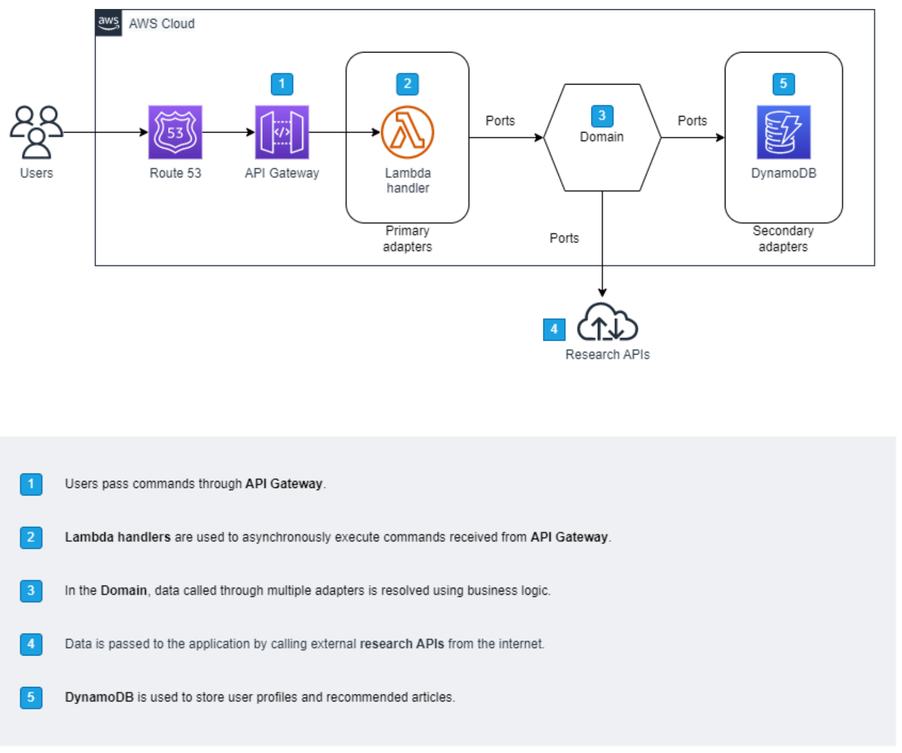
Fig. 1. The hexagonal architecture used in the project

In Figure 1, the hexagonal shape represents the layers between components. Each component is independent and communicates with other components in different layers only through ports and adapters. The domain inside the hexagon is responsible for the recommendation system, which is a business logic of this application. The recommendation system will be discussed in more detail in Section 3.2.

The port is a component that connects a domain and the outside of the domain. The interfaces are defined on the port to interact multiple services. These interfaces invert dependencies so that higher-level components such as domain, do not depend on lower-level components[25].

Each adapter provides its own services. The REST API adapter located on the left side of Figure 1 is an inbound adapter that forwards user requests to the domain. In this project, the REST API adapter consists of user controller and researchpaper controller. The user controller is responsible for creating and modifying the user data, and the researchpaper controller delivers commands to request the research publications. The external API adapter and database adapter, located on the right, are outbound adapters and are responsible for performing tasks requested by the domain. The external API adapter connects the API calls requested by the domain to the API service. The database adapter only works with databases. Database operations such as creation, deletion, and modification requested by the domain are processed by calling the database through the database adapter.

The application of this project is deployed through Amazon EC2. I built EC2 for a server and a client and use t2.micro, 1GB ram, 6 CPU and 30GB hard disk. Amazon EC2 provides an elastic virtual cloud computing environment and an elastic IP address to make it easily accessible from the web[21].



1 Users pass commands through **API Gateway**.

2 **Lambda handlers** are used to asynchronously execute commands received from **API Gateway**.

3 In the **Domain**, data called through multiple adapters is resolved using business logic.

4 Data is passed to the application by calling external **research APIs** from the internet.

5 **DynamoDB** is used to store user profiles and recommended articles.

Fig. 2. The architecture on AWS

The users' requests that passed through the API Gateway are forwarded to the domain asynchronously from the lambda handler. The domain executes the filtering algorithm that will be detailed in Section 3.2. The external API is the ArXiv API to send a research article request call and return the result. ArXiv has more than 2.1 million scientific and technical papers, which are constantly updated and easily accessible. Figure 2 shows that the database used AWS DynamoDB to store user information. Amazon DynamoDB is a fast, flexible NoSQL database service for all applications of any size that consistently requires less than 10 milliseconds of latency. In addition, it is a fully managed cloud database that stores data in the form of key-value, which makes fast to read data and increased scalability. Amazon DynamoDB features a flexible database schema. The data items in the table do not have to have the same number of attributes. Because the project uses various properties such as string, datetime, list, and dictionary, any information that is not easy to implement in RDBMS can be easily organized into a single table in the DynamoDB.

3.2 Recommendation system

The most important and the core feature of the application is the domain since the hexagonal architecture follows a domain-driven design[16]. The domain refers to the problem area that the software wants to solve, and the core problem in this project is the research paper recommendation system. The domain of this application is illustrated in the following figure.

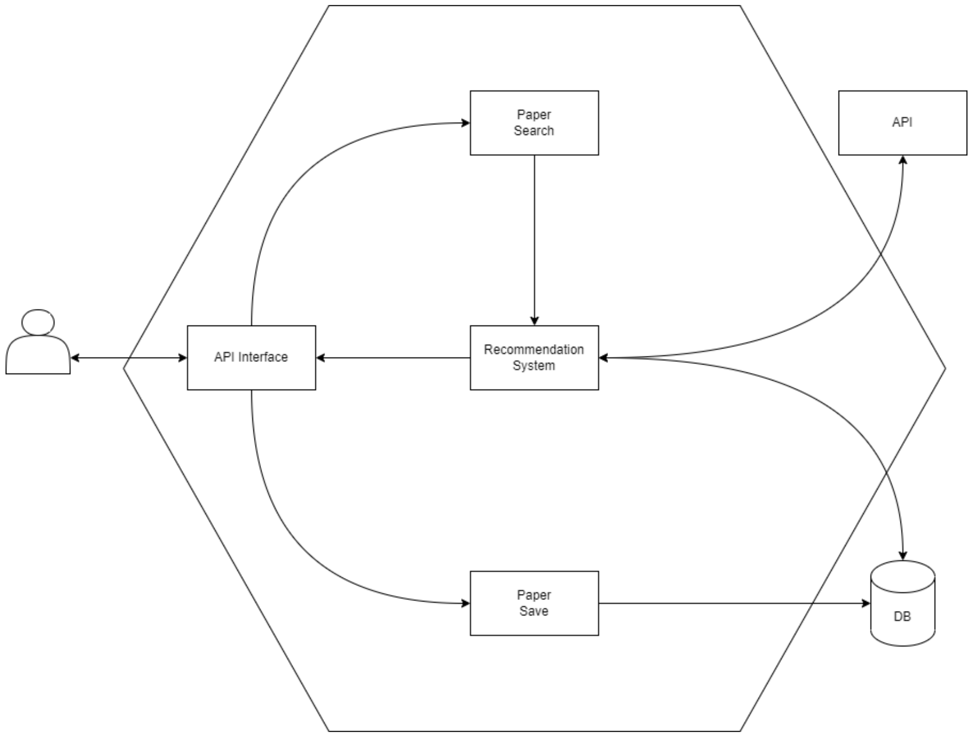


Fig. 3. The domain in the hexagonal architecture

In Figure 3, the hexagon frame represents the domain and the squares represent the components. The arrows indicate how the components interact. When a user interacts with the system through the interface, such as searching for or saving a research paper, the data sent by the user is stored in the database. The recommendation system makes API calls asynchronously by extracting the keywords through the natural language processing based on this stored data. The system creates a list of recommended papers using the response of API calls.
Natural language processing (NLP) is a branch of artificial intelligence that uses machine learning to process and interpret text and data[7]. The recommendation system of this project uses content-based filtering because it needs to identify the properties of a research paper and recommend it to the users. Since the metadata of a research paper consists of texts, selecting meaningful information among these texts is the key. Therefore, this application extracts keywords from the research papers through NLP and utilizes these extracted keywords to recommend the most similar research papers. Rapid automatic keyword extraction (RAKE) algorithm is used as the filtering algorithm. The algorithm excludes words that are only grammatically significant and takes important words in the corpus. In the RAKE algorithm, corpus is first filtered out with a stop word list such as ’a’, ’and’ and ’is’ or other words with minimal lexical meaning. Stop words are generally considered uninformative or meaningless and are not included in various text analyses. For example, if a user enters the phrase ’I like to eat an apple and a banana’, ’I’, ’like’, ’to’, ’an’, ’a’ and ’and’ are removed

because those are included in the list of stop words. It is based on the expectation that these stop words are used too frequently and extensively to help users in analysis or search operations[27]. The words 'eat', 'apple' and 'banana' are candidate keywords because they convey meaning. The candidate keyword group is then established using parameters such as the minimum number of strings that can be considered as a word, the maximum number of words included in the phrase, and how many times it appears in the entire document. The RAKE algorithm considers the co-occurrences of these filtered candidate words to be considered important words and measures their scores in the following way.

(1) Count the frequency of each word.
    For example, $freq(Apple) = 7$ and $freq(Banana) = 5$ in table 1. In this case, the word 'Apple' has higher score because 'Apple' co-occurs in the document more than 'Banana'
(2) When individual words are bound to a candidate group in the form of a phrase, the degree is calculated by adding the frequency of each word to the number of times it is used with other words.
    $degree(Apple) = 7 + 2 + 5 + 1 = 15$
    $degree(Banana) = 5 + 5 + 1 + 1 = 12$
(3) Calculate the ratio of (1) and (2). The ratio of a phrase is calculated by adding the value of the ratio of each word included in the phrase. This ratio is very important to create a keyword list.
    $ratio(Apple) = 15/7 \approx 2.14$
    $ratio(Banana) = 12/5 \approx 2.4$
    $ratio(AppleBanana) = 159/35 \approx 4.54$
    In this scenario, the word combination 'Apple Banana' has the highest score and is more likely to be a candidate word.
(4) One-third of the total number of individual words is regarded as the total number of keywords on the list. For example, if there are 28 words in total, 28/3, and about 9 words correspond to meaningful keywords.

	Apple	Chicken	Pants	Banana	Car	Hat	Pizza	Bike	Train
Apple	7		2	5				1	
Chicken		5			2		4	2	
Pants	2		6			5			
Banana	5			5			1		1
Car		2			6			3	2
Hat			5			5			
Pizza		4		1			4		
Bike	1	2			3			5	2
Train				1	2			2	3

Table 1. Example of RAKE matrix

The RAKE algorithm proves excellent performance in extracting the essential keywords, but there is a drawback to applying it to this project. The RAKE algorithm performs better on long text than on short titles because keywords must occur twice in the same order within the document. Because users usually do not enter an extensive list of search words, almost all of the search words except for the stop words are recognized as keywords. Therefore, I propose to use a combination of these keywords to further improve the performance of the recommendations. First, the system finds the the case containing all n keywords. Then, it finds the combinations of words by decreasing r by 1 until r equals 1.

The binomial theorem states that

$$\sum_{r=0}^{n} \binom{n}{r} x^r = (1+x)^n \tag{1}$$

Putting $x = 1$ gives

$$\sum_{r=0}^{n} \binom{n}{r} = 2^n \tag{2}$$

$$\sum_{r=1}^{n} \binom{n}{r} = 2^n - 1 \tag{3}$$

This approach not only finds papers containing all the keywords, but also recommends related papers through the keyword combinations. However, this creates a bottleneck problem that the number of API calls increases exponentially as the number of keywords increases. Equation (3) describes how the number of API calls increase by 2^n as the keyword increases by n. When all cases of the combination are called by API, a total of $2^n - 1$ calls are made. Therefore, the number of API calls and the amount of data called at a time should be restricted to solve this issue.
At first, the appropriate number of data should be set when requesting an API call. A large set of returned results can cause a significant load on the server and require a long time to render. According to the ArXiv API manual[1], 30000 API calls take a little over 2 minutes or more. This means it returns approximately 250 data per second. Therefore, the 250 data are returned when making an API call.
According to [20], slow system response times lead to user dissatisfaction, which is seen in a study[17] that states that users report increased level of intolerance around the 12-second response time. Theoretically, it was expected that one API call takes one second, but the response time is within 12 seconds when 7 API calls are requested as shown in Fig 4. A slight delay occurred as duplicate data were deleted among all the returned result data.
As shown in Figure 4, as the number of API calls increases, the returned result data increases. However, the API called the later returned results using only one or two keyword combinations, not all keyword combinations, resulting in a decrease in the relevance to the keyword entered by the user.
Based on these observations, the program uses 7 API calls to return at most 12 seconds response time. Therefore, the program can consistently return over 1000 research paper data in a given time.

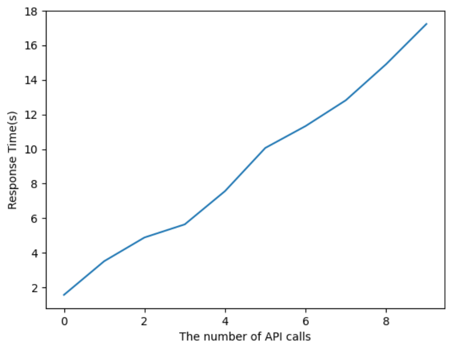


Fig. 4. Graph to response time per API call

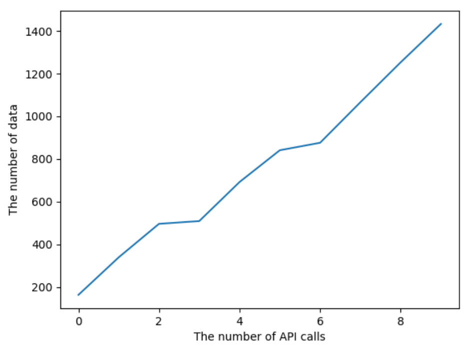


Fig. 5. Graph to the number of data per API call

The extracted keywords using the RAKE algorithm are asynchronously stored in the database in the form of a map. When a user enters search words, the system increments the value corresponding a keyword to track the user's keyword history. Also, users can save the research papers. The paper saved by the user are considered to have higher priority than the searched keywords. Therefore, the keywords extracted from the title of the research paper adds a weight of 7 to the search history.

The two message sequence diagrams below provide a more detailed explanation of how the logic works.

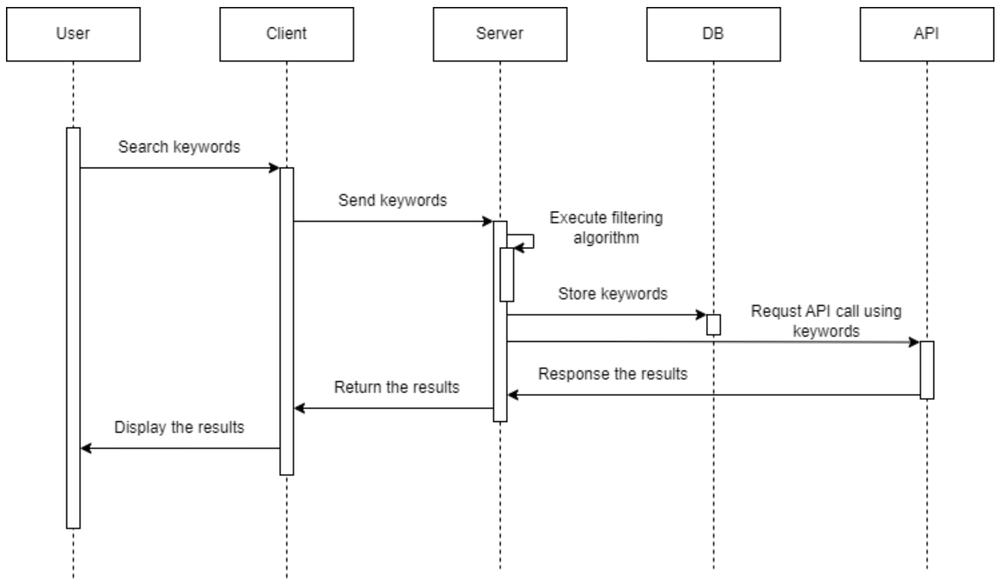


Fig. 6. The search message sequence diagram

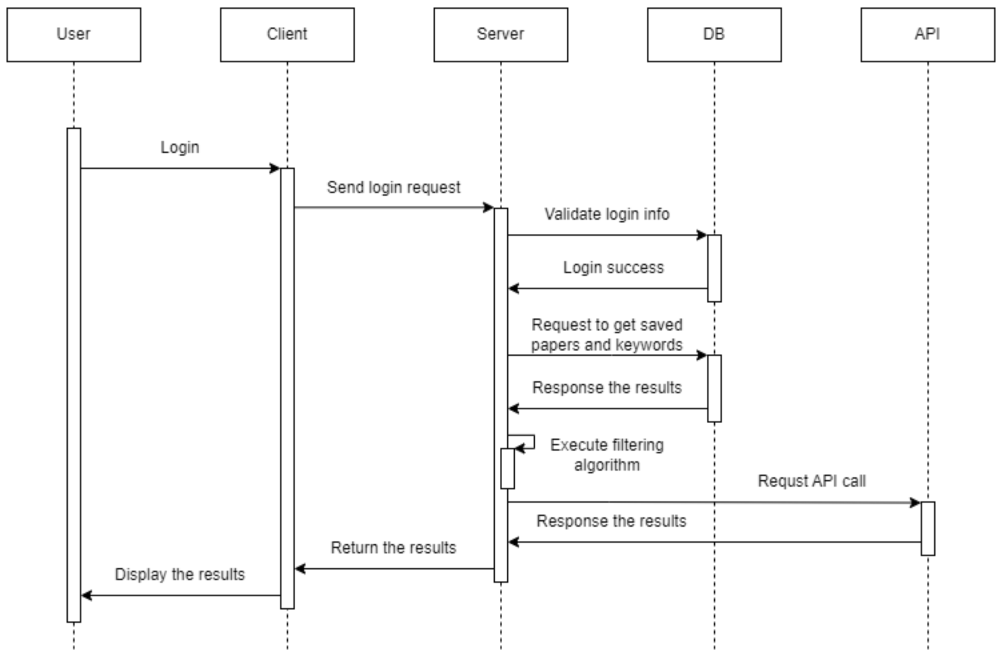


Fig. 7. The login message sequence diagram

4 PROJECT RESULTS

A sample of this project is available at http://ec2-44-240-38-221.us-west-2.compute.amazonaws.com/. I evaluate the two aspects of this project: soft architecture and filtering algorithms.

4.1 Software Architecture

The hexagonal architecture, which compensates for the weaknesses of the existing layered architecture, shows great advantages in program scalability and maintenance. Each component of the system is independent, which means a change in one component does not affect the other. As such, the hexagonal architecture makes for a reliable service.

4.2 Filtering Algorithm

The RAKE algorithm showed the performance of selecting keywords within 4 milliseconds. It helps to improve the search speed by effectively and quickly identifying keywords even in the corpus.
Normalized Discounted Cumulative Gain (NDCG) is used as an indicator to evaluate the performance of the recommendation algorithm[18]. NDCG is an indicator that evaluates the performance by giving more weight to the order of the recommendations and is especially useful in situations where a higher ranking list is significantly more important than a lower ranking list. Cumulative Gain (CG) is the sum of relevance scores. The relevance score is a score that indicates how much a user prefers each recommended item. This project focuses on the association between the keywords and the research titles. Relevance score is set on three levels: *completely relevant*, *somewhat relevant*, and *not relevant*.

$$CG_n = \sum_{i=1}^{n} relevance_i \tag{4}$$

Discounted Cumulative Gain (DCG) originates from CG. The lower the order of the recommended items, the larger the denominator, so that the overall DCG is less affected. However, there is a limitation in that accurate performance evaluation is difficult when the number of recommended items between domains is different.

$$DCG_n = \sum_{i=1}^{n} \frac{relevance_i}{\log_2(i+1)} \tag{5}$$

NDCG is the application of normalization to DCG to compensate for the limitations of DCG.

$$NDCG_n = \frac{DCG}{IDCG} \tag{6}$$

IDCG is the DCG value when the best recommendation is made. In conclusion, NDCG is an index indicating how accurate the recommendation list of the current model is compared to the most ideal combination of the recommendations. By normalizing the DCG value, NDCG has a value between 0 and 1.
The DCG scores were measured for the top 100 results of each information source, IEEE Xplore (IEEE), ACM Digital Library (ACM), ArXiv (ArXiv) and this project application (ResearchAdvisor). The average NDCG scores were obtained using 5 keywords for each information source. Table 2 shows the NDCG score of each information source.

Research search service	NDCG Score
IEEE	0.75
ACM	0.67
ArXiv	0.58
Research Advisor	0.77

Table 2. NDCG score of research search service

In comparison to the traditional search engines, the results obtained from in this project application demonstrate higher NDCG scores in favor of the Research Advisor. However, it is hard to argue that the scores in the table 2 is valid. First of all, the number of sample data is too small to produce meaningful results. If the minimum sample size criterion is not met, the information is insufficient to correctly judge results. It is possible to make errors that the result value is changed by another variable or that it can judge the results are different even though data is same. It is important to determine the appropriate sample size for the purpose of the study, because too large a sample size would be an excessive waste of resources, and on the contrary, too small a sample size to produce significant results would be a waste of effort. There is no clear sample size that is universally applicable in all situations, but this study did not proceed with an appropriate sample size, so it is insufficient to prove the validity of the results. There is also a possibility that the results of the experiment may have been distorted due to the biased selection of keywords. Therefore, it is necessary to conduct research by selecting more keywords data fairly to obtain more reasonable and validity results.

Despite the possibility of errors in these scores, the significance of the scores in the table 2 is that they outperformed other search engines in certain areas. As an example, if the search words were entered into the ArXiv search engine to find a paper called 'Network Data', the user's desired result is not displayed top of the list. When searching for 'Network Data' on the Research Advisor, the desired result is displayed on the top.

There were a few challenges during the development of this project. A bottleneck during the data transference was expected. To prevent this, I attempted to download and preprocess the entire ArXiv database to reduce the network latency. Contrary to my expectation, the data preprocessing was more expensive, time intensive, and resulted in loss of a notable amount of important data. ArXiv provides the entire data as a Json file (https://www.kaggle.com/datasets/Cornell-University/arxiv). In the process of serializing the Json file, the memory is overflowed, and to prevent this, the need to delete the abstract of the research paper came to the fore. This was recognized to be a huge loss in conveying information to users. Also, retrieval of information performed worse than API calls. There was a limitation because it uses only title information to search. Moreover, a major modification of the algorithm was required in order to catch more detailed information than when calling the API.
For these reasons, I looked for ways to make the API calls faster. To solve the bottleneck of API calls, I decided to request an appropriate amount of data at a time and exclude any combination of keywords that can lead to results that are too broad. With these changes, the result is returned within 1200 milliseconds at most. I also included a loading image to indicate to the user that the system is processing.

5 CONCLUSION

Recommendation systems reduce the effort required for users to find research papers related to topics of interest. The problems with current search engines serving this function are that the search results are not satisfactory, they are not easy to use, and the stored data is not useful. As a way to solve this problem, I proposed a recommendation system using a hexagonal architecture, implemented an application, and deployed it to Amazon EC2.
Hexagonal architecture is designed to remove the dependencies of the existing layered architecture and increase scalability. The RAKE algorithm is the core filtering algorithm of this project for implementing the recommendation system. It measures importance as a ratio of frequency to degree based on co-occurrence of keywords in the corpus. However, there was a disadvantage that the sentence length was not long enough to use the RAKE algorithm. I solved this problem through combinations of words. In addition, the algorithm is modified to limit the API calls to reduce the latency so that the wait time is shortened after requesting a command.
This project can further develop are in the following areas: (1) filtering algorithm and (2) using multiple APIs. Since the RAKE algorithm was published in 2010, many efficient filtering algorithms have been developed since then. I expect to produce better results in short sentences using improved filtering algorithms. I also think it is possible to derive improved search results by connecting more research APIs. Since the data provided by ArXiv is limited to science and technology fields, it is difficult to find papers related to the humanities and sociology. If other APIs can be connected, the program's scalability can be improved and more services can be provided.
Although this project is less than the number of codes required, it can be considered a master's project. The fact that the length of the code is not long proves that the application is efficiently designed. In addition to implementing the application, I conducted comparative studies with other services by objectively analyzing them using evaluation indicators of recommendation systems such as NDCG. This recommendation service shows the higher NDCG score than

existing services has been built. This verifies this application is effective and productive. In addition, this application is practically usable since it is deployed on the web using AWS cloud services. I applied the software architecture, NoSQL, machine learning, filtering algorithms, and cloud services I learned in graduate school to one project. Also, the issues that I encountered while developing this application have been overcome by researching several papers.

6 ACKNOWLEDGMENTS

Special thanks to Dr. Neil Harrison for his overall supervision and advice. I would also like to thank Ken Jensen for his advice and direction on AWS.

REFERENCES

[1] [n. d.]. ArXiv API user's Manual. https://arxiv.org/help/api/user-manual

[2] Yura Abharian. 2021. MECHANISMS FOR IMPLEMENTING HEXAGONAL ARCHITECTURE IN NODE JS. *Global Prosperity* 1, 3-2 (2021), 41–48.

[3] Omar Al-Debagy and Peter Martinek. 2018. A comparative review of microservices and monolithic architectures. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 000149–000154.

[4] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

[5] Elyas Ben Hadj Yahia, Laurent Réveillere, Yérom-David Bromberg, Raphaël Chevalier, and Alain Cadot. 2016. Medley: An event-driven lightweight platform for service composition. In *International Conference on Web Engineering*. Springer, 3–20.

[6] Krishna Bharat, Tomonari Kamba, and Michael Albers. 1998. Personalized, interactive news on the web. *Multimedia Systems* 6, 5 (1998), 349–358.

[7] KR1442 Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[8] Alistair Cockburn. 2005. Hexagonal architecture. *The Pattern: Ports and Adapters* (2005).

[9] Debashis Das, Laxman Sahoo, and Sujoy Datta. 2017. A survey on recommendation system. *International Journal of Computer Applications* 160, 7 (2017).

[10] Paolo Di Francesco. 2017. Architecting microservices. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 224–229.

[11] Konrad Gos and Wojciech Zabierowski. 2020. The comparison of microservice and monolithic architecture. In *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, 150–153.

[12] Jesse Griffin. 2021. Hexagonal-Driven Development. In *Domain-Driven Laravel*. Springer, 521–544.

[13] Dong Guo, Wei Wang, Guosun Zeng, and Zerong Wei. 2016. Microservices architecture based cloudware deployment platform for service computing. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, 358–363.

[14] Abdelhakim Hannousse and Salima Yahiouche. 2021. Securing microservices and microservice architectures: A systematic mapping study. *Computer Science Review* 41 (2021), 100415.

[15] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 230–237.

[16] Benjamin Hippchen, Pascal Giessler, Roland Steinegger, Michael Schneider, and Sebastian Abeck. 2017. Designing microservice-based applications by using a domain-driven design approach. *International Journal on Advances in Software* 10, 3&4 (2017), 432–445.

[17] John A Hoxmeier and Chris DiCesare. 2000. System response time and user satisfaction: An experimental study of browser-based applications. (2000).

[18] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.

[19] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[20] Fiona Fui-Hoon Nah. 2004. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology* 23, 3 (2004), 153–163.

[21] Simon Ostermann, Alexandria Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. 2009. A performance analysis of EC2 cloud computing services for scientific computing. In *International conference on cloud computing*. Springer, 115–131.

[22] Weike Pan and Li Chen. 2013. Cofiset: Collaborative filtering via learning pairwise preferences over item-sets. In *Proceedings of the 2013 SIAM international conference on data mining*. SIAM, 180–188.

[23] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[24] Chris Richardson. 2014. Pattern: monolithic architecture. *Posjećeno* 15 (2014), 2016.

[25] Chris Richardson. 2018. *Microservices patterns: with examples in Java*. Simon and Schuster.

[26] Komang Rinartha and Luh Gede Surya Kartika. 2021. Rapid Automatic Keyword Extraction and Word Frequency in Scientific Article Keywords Extraction. In *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*. IEEE, 1–4.

[27] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory* (2010), 1–20.

[28] Chaitanya K Rudrabhatla. 2020. Security Design Patterns in Distributed Microservice Architecture. *arXiv preprint arXiv:2008.03395* (2020).

[29] Upendra Shardanand and Pattie Maes. 1995. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 210–217.

[30] Poonam B Thorat, Rajeshwari M Goudar, and Sunita Barve. 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications* 110, 4 (2015), 31–36.

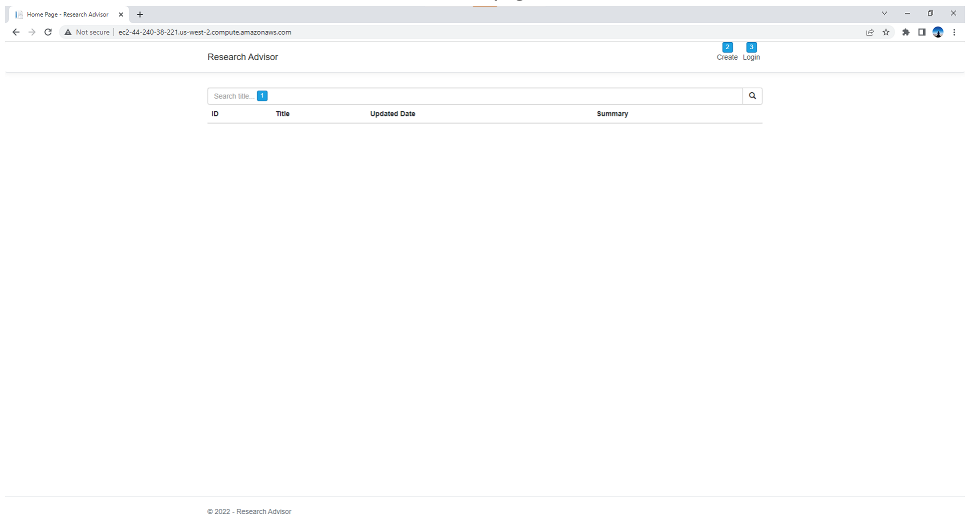
[31] Karen White. 2019. Publications Output: US Trends and International Comparisons. Science & Engineering Indicators 2020. NSB-2020-6. *National Science Foundation* (2019).

[32] Eberhard Wolff. 2016. *Microservices: flexible software architecture*. Addison-Wesley Professional.

[33] Yixue Zhao and Nenad Medvidovic. 2019. A microservice architecture for online mobile app optimization. In *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 45–49.

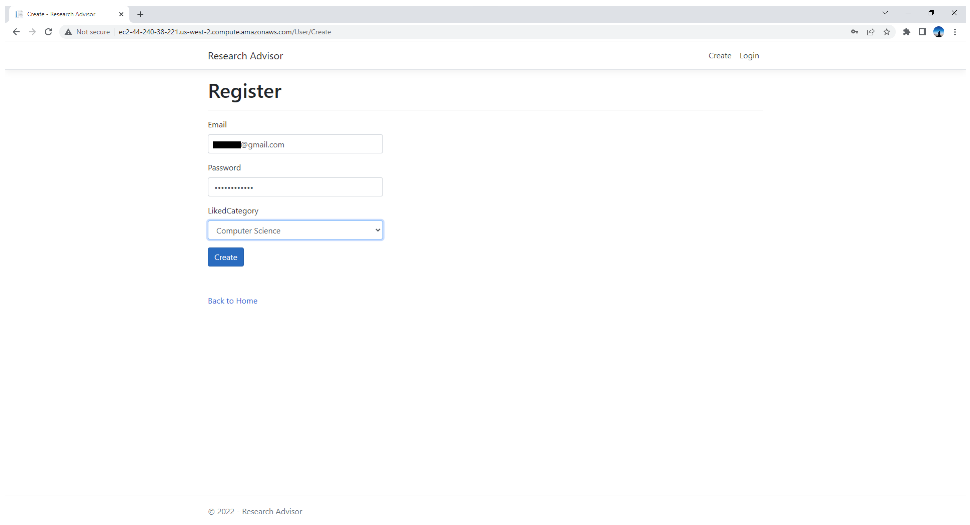
A USER’S MANUAL

Main page


Home Page - Research Advisor
Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com
Research Advisor
2
Create
3
Login
Search title...
1
ID
Title
Updated Date
Summary
© 2022 - Research Advisor

If you connect to http://ec2-44-240-38-221.us-west-2.compute.amazonaws.com/, user can see the main page as above. The main page provides the following features: (1) search research papers, (2) create account, and (3) login account.

Create account


Create - Research Advisor

Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com/User/Create

Research Advisor   Create   Login

Register

Email: [redacted]@gmail.com

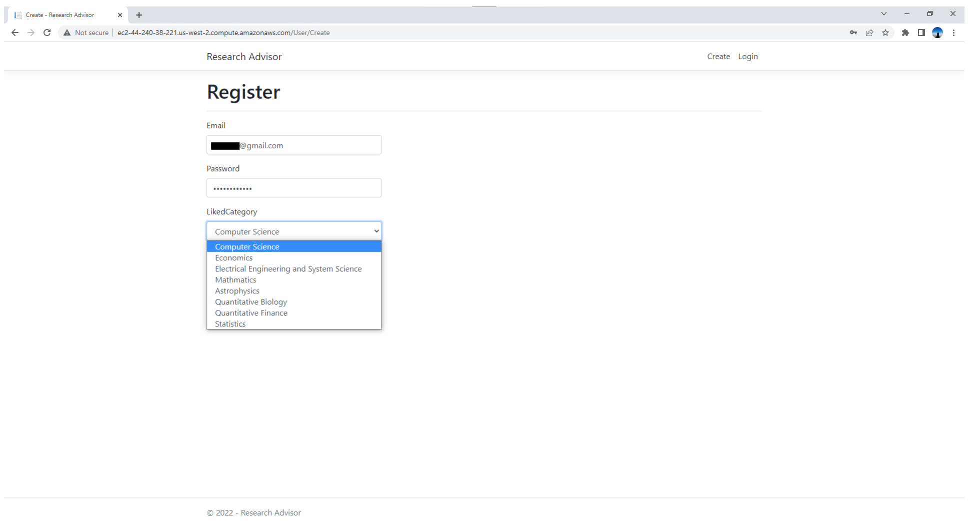
Password: ••••••••••••

LikedCategory: Computer Science

Create

Back to Home

© 2022 - Research Advisor


Create - Research Advisor

Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com/User/Create

Research Advisor   Create   Login

Register

Email: [redacted]@gmail.com

Password: ••••••••••••

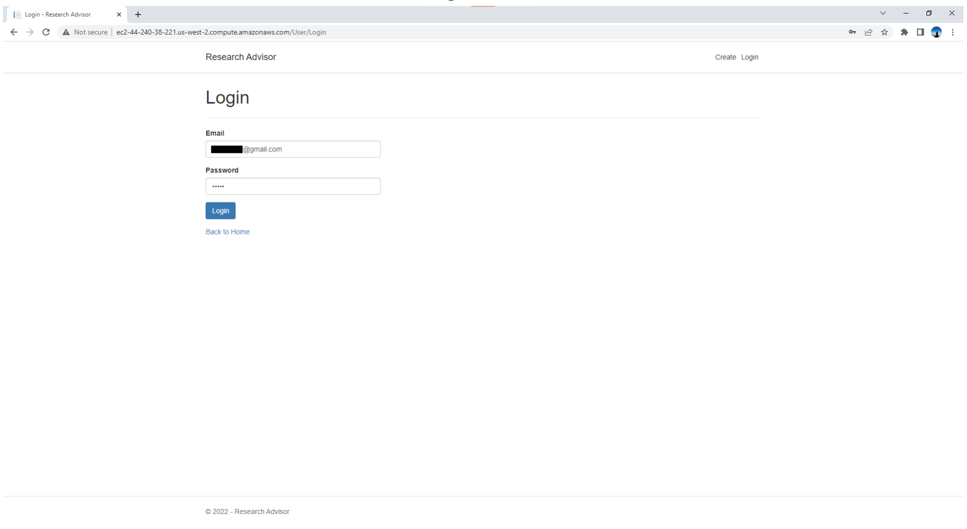
LikedCategory: Computer Science

- Computer Science
- Economics
- Electrical Engineering and System Science
- Mathmatics
- Astrophysics
- Quantitative Biology
- Quantitative Finance
- Statistics

© 2022 - Research Advisor

On the create account page, set email address, password and pick preferred category from the list. Then click the Create button.

Login account



Login - Research Advisor

Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com/User/Login

Research Advisor — Create Login

Login

Email: [redacted]@gmail.com

Password: •••••

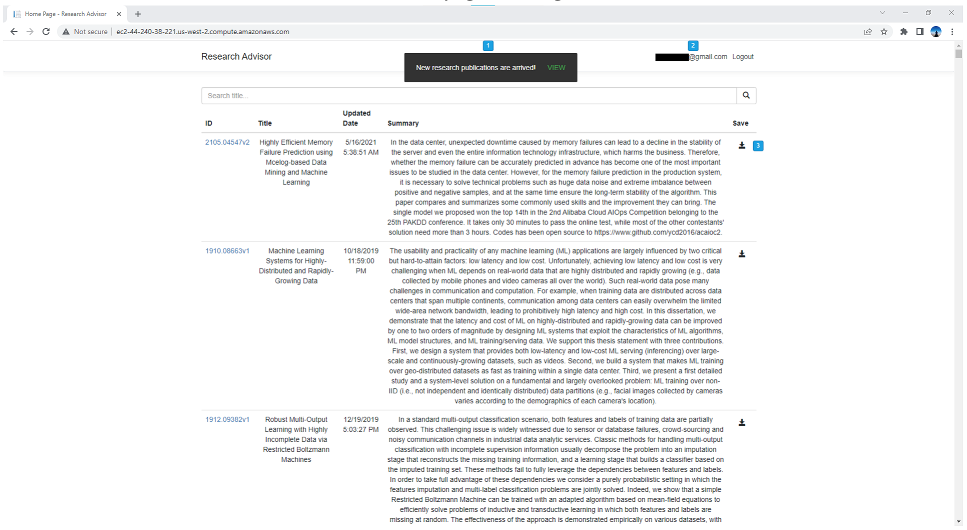
Login

Back to Home

© 2022 - Research Advisor

On the login page, enter your email and password. Click the Login button.

Main page after login



Home Page - Research Advisor

Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com

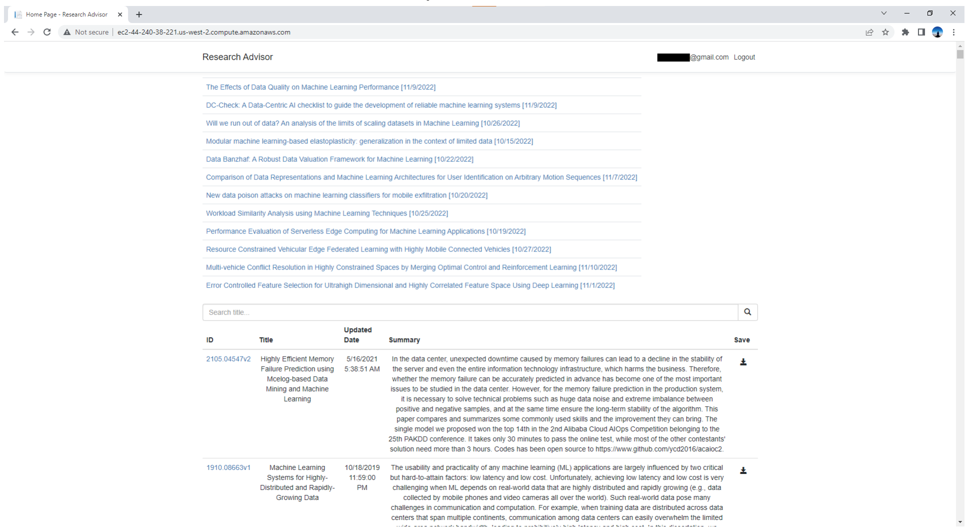
Research Advisor — 1 New research publications are arrived! VIEW — 2 [redacted]@gmail.com Logout

Search title...

ID	Title	Updated Date	Summary	Save
2105.04547v2	Highly Efficient Memory Failure Prediction using Mcelog-based Data Mining and Machine Learning	5/16/2021 5:38:51 AM	In the data center, unexpected downtime caused by memory failures can lead to a decline in the stability of the server and even the entire information technology infrastructure, which harms the business. Therefore, whether the memory failure can be accurately predicted in advance has become one of the most important issues to be studied in the data center. However, for the memory failure prediction in the production system, it is necessary to solve technical problems such as huge data noise and extreme imbalance between positive and negative samples, and at the same time ensure the long-term stability of the algorithm. This paper compares and summarizes some commonly used skills and the improvement they can bring. The single model we proposed won the top 14th in the 2nd Alibaba Cloud AIOps Competition belonging to the 25th PAKDD conference. It takes only 30 minutes to pass the online test, while most of the other contestants' solution need more than 3 hours. Codes has been open source to https://www.github.com/ycd2016/acaioc2.	3
1910.08663v1	Machine Learning Systems for Highly-Distributed and Rapidly-Growing Data	10/18/2019 11:59:00 PM	The usability and practicality of any machine learning (ML) applications are largely influenced by two critical but hard-to-attain factors: low latency and low cost. Unfortunately, achieving low latency and low cost is very challenging when ML depends on real-world data that are highly distributed and rapidly growing (e.g., data collected by mobile phones and video cameras all over the world). Such real-world data pose many challenges in communication and computation. For example, when training data are distributed across data centers that span multiple continents, communication among data centers can easily overwhelm the limited wide-area network bandwidth, leading to prohibitively high latency and high cost. In this dissertation, we demonstrate that the latency and cost of ML on highly-distributed and rapidly-growing data can be improved by one to two orders of magnitude by designing ML systems that exploit the characteristics of ML algorithms, ML model structures, and ML training/serving data. We support this thesis statement with three contributions. First, we design a system that provides both low-latency and low-cost ML serving (inferencing) over large-scale and continuously-growing datasets, such as videos. Second, we build a system that makes ML training over geo-distributed datasets as fast as training within a single data center. Third, we present a first detailed study and a system-level solution on a fundamental and largely overlooked problem: ML training over non-IID (i.e., not independent and identically distributed) data partitions (e.g., facial images collected by cameras varies according to the demographics of each camera's location).	
1912.09382v1	Robust Multi-Output Learning with Highly Incomplete Data via Restricted Boltzmann Machines	12/19/2019 5:03:27 PM	In a standard multi-output classification scenario, both features and labels of training data are partially observed. This challenging issue is widely witnessed due to sensor or database failures, crowd-sourcing and noisy communication channels in industrial data analytic services. Classic methods for handling multi-output classification with incomplete supervision information usually decompose the problem into an imputation stage that reconstructs the missing training information, and a learning stage that builds a classifier based on the imputed training set. These methods fail to fully leverage the dependencies between features and labels. In order to take full advantage of these dependencies we consider a purely probabilistic setting in which the features imputation and multi-label classification problems are jointly solved. Indeed, we show that a simple Restricted Boltzmann Machine can be trained with an adapted algorithm based on mean-field equations to efficiently solve problems of inductive and transductive learning in which both features and labels are missing at random. The effectiveness of the approach is demonstrated empirically on various datasets, with	

After logging in, it provides the following features: (1) view updated research papers last 30 days of recommendations based on the keywords they searched and papers they have saved (2) edit user's profile, and (3) save a research paper to profile.

View recent published research articles

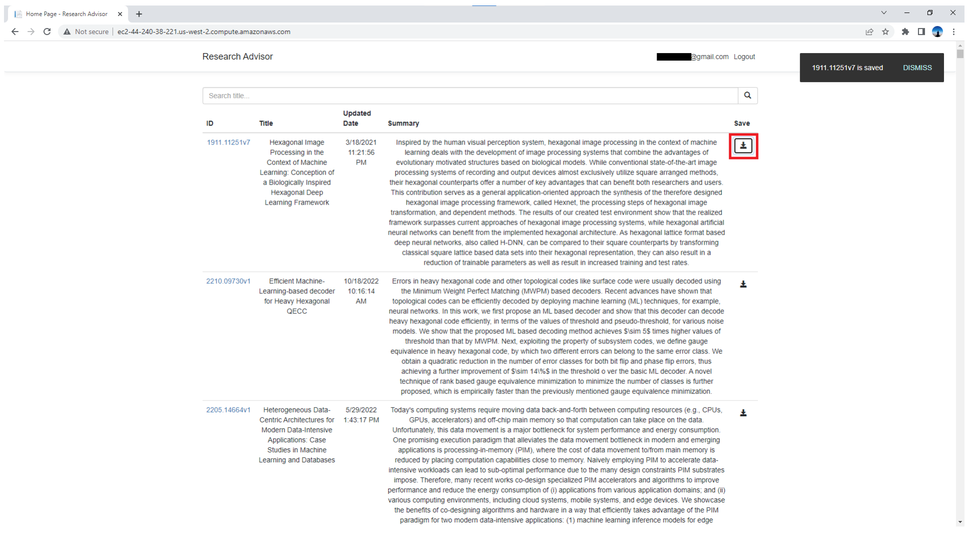

Home Page - Research Advisor
Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com
Research Advisor
[redacted]@gmail.com Logout
Search title...

The Effects of Data Quality on Machine Learning Performance [11/9/2022]
DC-Check: A Data-Centric AI checklist to guide the development of reliable machine learning systems [11/9/2022]
Will we run out of data? An analysis of the limits of scaling datasets in Machine Learning [10/26/2022]
Modular machine learning-based elastoplasticity: generalization in the context of limited data [10/15/2022]
Data Banzhaf: A Robust Data Valuation Framework for Machine Learning [10/22/2022]
Comparison of Data Representations and Machine Learning Architectures for User Identification on Arbitrary Motion Sequences [11/7/2022]
New data poison attacks on machine learning classifiers for mobile exfiltration [10/20/2022]
Workload Similarity Analysis using Machine Learning Techniques [10/25/2022]
Performance Evaluation of Serverless Edge Computing for Machine Learning Applications [10/19/2022]
Resource Constrained Vehicular Edge Federated Learning with Highly Mobile Connected Vehicles [10/27/2022]
Multi-vehicle Conflict Resolution in Highly Constrained Spaces by Merging Optimal Control and Reinforcement Learning [11/10/2022]
Error Controlled Feature Selection for Ultrahigh Dimensional and Highly Correlated Feature Space Using Deep Learning [11/1/2022]

ID	Title	Updated Date	Summary	Save
2105.04547v2	Highly Efficient Memory Failure Prediction using Mcelog-based Data Mining and Machine Learning	5/16/2021 5:38:51 AM	In the data center, unexpected downtime caused by memory failures can lead to a decline in the stability of the server and even the entire information technology infrastructure, which harms the business. Therefore, whether the memory failure can be accurately predicted in advance has become one of the most important issues to be studied in the data center. However, for the memory failure prediction in the production system, it is necessary to solve technical problems such as huge data noise and extreme imbalance between positive and negative samples, and at the same time ensure the long-term stability of the algorithm. This paper compares and summarizes some commonly used skills and the improvement they can bring. The single model we proposed won the top 14th in the 2nd Alibaba Cloud AIOps Competition belonging to the 25th PAKDD conference. It takes only 30 minutes to pass the online test, while most of the other contestants' solution need more than 3 hours. Codes has been open source to https://www.github.com/ycd2016/acaioc2.	⤓
1910.08663v1	Machine Learning Systems for Highly-Distributed and Rapidly-Growing Data	10/18/2019 11:59:00 PM	The usability and practicality of any machine learning (ML) applications are largely influenced by two critical but hard-to-attain factors: low latency and low cost. Unfortunately, achieving low latency and low cost is very challenging when ML depends on real-world data that are highly distributed and rapidly growing (e.g., data collected by mobile phones and video cameras all over the world). Such real-world data pose many challenges in communication and computation. For example, when training data are distributed across data centers that span multiple continents, communication among data centers can easily overwhelm the limited	⤓

When a user clicks the view from the previous step, the user can see the last 30 days of published papers according to the user's preference.

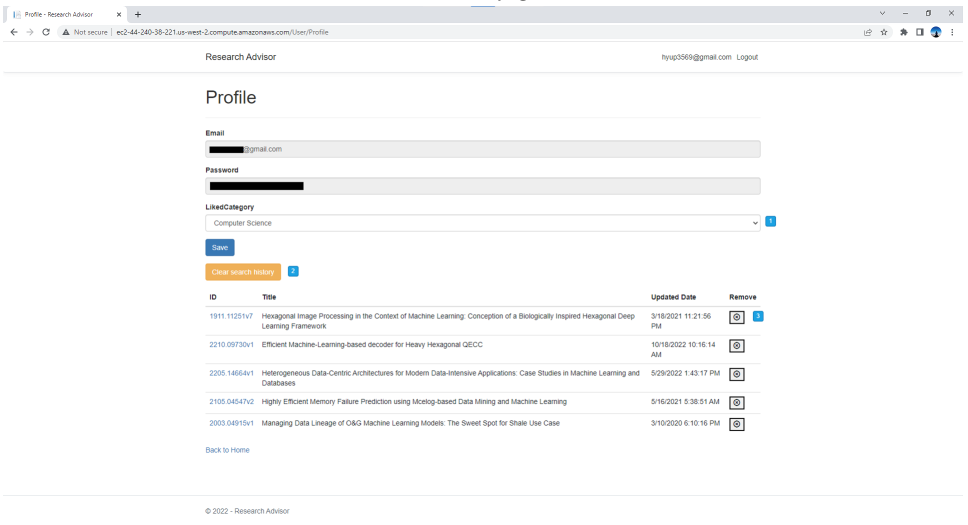
Save a research paper


Home Page - Research Advisor
Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com
Research Advisor
[redacted]@gmail.com Logout
1911.11251v7 is saved
DISMISS
Search title...

ID	Title	Updated Date	Summary	Save
1911.11251v7	Hexagonal Image Processing in the Context of Machine Learning: Conception of a Biologically Inspired Hexagonal Deep Learning Framework	3/18/2021 11:21:56 PM	Inspired by the human visual perception system, hexagonal image processing in the context of machine learning deals with the development of image processing systems that combine the advantages of evolutionary motivated structures based on biological models. While conventional state-of-the-art image processing systems of recording and output devices almost exclusively utilize square arranged methods, their hexagonal counterparts offer a number of key advantages that can benefit both researchers and users. This contribution serves as a general application-oriented approach the synthesis of the therefore designed hexagonal image processing framework, called Hexnet, the processing steps of hexagonal image transformation, and dependent methods. The results of our created test environment show that the realized framework surpasses current approaches of hexagonal image processing systems, while hexagonal artificial neural networks can benefit from the implemented hexagonal architecture. As hexagonal lattice format based deep neural networks, also called H-DNN, can be compared to their square counterparts by transforming classical square lattice based data sets into their hexagonal representation, they can also result in a reduction of trainable parameters as well as result in increased training and test rates.	⤓
2210.09730v1	Efficient Machine-Learning-based decoder for Heavy Hexagonal QECC	10/18/2022 10:16:14 AM	Errors in heavy hexagonal code and other topological codes like surface code were usually decoded using the Minimum Weight Perfect Matching (MWPM) based decoders. Recent advances have shown that topological codes can be efficiently decoded by deploying machine learning (ML) techniques, for example, neural networks. In this work, we first propose an ML based decoder and show that this decoder can decode heavy hexagonal code efficiently, in terms of the values of threshold and pseudo-threshold, for various noise models. We show that the proposed ML based decoding method achieves ~ 5 times higher values of threshold than that by MWPM. Next, exploiting the property of subsystem codes, we define gauge equivalence in heavy hexagonal code, by which two different errors can belong to the same error class. We obtain a quadratic reduction in the number of error classes for both bit flip and phase flip errors, thus achieving a further improvement of $\sim 14\%$ in the threshold o ver the basic ML decoder. A novel technique of rank based gauge equivalence minimization to minimize the number of classes is further proposed, which is empirically faster than the previously mentioned gauge equivalence minimization.	⤓
2205.14664v1	Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies in Machine Learning and Databases	5/29/2022 1:43:17 PM	Today's computing systems require moving data back-and-forth between computing resources (e.g., CPUs, GPUs, accelerators) and off-chip main memory so that computation can take place on the data. Unfortunately, this data movement is a major bottleneck for system performance and energy consumption. One promising execution paradigm that alleviates the data movement bottleneck in modern and emerging applications is processing-in-memory (PIM), where the cost of data movement to/from main memory is reduced by placing computation capabilities close to memory. Naively employing PIM to accelerate data-intensive workloads can lead to sub-optimal performance due to the many design constraints PIM substrates impose. Therefore, many recent works co-design specialized PIM accelerators and algorithms to improve performance and reduce the energy consumption of (i) applications from various application domains; and (ii) various computing environments, including cloud systems, mobile systems, and edge devices. We showcase the benefits of co-designing algorithms and hardware in a way that efficiently takes advantage of the PIM paradigm for two modern data-intensive applications: (1) machine learning inference models for edge	⤓

When a user clicks the save button, the research article is saved to the user's profile page.

Profile page



Profile - Research Advisor

Not secure | ec2-44-240-38-221.us-west-2.compute.amazonaws.com/User/Profile

Research Advisor hyup3569@gmail.com Logout

Profile

Email

[redacted]@gmail.com

Password

[redacted]

LikedCategory

Computer Science 1

Save

Clear search history 2

ID	Title	Updated Date	Remove
1911.11251v7	Hexagonal Image Processing in the Context of Machine Learning: Conception of a Biologically Inspired Hexagonal Deep Learning Framework	3/18/2021 11:21:56 PM	⊗ 3
2210.09730v1	Efficient Machine-Learning-based decoder for Heavy Hexagonal QECC	10/18/2022 10:16:14 AM	⊗
2205.14664v1	Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies in Machine Learning and Databases	5/29/2022 1:43:17 PM	⊗
2105.04547v2	Highly Efficient Memory Failure Prediction using Mcelog-based Data Mining and Machine Learning	5/16/2021 5:38:51 AM	⊗
2003.04915v1	Managing Data Lineage of O&G Machine Learning Models: The Sweet Spot for Shale Use Case	3/10/2020 6:10:16 PM	⊗

Back to Home

© 2022 - Research Advisor

The Profiles page provides the following features: (1) a user can change the category he or she is interested in, (2) a user can initialize the search history. (3) a user can remove saved research articles.