Peter the Great Saint-Petersburg Polytechnic University

Institute of Computer Science and Technology

High School Of Programming Engineering

Hybrid image recommendation algorithm combining content and collaborative approaches

Kobyshev Kirill – ks.kobyshev@edu.spbstu.ru

Voinov Nikita – voinov@ics2.ecd.spbstu.ru

Relevance

Application fields of recommender systems:

- Search of suitable publications is scientific communities, social networks, photo hosting
- Search of target audience to organize thematic event
- HR industry (selection of vacancies and candidates)
- Sales field (selection of products in internet shops)
- Selection of materials (movies, music, books) in entertainment portals
- Advertising sphere

Recommender systems relate with scientific disciplines:

- Mathematical statistics
- Machine learning
- Discrete math
- Theory of optimization

Technologies used in recommender systems:

- Systems for storage and processing of big data volumes (Hadoop HDFS, Apache Spark)
- Implementations of machine learning algorithms (TensorFlow, Keras, Deeplearning4j)
- Graph database management systems (InfiniteGraph, Neo4j)

Conclusion: recommendation automation issue is **actual** and is **in demand**.

Goal and issues

Goal:
Decrease complexity and increase the quality of image recommendation for Internet users by automated software tool, implemented based on graph database containing images, topics and users.

Issues to achieve the goal:

1. Analyze existing solutions for image recommendation and define their shortcomings.
2. Propose solution to address the shortcomings of existing solutions.
3. Define image recommendation algorithm based on the proposed solution.
4. Develop prototype of recommendation algorithm and estimate accuracy, completeness and execution time.

Existing solutions for image recommendation

Images representation	Metadata (Textual features)	Visual Features		Hybrid Features
		Impl. 1	Impl. 2	
Algorithm parts	• Calculation of user features • Pearson correlation coefficient	• Calculation of image features		• Calculation of image features • GCN • random walks
		• SIFT • SURF • LBP • k-NN	• CNN • k-NN	
Filtration type	Collaborative Memory-based	Content		Collaborative Model-based
User ratings consideration	Yes	No		No
Doesn’t require manual actions	No	Yes		No
Computation resources	Intel Core i5-6500 CPU, 8GB RAM (insignificant)	Unknown	Intel i7-6850K CPU, NVIDIA 1080Ti GPU (significant)	16 Tesla K80 GPU (quite significant)

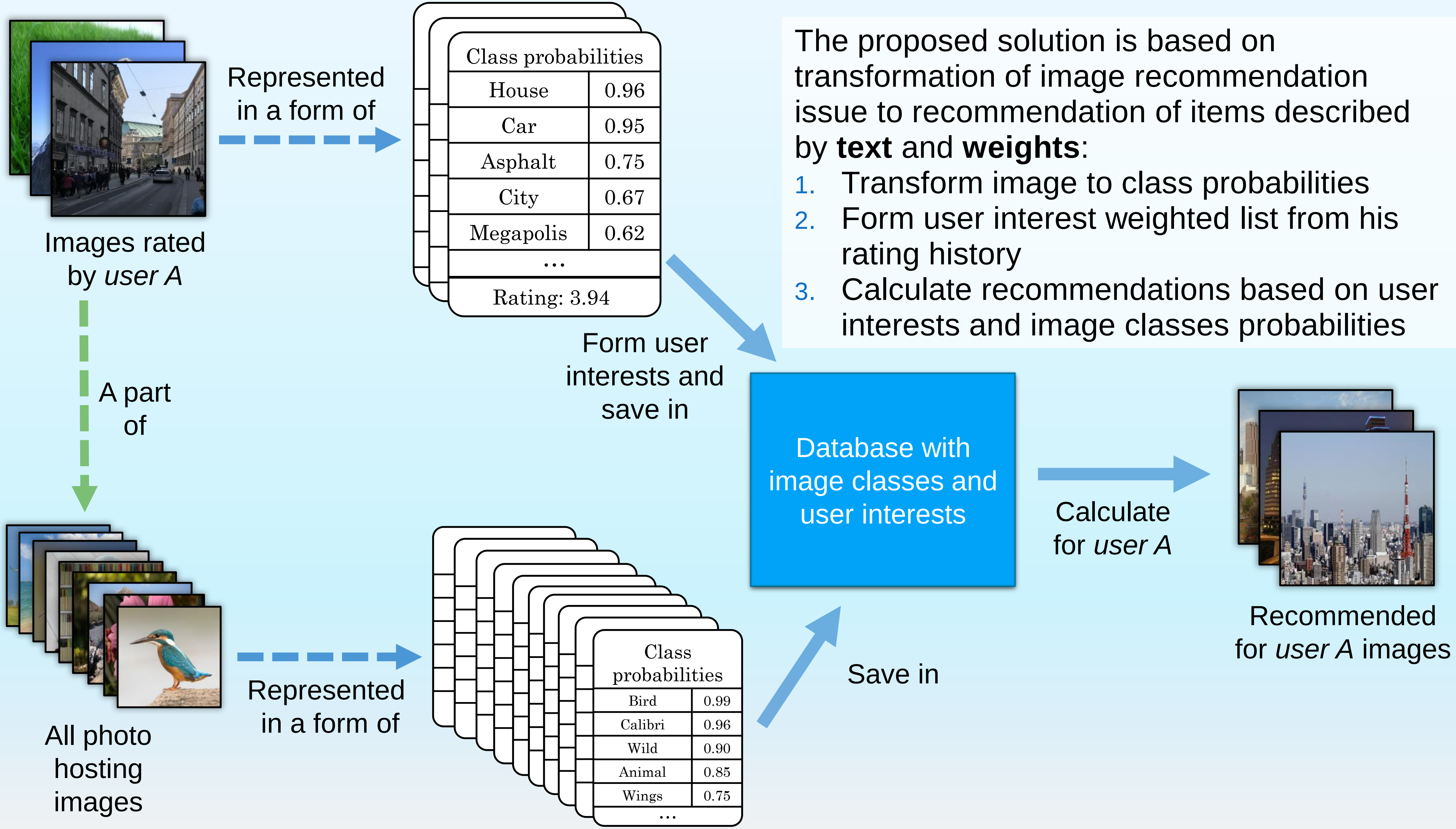
Existing solution shortcomings:

- Necessity of manual filling of metadata
- Lack of user rating history consideration
- Necessity of significant computation resources

Proposed Solution

The proposed solution is based on transformation of image recommendation issue to recommendation of items described by **text** and **weights**:

1. Transform image to class probabilities
2. Form user interest weighted list from his rating history
3. Calculate recommendations based on user interests and image classes probabilities



Images rated by *user A*

Represented in a form of

Class probabilities	
House	0.96
Car	0.95
Asphalt	0.75
City	0.67
Megapolis	0.62
…	
Rating: 3.94	

Form user interests and save in

A part of

All photo hosting images

Represented in a form of

Class probabilities	
Bird	0.99
Calibri	0.96
Wild	0.90
Animal	0.85
Wings	0.75
…	

Save in

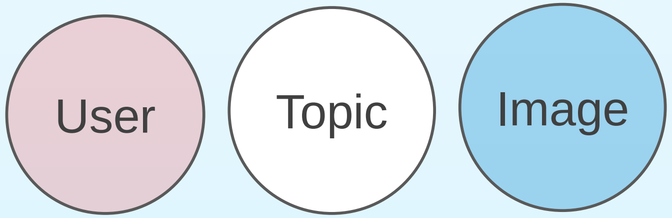
Database with image classes and user interests

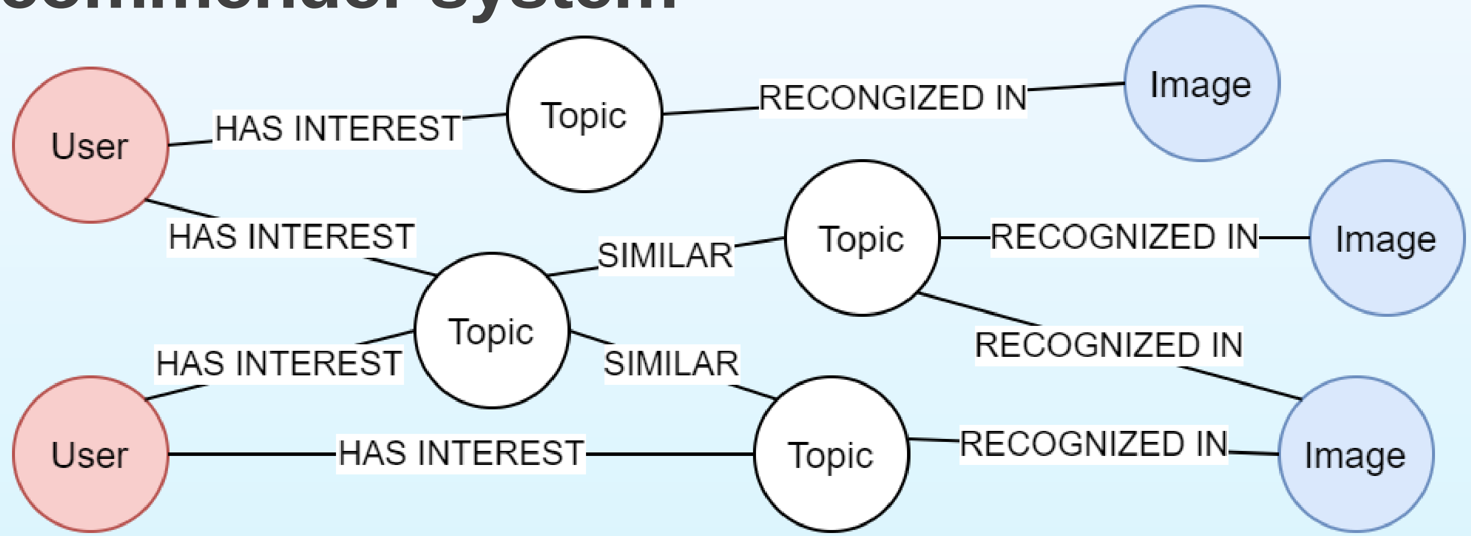
Calculate for *user A*

Recommended for *user A* images

Graph data model of recommender system

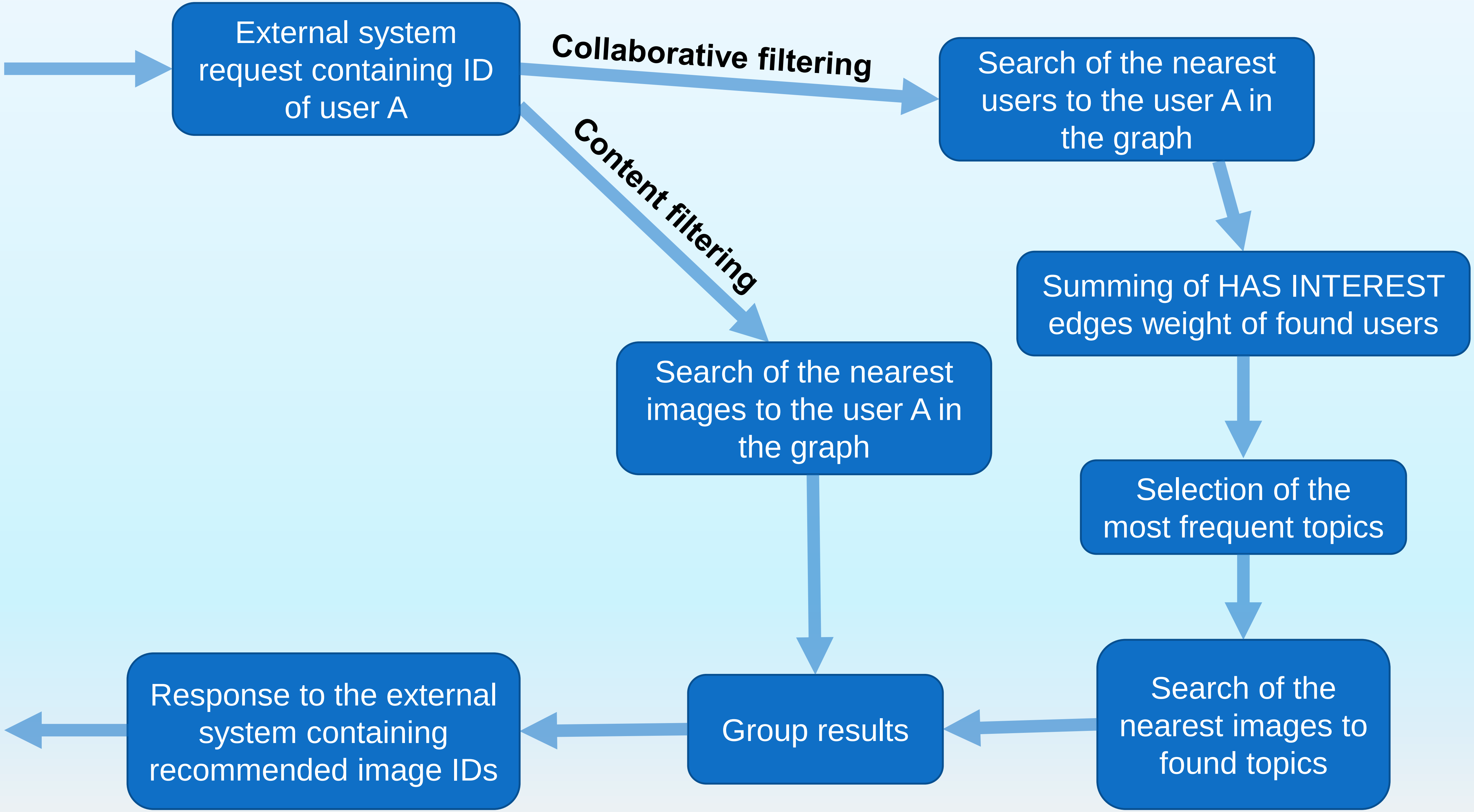
Node types





Edge type	HAS INTEREST	SIMILAR	RECOGNIZED IN
Between edges	User – Topic	Topic – Topic	Topic – Image
Edge weight is calculated from	User rating history	Euclidian distance between words in GloVe model	Class probability
Formula to calculate edge weight	weight(A,B) = $\frac{M_{AB}}{\sum_{j=1}^{k} M_{Aj}}$, A - user node, B - topic node, M_A- interest weight vector of user A, $M_{AB} = \sum_{i=1}^{n} P\big(B \in classes(I_i)\big) \times R_i$	weight(A,B) = $\sqrt{\sum_{i=1}^{n}(X_A^i - X_B^i)^2}$ A - one topic node, B - another topic node, X_A ,X_B - coordinates in semantic space	weight(A,B) = $1 - P\big(W_A \,\epsilon\, classes(I_B)\big)$, A – topic node, B – image node
Possible values	From 0 to 1	From 0 to ∞	From 0 to 1

Image recommendation algorithm


External system request containing ID of user A

Collaborative filtering

Search of the nearest users to the user A in the graph

Summing of HAS INTEREST edges weight of found users

Selection of the most frequent topics

Search of the nearest images to found topics

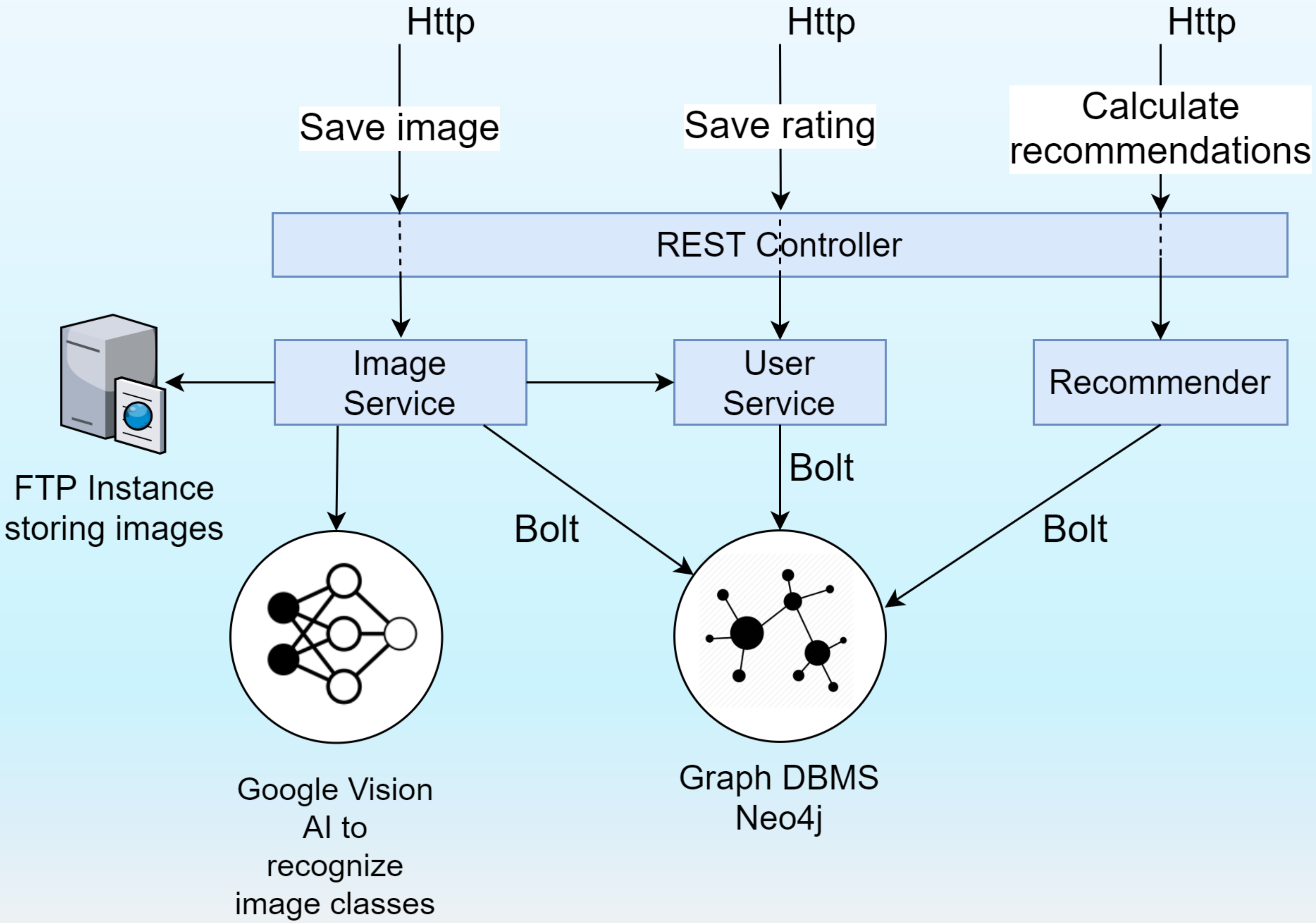
Content filtering

Search of the nearest images to the user A in the graph

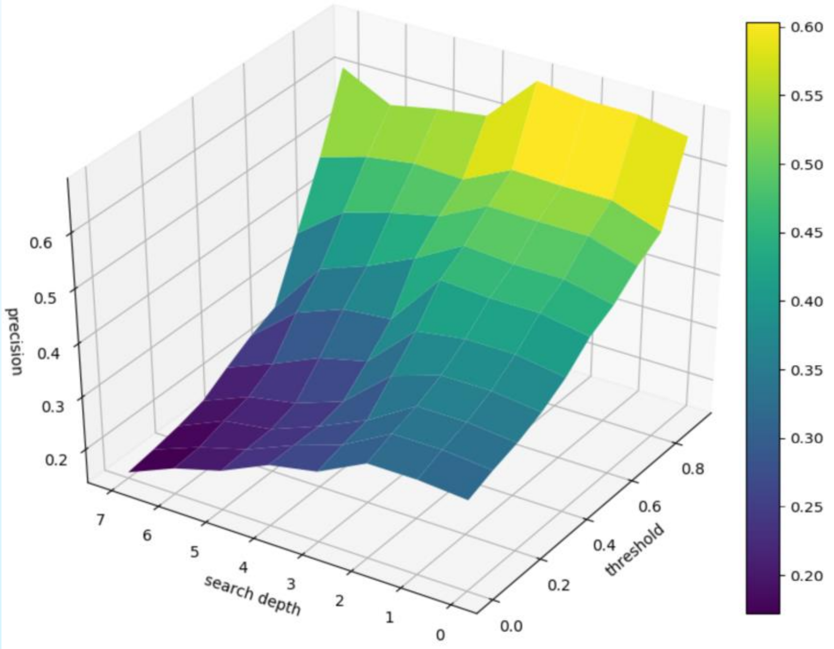
Group results

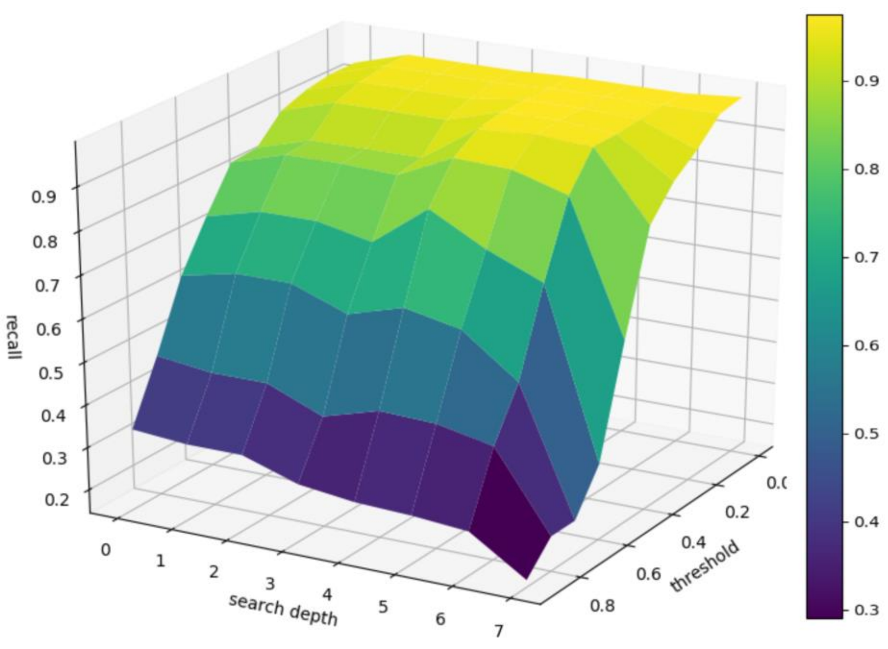
Response to the external system containing recommended image IDs

Recommender system architecture



Results





Parameters:

$search\ depth = 4$

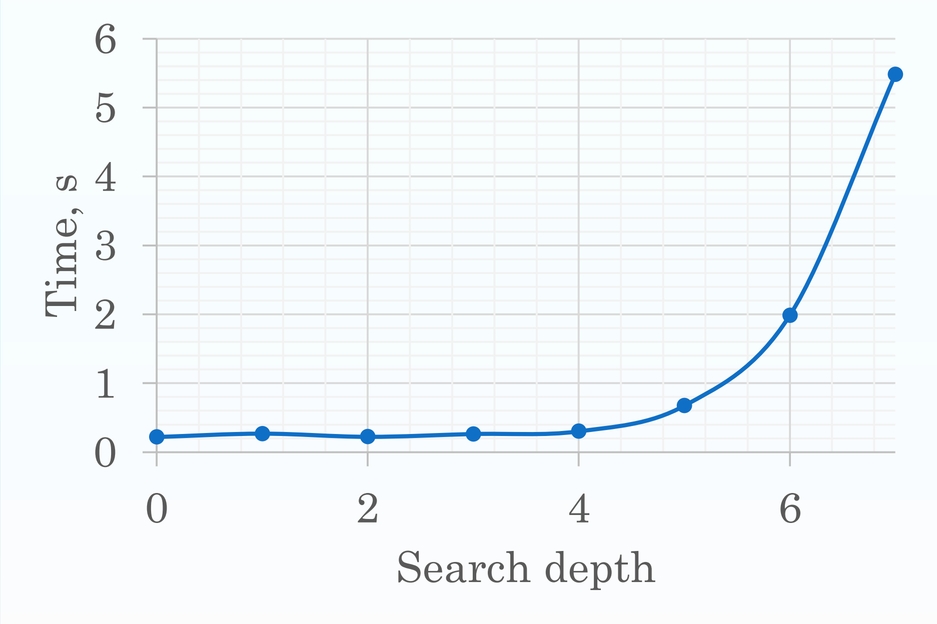
$threshold = 0.5$

Obtained metrics:

$Time = 303 \text{ ms}$

$Precision = 0.61$

$Recall = 0.72$



Test data:
Image amount – 1500
Topic amount – 50000
User amount – 3
Rating history size of each user – 100

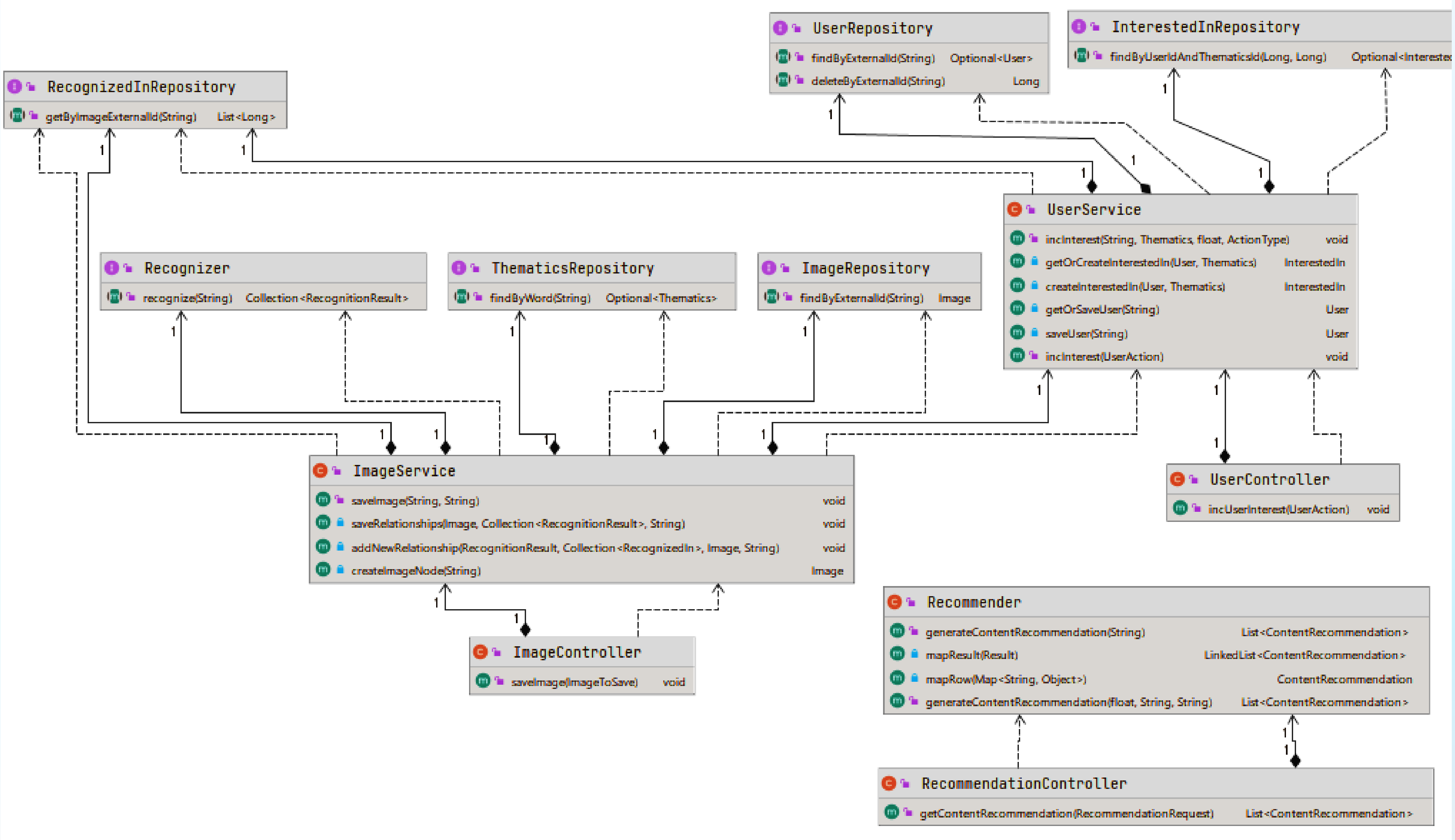
Characteristic	Node 1	Node 2
What was deployed	Java application	Neo4j DBMS
CPU	Intel Core i5-8250U	Intel Core i7-4702 MQ
RAM	8 Gb	8 Gb

Conclusion

- Existing solutions for image recommendation were analyzed and their shortcomings were defined.
- Defined algorithm to address the shortcomings of existing solutions.
- Recommendation algorithm was implemented in a form of recommender system prototype.
- The recommendation algorithm was tested for accuracy, completeness and execution time.

Thank you !

Class diagram



RecognizedInRepository
getByImageExternalId(String) List<Long>

UserRepository
findByExternalId(String) Optional<User>
deleteByExternalId(String) Long

InterestedInRepository
findByUserIdAndThematicsId(Long, Long) Optional<Intereste

Recognizer
recognize(String) Collection<RecognitionResult>

ThematicsRepository
findByWord(String) Optional<Thematics>

ImageRepository
findByExternalId(String) Image

UserService
incInterest(String, Thematics, float, ActionType) void
getOrCreateInterestedIn(User, Thematics) InterestedIn
createInterestedIn(User, Thematics) InterestedIn
getOrSaveUser(String) User
saveUser(String) User
incInterest(UserAction) void

ImageService
saveImage(String, String) void
saveRelationships(Image, Collection<RecognitionResult>, String) void
addNewRelationship(RecognitionResult, Collection<RecognizedIn>, Image, String) void
createImageNode(String) Image

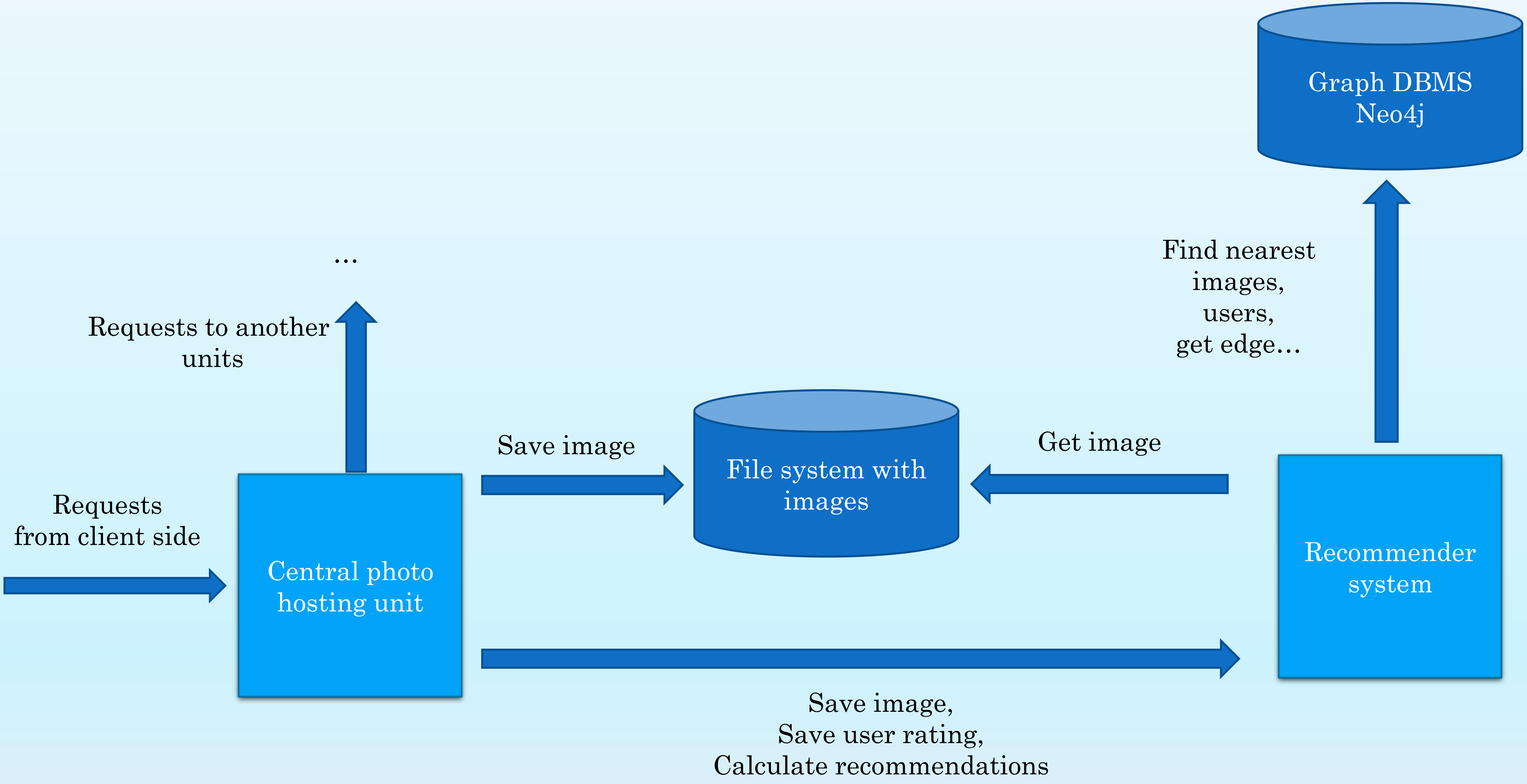
UserController
incUserInterest(UserAction) void

ImageController
saveImage(ImageToSave) void

Recommender
generateContentRecommendation(String) List<ContentRecommendation>
mapResult(Result) LinkedList<ContentRecommendation>
mapRow(Map<String, Object>) ContentRecommendation
generateContentRecommendation(float, String, String) List<ContentRecommendation>

RecommendationController
getContentRecommendation(RecommendationRequest) List<ContentRecommendation>

Recommender system integration with external system



Graph DBMS Neo4j

Find nearest images, users, get edge…

…

Requests to another units

Save image

File system with images

Get image

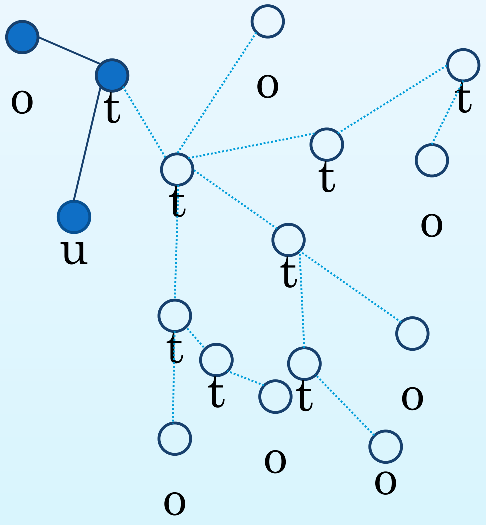
Requests from client side

Central photo hosting unit

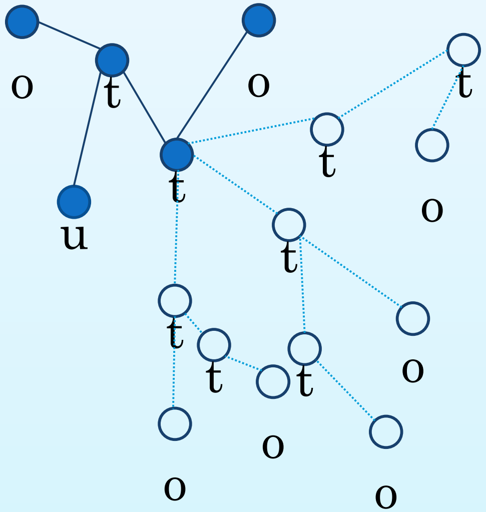
Recommender system

Save image,
Save user rating,
Calculate recommendations

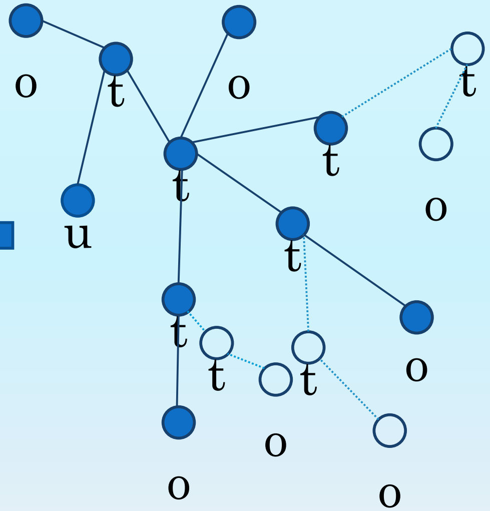
Request to find the nearest nodes (images/users)



Search depth = 0



Search depth = 1



Search depth = 2

Filter the nodes (images/users) having *weight(i)* < threshold * max(*weight*)

Return the nearest nodes (images/users)