Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 193 (2021) 200–209

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Young Scientist Conference on Computational Science

Hybrid image recommendation algorithm combining content and collaborative filtering approaches

Kirill Kobyshev[a], Nikita Voinov[a,]*, Igor Nikiforov[a]

[a]*Peter the Great St.Petersburg Polytechnic University, Polytechnicheskaya, 29, St.Petersburg, 195251, Russia*

Abstract

The paper relates to the subject field of image recommender systems. The proposed approach of automatic image recommendation addresses the following shortcomings of existing solutions: manual input of metadata by users, the lack of user rating history consideration, significant computational resources. The main idea of the proposed approach is to recognize object classes from images using a convolutional neural network to make recommendations. In the proposed solution users and images are located in the semantic space represented by a graph. Software implementation of the proposed approach and obtained results are considered.

Keywords: image recommender system; graph database; convolutional neural network; word2vec; collaborative filtering; content filtering

1. Introduction

Recommender systems have been developed since the mid-1990s as Internet-services to find the content interesting for users: articles (Arxiv.org), news (Surfingbird), people (LinkedIn), products (Amazon, Ozon, Aliexpress), images (500px, Pinterest, Instagram), videos and movies (YouTube, Netflix), music (Fast.fm, Pandora, Spotify). There are two main approaches used in software recommender systems for automatic recommendation: collaborative and content filtering [1,2]. Content filtering searches for a content which is similar to what the user liked before. Collaborative filtering searches for a content which is similar to what other users with same interests liked. The proposed solution is a hybrid algorithm for image recommendations combining content and collaborative

* Corresponding author. Tel.: +7-921-744-2898.
E-mail address: voinov@ics2.ecd.spbstu.ru

1877-0509
10.1016/j.procs.2021.10.020

filtering. Image recommendations are widely used in photo hosting services, social networks and other applications processing materials with images. The proposed algorithm allows users to avoid manual filling of metadata and to consider user rating history.

2. Issues of existing solutions for image recommendation

Existing automatic image recommendation approaches include analysis of images metadata [3,4]; extraction and analysis of visual features from images [5,6]; analysis of hybrid features [7] generated from metadata and visual features. Results of the comparative analysis of several existing implementations are presented in Table I.

Table 1. Considered existing software implementations of image recommender systems.

	Based on images metadata	Based on visual features	Based on hybrid features
Algorithm	Pearson correlation coefficient	1) SIFT, SURF, LBR, k-NN 2) CNN, k-NN	GCN, random walks
Filtering type	Collaborative, memory-based	Content	Collaborative, model-based
Considering history of user ratings	Yes	No	No
Manual actions are required	Yes	No	No
Computational resources	Intel Core i5-6500 CPU, 8GB RAM	Intel i7-6850K CPU, NVIDIA 1080Ti GPU	16 Tesla K80 GPU

The first image recommendation approach is based on the analysis of image metadata and one of its existing implementations is described in [3]. Recommender system creates a matrix of user interests from metadata of the images the user rated before. Authors also use such simple visual feature as image color density (red, green and blue) which was added to the matrix of user interests, but this feature does not contain significant information about images and does not improve image recommendation significantly. So, this implementation was classified as “use images metadata to make recommendations“ approach. The system predicts user ratings of images using Pearson correlation coefficient to calculate distances between users in feature space (one of the commonly used collaborative filtering algorithms). There are a lot of other possible solutions and algorithms not necessarily related to image recommendations because the source data describing images is represented in textual form. When image recommender system is implemented based on metadata analysis it is possible to reuse any solution working with textual data. But all of these solutions have same disadvantages one of which is necessity of manual filling of metadata (for example, a text description of the image). When the metadata is being analyzed the full automation is excluded. Furthermore, metadata can be described inaccurately and may contain insufficient or excessive amount of information. For example, average metadata error of the Instagram application is 80% [8].

The second approach is based on visual features in the feature space. Considered are two similar solutions based on this approach [5,6]. Firstly the recommender system groups images into clusters by k-NN algorithm in both implementations. The first implementation creates image features from the input image by SIFT, SURF and LBR algorithms, and the second implementation creates image features by a convolutional neural network. Then in both implementations the recommender system finds images in the cluster where the input image is located. The image itself is accepted as input data and the textual description of image is not required to be filled manually. But these implementations do not consider user rating history.

The third approach is based on hybrid features from images and image metadata. The approach implementation is described in [7]. Authors propose to train the convolutional neural network by image relationships graph in the Pinterest application (Graph Convolutional Neural Network, GCN). The relationship graph is generated from topics (“pin“) and collections of topics (“board“). Each image corresponds to one or more topics. The neural network is multilayered, the feature space dimension equals to the amount of layers of the neural network. Each layer outputs

one X_i^j coordinate of the image I_j after training of the network. The first layer uses the information about images and image metadata as input data. Each image has coordinates in the feature space and the recommender system finds appropriate images with similar content using the data prepared during the training on the graph. Users create the graph, thus the filtering type of this implementation is collaborative. The solution has several disadvantages due to large initial training sample (18 TB), significant computational resources, complex relationship of graphs between images. This solution does not consider user rating history and recommends only one chosen image.

In this research the algorithm based on the second approach is proposed. It includes several improvements in compare to existing solutions:

1. No manual metadata configuration is required. This improvement allows users to obtain fully automated recommendation algorithm and to avoid errors caused by human factor.
2. Consideration of user rating history. This shall make recommendations more complete, however it is not possible to compare the completeness of recommendations (recall metric) of the proposed solution and other solutions which do not consider user rating history. Metrics in [5-7] are based on test data marked by quite another method: test data there includes images marked as recommended if they are similar to the passed image. While in the proposed solution the image is marked as recommended in case it corresponds to user interests based on his ratings. Completeness cannot be compared to the solution in [3] as well because results of the recall metric are not provided there.

Non-functional requirements specified for the proposed algorithm are the following:

1. Expected precision and recall values shall exceed 0.6. Recommended images in test data shall take about 5% of all images.
2. Expected calculation time shall not exceed 300 ms.
3. Reasonable cost of computational resources. Configuration of a typical PC shall be enough to calculate recommendations in expected time limits.

3. Proposed solution

Two main features of the proposed solution are the following:

1. Object class recognition which allows performing text analysis instead of visual features analysis.
2. Storing users, images and topics in the graph structure. Edges between topics and images allow considering object class probabilities. Edges between users and topics allow considering user interest weight. And edges between topics allow users to find the closest (and the most relevant) one to user images.

The scheme of the proposed solution is presented in Fig.1a, considering a set of images in a photo hosting. For each image the recommender system defines a list of classes while some of the images are rated by the user. A list of user interests created from the set of rated images and a list of image classes are saved in the database. The recommender system generates recommendations based on user interests and information about object classes in images.

3.1. Object class recognition

The proposed solution is based on the second approach mentioned above. This approach was improved by using fully-connected layers and output layer where softmax function is used to get probabilities of object classes.

Object classes recognized on images that interested the user create user interests. Thus the image recommendation task is transformed into the task of image recommendation based on the textual information about images with weights.

Each user action is represented as a pair of $I_i$and R_i where I_i is an image and R_i is a user rating of this image. For the user U the weight M_{UT} of the interest T is calculated by (1). In (1) $P(T \in classes(I_i))$ is a probability of the object class T in the image I_i and if the object class T wasn't recognized in the image I_i then $P(T \in classes(I_i)) = 0$.

$$M_{UT} = \sum_{i=1}^{n} P(T \in classes(I_i)) \times R_i \quad (1)$$

3.2. Graph with users, images and topics

In the proposed solution all object class IDs correspond to the words of natural language. Representation of object classes by words allow users to find semantically close words by using the Word2Vec model [9,10] or its modifications (for example, GloVe [11]). The Word2Vec model is used in many recommender systems [12-14]. The application of Word2Vec in the current solution is based on the solution [13] where user coordinates midpoints in the semantic space between the items other users liked. The content filtering in this solution is performed by searching for the nearest user items in the semantic space, while collaborative filtering is performed by searching for the users nearest in semantic space to the current user and recommendation of items liked by them. This solution has two disadvantages:

1. If a user liked the items which are semantically far from each other then the midpoint between these items will not describe the user interests properly.
2. If this solution is applied in image recommendations then interest weights and object class probabilities will not be considered.

To overcome the issues above the semantic space with images and users was converted into the graph G. The graph has a base constant part $G_T \in G$ containing topics (words in natural language words). Recommender system adds images and users to the graph $G = (E, V)$. A simple example of the graph is presented in Fig.1b. The graph G contains a set of nodes V containing a subset of images $I \in V$, a subset of topics $T \in V$ and a subset of users $U \in V$. The graph G contains a set of edges E containing a subset of edges between topics $E_{TT} \in E$, between topics and images $E_{TI} \in E$ and between users and topics $E_{TU} \in E$.

The algorithm described in [15] was used to convert the semantic space containing words in natural language to the base graph G_T. This algorithms converts GloVe model [11] into a graph. The nearest words are grouped into clusters, all cluster words are linked by edges to the central word. Central words are grouped into clusters as well and the clustering algorithm is recursive. The resulting graph is a set of nodes of topics connected to each other. An example with a two-layer cluster is shown in Fig.1c.

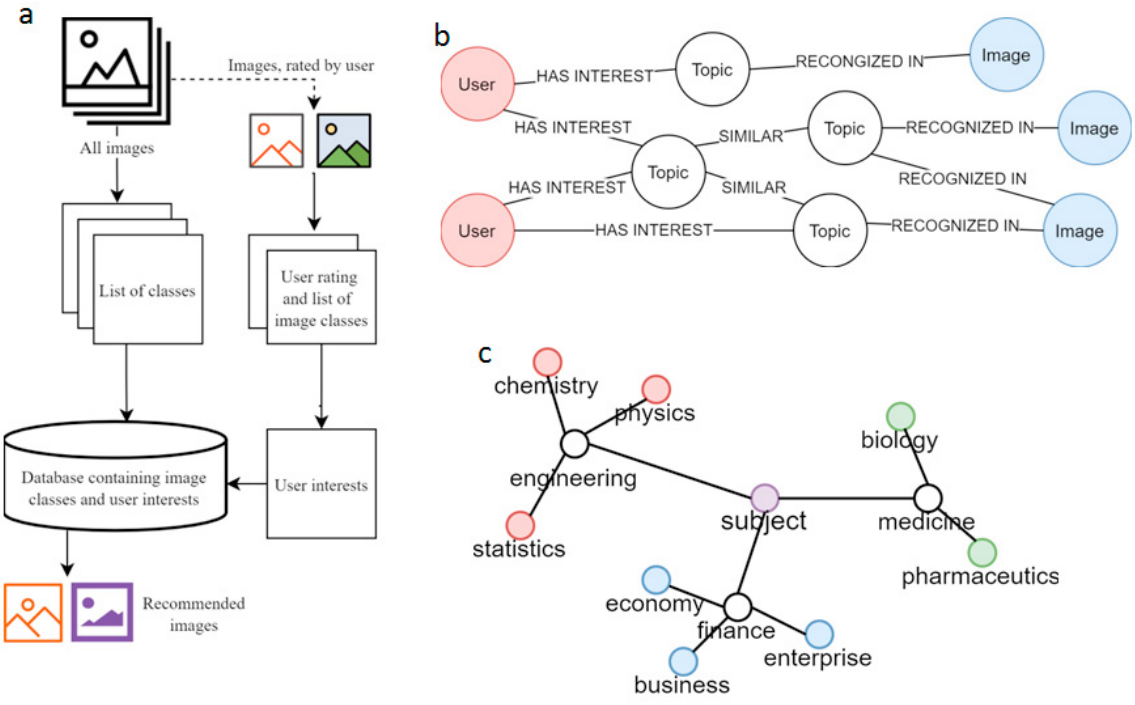

Fig. 1. (a) Overall scheme of the proposed solution; (b) a graph example; (c) example of a two-layer cluster.

Each topic node T_S contains a vector with coordinates of its words W_S located in the semantic space $X \in R^n$ of the dimension n. The weight of edge between the topics T_A and T_B equals to the Euclidean distance (2), where X_A are the coordinates of the topic T_A in the semantic space and X_B are the coordinates of T_B.

$$weight(E_{T_A T_B}) = \sqrt{\sum_{i=1}^{n} (X_A^i - X_B^i)^2} \tag{2}$$

Edges connect images to topics which are the object classes for images. The edges weight is calculated by (3), where T_A is the topic A, I_B is the image B, W_A is the word in natural language related to the topic T_A, $classes(I_B)$ is a set of object classes recognized in the image I_B, $P(W_A \in classes(I_B))$ is a probability of the class W_A in the image I_B.

$$weight(E_{T_A I_B}) = 1 - P(W_A \in classes(I_B)) \tag{3}$$

Users are connected to the graph G_T by edges according to the list of their interests with weights. A topic node to which a user was connected is one of his interests. Each edge $E_{T_i U_j}$ corresponds to the interest weight M_{ij}. The weight of the edge between the topic A and the user B is calculated by (4), where M_{AB} is the interest weight of the user B in relation to the topic A, M_{iB} is the interest weight of the user B in relation to the topic i, k is the amount of user interests. In other words, the weight of the edge is a relation of the interest weight to the sum of all user interest weights.

$$weight(E_{T_A U_B}) = \frac{M_{AB}}{\sum_{i=1}^{k} M_{iB}} \tag{4}$$

4. Implementation of the image recommender system

The architecture of the implemented image recommender system is presented in Fig.2. The system interacts with the external system (for example, photo hosting service) by HTTP requests. “REST Controller“ receives requests from the external system and transfers parameters to other services. There are the following types of requests from the external system: “Save Image“, “Update User Interests“, “Calculate Recommendations“.

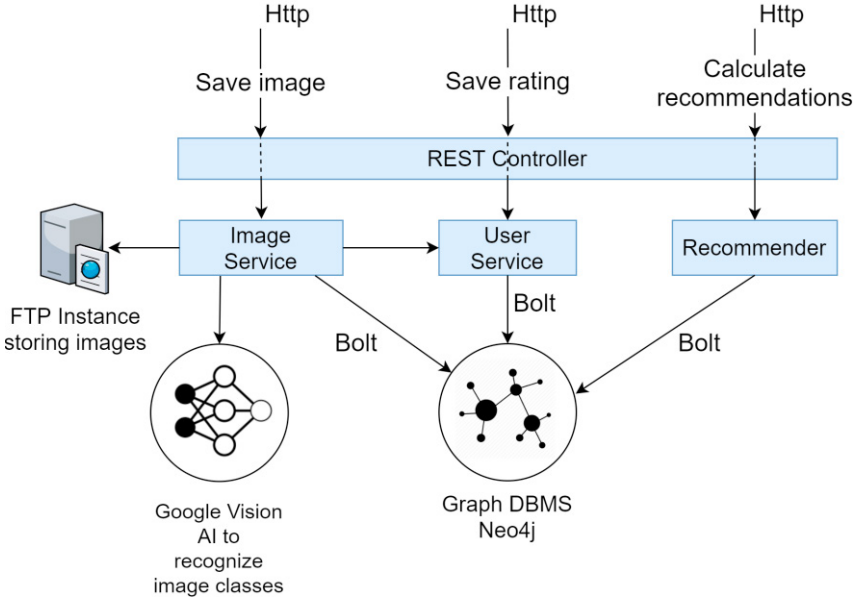

Fig. 2. The architecture of the implemented image recommender system.

The component “ImageService“ performs the following:

- Saving image node to the graph database.
- Object class recognition from images.
- Saving object classes to graph database as edges between images and topics.

The component “UserService“ updates edges between users and topics by received information about user ratings.

The component “Recommender“ creates recommendations from content and collaborative filtering results.

4.1. Processing of the “Save Image“ request

The algorithm processing the “Save Image“ request is presented in Fig.3a. The recommender system creates an image node in the graph database (line 1). Then it loads the image from FTP server and sends the image to the input of the convolutional neural network, which outputs the list of object classes with probabilities (lines 2–3). Then the recommender system saves edges between image and topics that are object classes of the image (lines 4–9).

4.2. Processing of the “Update User Interests“ request

The algorithm processing the “Update User Interests“ request is presented in Fig.3b. The recommender system loads topics related to the image from the graph database and weights from these topics to the image (line 1). Then for each topic it increases edges from user weight to the value of the user rating (lines 2–10).

a

Algorithm 1 Image node saving algorithm

Input: $userId, imageId, G = (V, E)$

1: create image node with $imageId$
2: $imageBitmap \leftarrow$ load image from FTP by $imageId$
3: $classes \leftarrow$ recognize from $imageBitmap$
4: **for** $class$ in $classes$ **do**
5: $topicId \leftarrow class.name$
6: $P \leftarrow class.probability$
7: create edge $E_{class} \in E$ between image node with $imageId$ and topic node with $topicId$
8: $E_{class}.weight = 1 - P$
9: **end for**

b

Algorithm 2 “Update User Interests“ algorithm

Input: $userId, imageId, rating, G = (V, E)$

1: $topics \leftarrow$ next to image with $imageId$ nodes $\in G$
2: **for** $topic$ in $topics$ **do**
3: $topics_{next} \leftarrow$ next to $user$ nodes $\in G$
4: $edges_{next} \leftarrow$ edges $\in E$ between $user$ and $topics_{next}$
5: $edge_{current} \leftarrow$ edge $\in E$ between $user$ and $topic$
6: **if** $edge_{current} = \emptyset$ **then**
7: create edge $\in E$ between $user$ and $topic$
8: **end if**
9: increase $edge_{current}.interest$ to $rating$
10: **end for**

c

Algorithm 3 Image recommendation algorithm

Input: $userId, K, M, N, G = (V, E)$

Output: $recommended$

1: $contentRes \leftarrow$ the nearest to user with $userId$ image nodes $\in G$
2: $anotherUsers \leftarrow$ the nearest to user with $userId$ user nodes $\in G$
3: $currTopics \leftarrow$ next to user with $userId$ nodes $\in G$
4: $interests \leftarrow \emptyset$
5: **for** $anotherUser$ in $anotherUsers$ **do**
6: $topics \leftarrow$ next to $anotherUser$ nodes $\in G$
7: **for** $topic$ in $topics$ **do**
8: **if** $topic \notin currTopics$ **then**
9: $weight \leftarrow$ weight of edge $\in E$ between $anotherUser$ and $topic$
10: **if** $\exists interests_i : interests_i.topic = topic$ **then**
11: increase $interests_i.weight$ to $weight$
12: **else**
13: $interest \leftarrow$ object with $topic$ and $weight$
14: $interests \leftarrow interests \cup interest$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: $interest \leftarrow \forall interest_i : \frac{interest_i.weight}{\sum_{j=1}^{n} interest_j.weight} > \alpha$
20: $interestTopics \leftarrow$ map $interests$ array to topics array
21: $collabRes \leftarrow$ N next to $interestTopics$ image nodes $\in G$
22: **return** $contentRes \cup collabRes$

Fig. 3. (a) Processing the “Save Image“ request; (b) processing the “Update User Interests“ request; (c) processing the “Calculate Recommendations“ request.

4.3. Processing of the "Calculate Recommendations" request

The algorithm processing the "Calculate Recommendations" request is presented in Fig.3c. The recommender system loads the parameters K, M, N from the configuration. The system searches for K images which are the nearest to the user images in the graph (line 1). This is the result of content filtering. Then the recommender system searches for M other users which are the nearest to the current user (line 2) and for topics interesting for the current user (line 3). The system creates a list of interests of these M nearest users (lines 5–18). Thus a new list of potential interests for the current user is created. Then the system filters interests by their weights and only interests with a share of more than some $\propto$ are left. Further it searches for N images of selected topics in the graph (line 21) which is the result of collaborative filtering. The system returns results of content and collaborative filtering to the external system (line 22).

4.4. Searching for the nearest nodes

Consider the lines 1-2 of the Algorithm 3 in more details. The recommender system searches for the nearest image nodes in the line 1 and for user nodes which are the nearest to the current user node in the line 2. The searching algorithm is the same in both cases and schematically described in Fig.4.

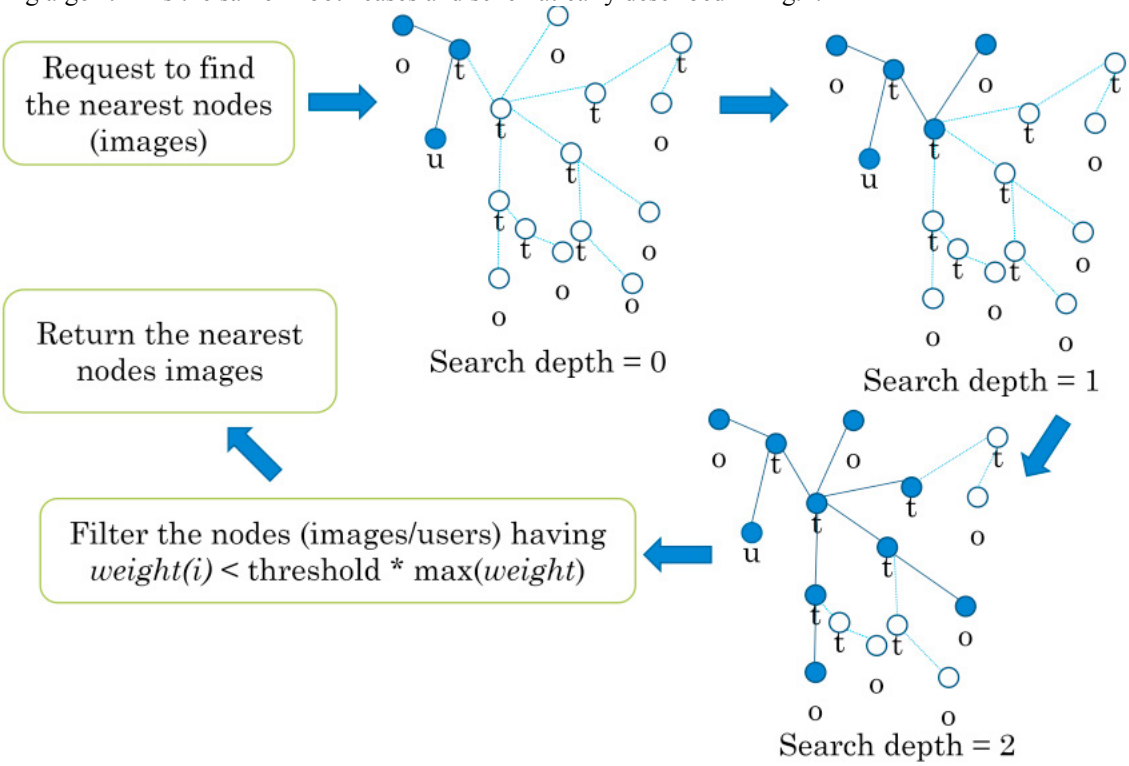

Fig. 4. Searching for the nearest nodes.

The parameters *threshold* and *search depth* are specified in the configuration of the recommender system. The aim is to find the nodes within the limits specified by *search depth*. In the example shown in Fig.5 *search depth* equals 2. Firstly the recommender system marks the node of the current user (marked by "u"). Then it marks next nodes (topics marked by "t") and neighbor images or users (marked by "o") to these nodes. Initial value of *search depth* is 0. Then the recommender system repeats this procedure and starts to move from the marked nodes. In each iteration the current *search depth* value is incremented until it reaches its specified limit.

Further the recommender system filters the nearest found nodes β_i (images for content filtering or users for collaborative filtering). For each β_i the value $weight^i_{User}$ is calculated by (5):

$$weight^i_{User} = weight^i_{User \to Topic} + weight^i_{Topic \to Topic} + weight^i_{Topic \to \beta} \quad (5)$$

where $weight^i_{Topic \to Topic} = \gamma \times \sum_{j=1}^{k} weight^i_{Topic_j \to Topic_{j+1}}$

In (5) $weight^i_{User \to Topic}$ is the weight of the edge from the user to the neighbor topic; $weight^i_{Topic_j \to Topic_{j+1}}$ is the weight of the edge from one topic into another topic including a path from the current user to the found node β_i; $weight^i_{Topic \to \beta}$ is the weight of the edge from the topic to the node β_i; γ is the coefficient which signifies the importance of $weight^i_{Topic_j \to Topic_{j+1}}$ terms (weights of edges between topics) in the overall sum. This sum considers the following factors:

- user interests ($weight^i_{User \to Topic}$)
- topics similarity in the path ($weight^i_{Topic \to Topic}$)
- $weight^i_{Topic \to \beta}$, which is a probability of the class (topic) in the image in case of content filtering or the weight of the interest in case of collaborative filtering.

Finally the recommender system calculates the maximal $weight^i_{User}$ and returns the nodes with the weight less than $threshold \times \max(weight^i_{User})$.

4.5. Used technologies

A prototype of the image recommender system is written in Java and based on the Spring Boot platform. Medium GloVe model [11] containing 50000 words of natural language and 100 coordinates in semantic space for each word is converted into the graph of topics G_T. The graph is stored in Neo4j database. Google Vision AI service is used to recognize image classes. Further it is planned to use Inception CNN model [16] locally to exclude network load caused by sending requests to Google Vision AI service.

5. Testing and results analysis

To test the implemented image recommender system a dataset containing 3 users, 1500 images and lists of user positive ratings was prepared. A set of images were collected from Google Images service using Selenium Webdriver framework. 100 topics were specified in .csv file and 15 images were downloaded for each topic. For each user 50 images positively rated by them and one abstract topic were defined (the topic “wild“ for the first user, “art“ - for the second user, “cities“ - for the third user). Recommended images containing objects interesting for users were marked manually.

Values of parameters were selected to achieve as more accurate and effective recommendations as possible considering manual preparation of the dataset. The parameter γ was defined to have most values of $weight^i_{Topic \to Topic}$ between 0 and 1, its value was selected as 0.01. Values for *threshold* and *search depth* were selected by complete enumeration to reach the maximum values of precision and recall metrics. The source metrics are the following:

- TP - correctly recommended images (true positives).
- FN - not recommended images, but shall be recommended (false negatives).
- FP - recommended images, but shall not be recommended (false positives).
- TN - correctly not recommended images (true negatives).

The precision metric was calculated by (6) and its chart is presented in Fig.5a. It grows with the growth of *threshold* value and decreases when *search depth* is more than 3.

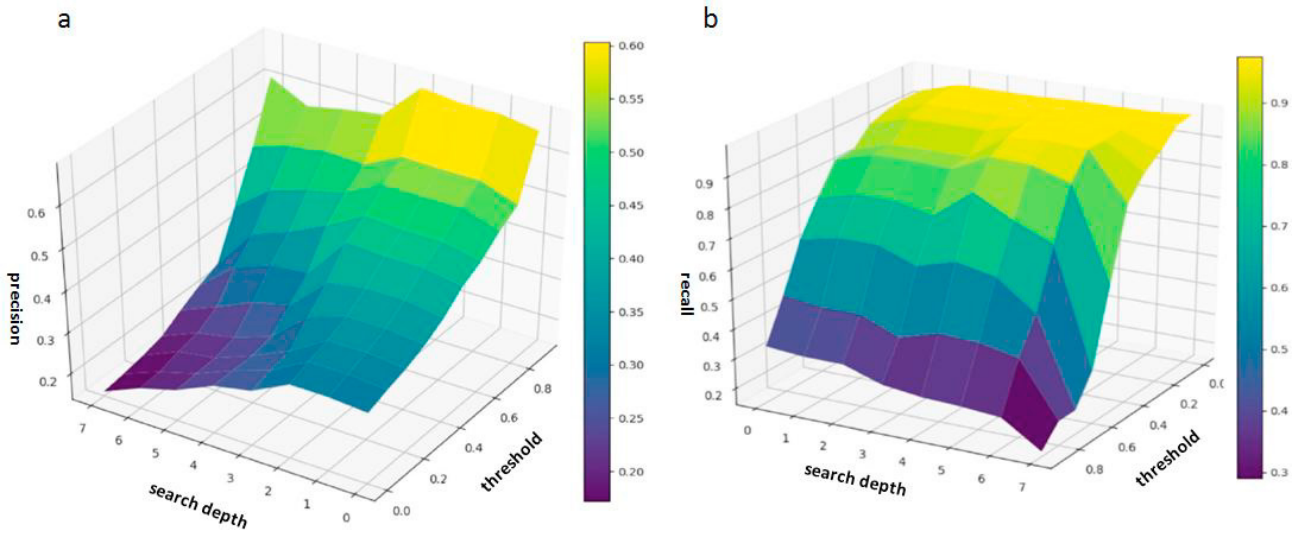

Fig. 5. (a) Precision metric chart; (b) Recall metric chart.

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

The recall metric was calculated by (7) and its chart is presented in Fig.5b. It grows with the decrease of *threshold* and *search depth* values.

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

The optimal point with expected precision and recall values is reached when *threshold* equals 0.8 and *search depth* equals 4. In this case precision equals 0.61 and recall equals 0.72.

The recommender system was deployed on two computational nodes connected to the same local network. The node for application part was configured with Intel Core i5-8250U and 8Gb RAM while the node for Neo4j database was configured with Intel Core i7-4702MQ and 8Gb RAM.

Time measurements are presented in Fig.6. A relatively fast growth of time is observed when *search depth* reaches 5. A delay of 300 ms between *search depth* values from 0 to 4 is caused by network load while connecting to the graph database. When *search depth* equals 2 the load is minimal.

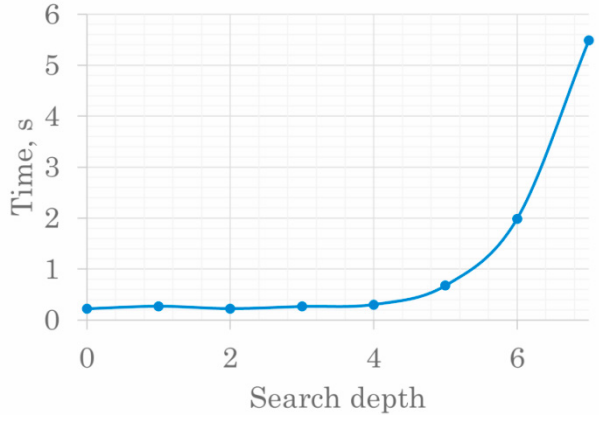

Fig. 6. Execution time chart.

6. Conclusion

In this paper existing solutions for automation of image recommendations are considered. To eliminate shortcomings of existing solutions a new approach is proposed to avoid manual input of metadata by users and the lack of user rating history consideration. The idea is to create a semantic space represented by a graph structure which stores users, topics and images. Object classes are recognized from images using a convolutional neural network. Testing of software implementation of the developed approach demonstrated appropriate values of precision, recall and execution time. Further it is planned to provide more experiments with parameters adjustment to improve the accuracy.

References

[1] Folasade Olubusola Isinkaye, Yetunde O. Folajimi, and Bolanle Adefowoke Ojokoh. (2015) "Recommendation systems: Principles, methods and evaluation." *Egyptian Informatics Journal* **16 (3)**: 261-273.

[2] Houtao Deng. (2019) "Recommender Systems in Practice.": https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a

[3] Fuhu Deng, Panlong Ren, Zhen Qin, Gu Huang, and Zhiguang Qin. (2018) "Leveraging Image Visual Features in Content-Based Recommender System." *Scientific Programming* **2018.**

[4] Andrey G. Gomzin, and Anton V. Korshunov. (2012) "Recommender Systems: modern methods review." *Proceedings of the Institute for System Programming of the RAS* **22**: 401-417.

[5] Zuhal Kurt, and Kemal Ozkan. (2017) "Image-based recommender system based on feature extraction techniques." *2nd International Conference on Computer Science and Engineering*: 769-774.

[6] Hessel Tuinhof, Clemens Pirker, and Markus Haltmeier. (2019) "Image Based Fashion Product Recommendation with Deep Learning.", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **11331**: 472-481 .

[7] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. (2018) "Graph Convolutional Neural Networks for Web-Scale Recommender Systems." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*: 974-983.

[8] Stamatios Giannoulakis, Nicolas Tsapatsoulis, and Klimis Ntalianis. (2018) "Identifying Image Tags from Instagram Hashtags Using the HITS Algorithm." *DASC-PICom-DataCom-CyberSciTec 2017* **2018**: 89-94.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. (2013) "Distributed Representations of Words and Phrases and their Compositionality.": https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[10] Igor Nikiforov, Nikita Voinov, Pavel Drobintsev. (2018) "A System Prototype for Real Time Automatic Fraud Detection in Text Data." *Proc. of XXI International Conference of Soft Computing and Measurement*: 724-727.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. (2014) "GloVe: Global Vectors for Word Representation." *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*: 1532-1543.

[12] Oren Barkan, and Noam Koenigstein. (2016) "Item2Vec: Neural Item Embedding for Collaborative Filtering." *IEEE International Workshop on Machine Learning for Signal Processing*.

[13] Makbule Gulcin Ozsoy. (2016) "From Word Embeddings to Item Recommendation.": https://arxiv.org/pdf/1601.01356.pdf

[14] Ramzi Karam. (2017) "Using Word2vec for Music Recommendations.": https://towardsdatascience.com/using-word2vec-for-music-recommendations-bb9649ac2484

[15] Thomas A. Trost, and Dietrich Klakow. (2020) "Parameter Free Hierarchical Graph-Based Clustering for Analyzing Continuous Word Embeddings." *Proceedings of TextGraphs@ACL 2017: The 11th Workshop on Graph-Based Methods for Natural Language Processing*: 30-38.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. (2015) "Going Deeper with Convolutions." *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **07-12-June-2015**: 1-9.