# Hybrid image recommendation algorithm combining content and collaborative filtering approaches

Kirill Kobyshev
*High School of Software Engineering*
*Peter the Great St. Petersburg Polytechnic University*
St. Petersburg, Russia
kobyshev.ks@edu.spbstu.ru

Nikita Voinov
*High School of Software Engineering*
*Peter the Great St. Petersburg Polytechnic University*
St. Petersburg, Russia
voinov@ics2.ecd.spbstu.ru

***Abstract*—Recommender systems are software tools for search of suitable for users content. This research relates to the subject field of image recommendations. Image recommender systems are used in photo hosting services, social networks and other applications, that have materials containing images. The proposed approach of automatic image recommendations addresses the following shortcomings of existing solutions: the necessity of the manual filling of metadata by users, lack of user rating history consideration, necessity of significant computation 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 as a graph. The proposed approach was implemented in the form of a correctly working prototype of the recommender system. Currently it is planned to finalize the prototype and deploy it on the webserver.**

***Index Terms*—recommender system, graph database, graphs, image recognition, cnn, inception, word embedding, word2vec, glove, hybrid filtering, collaborative filtering, content filtering, image recommendation**

## I. Introduction

Recommender systems are software tools that have been developed since the mid-1990 with services for Internet users and used to find suitable for users content: 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). The development of recommender systems is based on methodologies and algorithms from the following scientific and applied disciplines: discrete mathematics (social graphs, interest graphs), mathematical statistics (statistic criteria), machine learning (algorithms of semantic analysis, clusterization, neural networks), data science. Recommender systems might be developed using the following instruments and technologies: big data processing systems (Hadoop HDFS, Apache Spark, Apache HBase), graph database (InfiniteGraph, Neo4j), machine learning algorithms implementation (TensorFlow, DeepLearning4j, Keras).

There are two main approaches for automatic recommendation: collaborative and content filtering [1]. Content filtering is a search of content, similar to the interested user previously content. Collaborative filtering is a search of content similar to the content other users interested in which are similar to the current user.

The proposed solution is hybrid (combines content and collaborative filtering) and relates to image recommendations. Image recommendations are used in photo hosting services, social networks and other applications, that have materials containing images. The proposed algorithm allows us to avoid the necessity of manual filling of metadata by users and to consider user rating history. Source code of the solution is posted in the public repository: https://github.com/kruchon/ImageBasedRecommender.

## II. Problems of Existing Image Recommendation Solutions

There are the following automatic image recommendation approaches: analysis of images metadata [3]; 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 existing implementations is presented in Table I.

The first image recommendation approach is an analysis of image metadata, and the solution based on the approach is described in [3]. Recommender system forms a matrix of user interests from metadata of the images user rated before. Authors described the such simple visual feature as image color density also (red, green and blue), which was added to user interest matrix, but this feature does not contain significant information about images and doesn't improve image recommendation significantly. So, this implementation was classified as "use images metadata to make recommendations" approach. The system predicts user ratings relative to images with using of Pearson correlation coefficient to calculate distances between users in feature space (one of the commonly used collaborative filtering algorithms). And there are a lot of other possible solutions and algorithms, even not related to image recommendations, because the source data describing images is represented in textual form. When we implement image recommender system using an approach based on metadata analysis, we able to reuse any solution working with textual data. But all of these solutions will have same disadvantages. Necessity of manual filling of metadata (for example, a text description

TABLE I
EXISTING IMAGE RECOMMENDAION APPROACHES AND SOLUTIONS BASED ON THE APPROACHES

| Description \ Approach | Use images metadata to make recommendations | Use visual features to make recommendations | Use hybrid features to make recommendations |
|---|---|---|---|
| Author(s) | Fuhu Deng, Panlong Ren, Zhen Qin, Gu Huang, Zhiguang Qin | Implementation 1: Kemal Ozkan, Zuhal Kurt, Erol Seke; Implementation 2: Hessel Tuinhof,Clemens Pirker, Markus Haltmeier | Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai |
| Algorithm | Pearson correlation coefficient | Implementation 1: SIFT, SURF, LBR, k-NN Implementation 2: CNN, k-NN | GCN, random walks |
| Filtering type | Collaborative, memory-based | Content | Collaborative, model-based |
| Considers user ratings history | Yes | No | No |
| Doesn't require manual actions | No | Yes | Yes |
| Computation resources | Intel Core i5-6500 CPU, 8GB RAM (insignificant*) | Implementation 1: unknown; Implementation 2: Intel i7-6850K CPU, NVIDIA 1080Ti GPU (more significant, than images metadata approach implementation) | 16 Tesla K80 GPU (very significant) |

of the image) is the one of disadvantages. When we analyse the metadata we exclude the full automation. Furthermore, metadata can be described inaccuratelly and can contains insufficient or excessive amount of information. For example, average metadata error of the Instagram application is 80 percent [8].

The second approach relates to the search of images by visual features in the feature space. There are two similar solutions based on the approach [5], [6]. Before forming of recommendations the recommender system groups images to clusters by k-NN algorithm in both implementations. The first implementation forms image features from the input image by SIFT, SURF and LBR algorithms, and the second implementation forms image features by a convolutional neural network. Then the recommender system finds images in the cluster, where the input image is located (in both implementations). These implementations accept just an image as an input and don't require to manually fill the textual description of image. But these implementations don't consider user rating history.

The third approach is based on the forming of 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 the one of the topics and more. The neural network is multilayer, the feature space dimension is equal to amount of neural network layers. Each layer outputs the one of coordinates $X_i^j$ of the one of images $I_j$ after network training completion. The first layer takes as input the information about images including image and image metadata. Each image has coordinates in the feature space, and the recommender system finds suitable images with similar content using the data prepared during the training on the graph. Users form the graph, thus the filtering type of this implementation is collaborative. The solution has several disadvantages related to the training: the necessity of the large initial sample (18 TB), significant computing resources, the necessity of relationship graphs between images. This solution doesn't consider user rating history and forms recommendations from the one chosen image.

In this research an algorithm based on the second approach was proposed. The proposed algorithm includes several improvements relative to existing solutions:

1) *No necessity of manual metadata configuration.* This improvement allows us to make the fully automated recommendation algorithm and to avoid errors, caused by human factor.
2) *Consideration of user rating history*. If a recommendation system recommends not only similar to the passed image another images and recommends images corresponding user interests (formed from his rating history) the recommendations are more complete. And we aren't able to compare the recommendation completeness (recall metric) of the proposed solution and another solutions not considering user rating history. Authors of these solutions [5]–[7] got metrics after execution of recommendation algorithm on the test data marked by the another method distinctive from marking method applied on the test data for proposed solution. Test data of these solutions includes images, which marked, as recommended, if they are

similar to the passed image. In the proposed solution we mark image as recommended, if it corresponds to user interests, that are formed from his ratings. Also we aren't to compare completess with the solution related with metadata analysis [3], because there are not provided results of this metric measurement.

Non-functional requirements were defined also relative to the proposed algorithm:

1) *Satisfactory precision and recall values*. Precision, that we expected in our research, is more than 0.6. Expected recall is more than 0.6 also. Recommended images in test data should consist around 5 percent of all images.
2) *Satisfactory time of recommendation calculation*. We expected calculation time less than 300 ms.
3) *No necessity in significant resources* like implementation of the approach based on analysis of hybrid features [7]. The configuration of typical PC should be enough to calculate recommendations in expected time limits (like computation resources of implementations of the metadata analysis and visual feature analysis approaches).

## III. Proposed Image Recommendation Solution

Consider the main idea of the proposed solution and what methods, algorithms, distinctive features the solution contains. The distinctive features of the proposed solution are:

1) *Object class recognition*, that allows us to do the text analysis instead of visual features analysis.
2) *Storing users, images, and topics in the graph structure*. Edges between topics and images allows us to consider object class probabilities. Edges beetween users and topics allows us to consider user interest weight. And edges between topics allows us to find the closest (and the most relevant) to user images. The proposed solution mostly specializes exactly on image recommendations, because it consider class probabilities of images and stores them in edge weights.

The scheme of the proposed solution is presented in Fig.1. Let's say, there is a set of images in some photo hosting. For each of the images, the recommendation system defines a list of images. Some of these images are rated by the user. A list of user interests is formed from the set of rated images. A list of image classes and a list of user interests are saved in the database. The recommender system generate recommendations based on user interests and information about object classes in images.

### A. Object Class Recognition

So, the proposed solution relates to the second approach (image visual features extraction and analysis). This approach was adjusted by using of 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, forms user interests. Thus the image recommendation

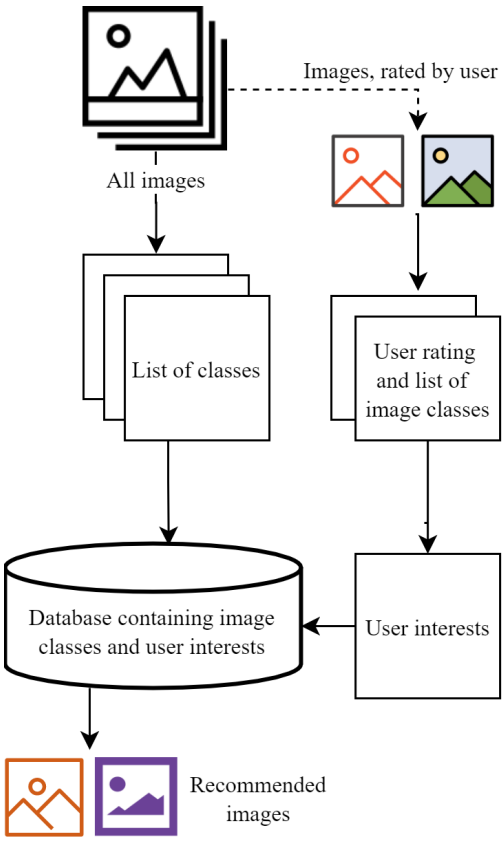


Fig. 1. Scheme of the proposed image recommendation solution

issue was transformed to the issue of recommendation based on the textual information about images with weights.

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

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

### B. Graph Consisting of Users, Images and Topics

In the proposed solution all object class IDs correspond to natural language words. Representation of object class in words allows us to find semantically close words by using the Word2Vec model [13] [14] or modifications (for example, GloVe [15]). The Word2Vec model is used in many recommender systems [9], [11], [12]. The application of Word2Vec in the current solution is based on the solution [11], where user coordinates in the semantic space are midpoints between items users liked. The content filtering in this solution is a search of the nearest to user items in the semantic space, collaborative filtering is a search of the nearest users in semantic space to current user and recommendation of items liked by them. This solution has two disadvantages:

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

Because of the solution [11] has disadvantages, the semantic space with images and users was converted to the graph $G$. The graph has a base constant part $G_T \in G$ containing topics (natural language words). And the recommender system adds images and users to the graph $G = (E, V)$. The simple example of the graph is presented in Fig.2. The graph $G$ contains a set of nodes $V$ containing a subset of images $I \in V$, a subset of topics $T \in V$, a subset of users $U \in V$. And the graph $G$ contains the 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 $E_{TU} \in E$.

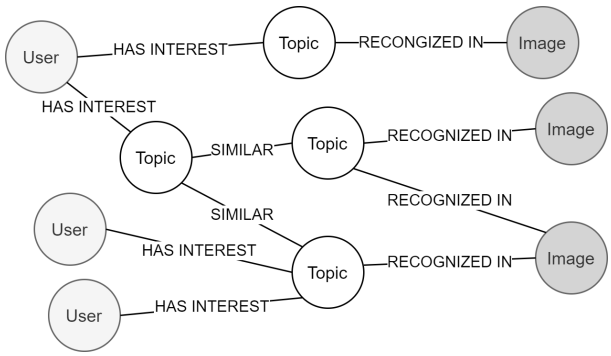


Fig. 2. The example of the graph with images, users and topics

The algorithm described in [10] was used to convert the semantic space containing natural language words to the base graph $G_T$. Authors describe the algorithm converting GloVe model [15] to graph. The nearest words are grouped to clusters, all cluster words are linked by edges to the central word. Central words should be grouped to clusters also and the clustering algorithm is recursive. The resulting graph is a set of connected to each other nodes of topics. The example with the two-layer cluster is shown in Fig.3.

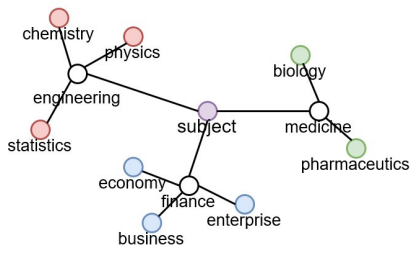


Fig. 3. Graph with the two-layer cluster

Each of the topic nodes $T_S$ contains a vector with coordinates of their words $W_S$ located in the sematic space $X \in \mathbb{R}^n$ of dimension $n$. Weight of edge between topics $T_A$ and $T_B$ is equal to the Euclidean distance (2), where $X_A$ are coordinates of the topic $T_A$ in the semantic space and $X_B$ are coordinates of $T_B$.

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

Images are connected by edges to topics, which are object classes for images. These edges weight calculated by a formula (3), $T_A$ is a topic $A$ and $I_B$ is an image $B$, $W_A$ is a natural language word 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 image $I_B$.

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

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

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

## IV. Image Recommender System

Consider the structure of the recommender system implementing the proposed solution, how the recommender algorithm works in detail.

The architecture of the recommender system is presented in Fig.4. The recommender system interacts with the external system (photo hosting service) by HTTP requests. “REST Controller“ receives requests from the external system, transfers parameters to other services. There are the following types of requests from the external system: “Save Image“, “Update User Interests“, “Calculate Recommendations“.

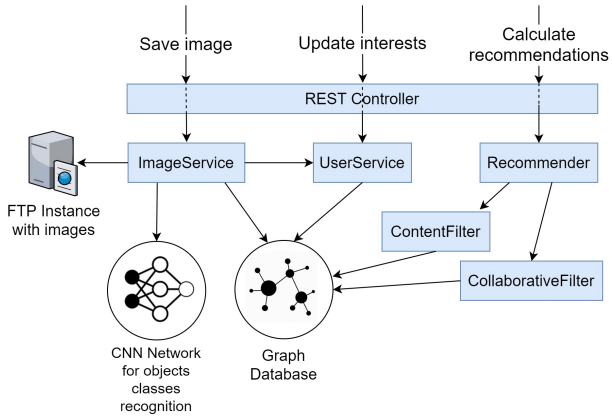


Fig. 4. Recommender system architecture

The component "ImageService" does:

1) Saving of image node to the graph database.
2) Object class recognition from images.
3) Saving of 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" forms recommendations from content and collaborative filtering results.

## V. Processing of External System Requests

### A. Processing of the "Save Image" Request

The sequence of actions on the receiving of the "Save image" request is described in Algorithm 1.

**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**

The recommender system creates an image node in the graph database (line 1). Then recommender system 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).

### B. Processing of the "Update User Interests" Request

The sequence of actions on receiving of "Update User Interests" request is presented in Algorithm 2.

**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**

The recommender system loads from the graph database topics related to the image and weights from these topics to image (line 1). Then the recommender system for each of the topics increases edges from user weight to the value of the user rating (lines 2–10).

### C. Processing of the "Calculate Recommendations" Request

The sequence of actions on the "Calculate recommendations" request is presented in Algorithm 3. The recommender system loads from the configuration the following parameters: K, M, N. The system finds K nearest to the user images in the graph (line 1). The obtained result is a result of content filtering. Then the recommender system finds M nearest to the current user other users (line 2) and topics that are of interest for the current user (line 3). The recommender system forms a list of interests, which another M nearest users have (lines 5–18). Thus, the new list of potential interests for the current user is formed. Then the recommender system filters interests by weight, only interests having share more $\alpha$ part are left. The recommender system finds the N next to these topics images in the graph (line 21). The obtained result is a result of collaborative filtering. The system returns the content and collaborative filtering results to the external system (line 22).

**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: $interest \leftarrow \forall interest_i : \dfrac{interest_i.weight}{\sum_{j=1}^{n} interest_j.weight} > \alpha$
20: $interestTopics \leftarrow$ map $interests$ array to topics array
21: $collabRes \leftarrow$ N next to $interestTopics$ image nodes $\in G$
22: **return** $contentRes \cup collabRes$

### D. Search of the Nearest Nodes Algorithm

Consider the lines 1–2 from Algorithm 3 in detail. The recommender system finds the nearest image nodes in line 1

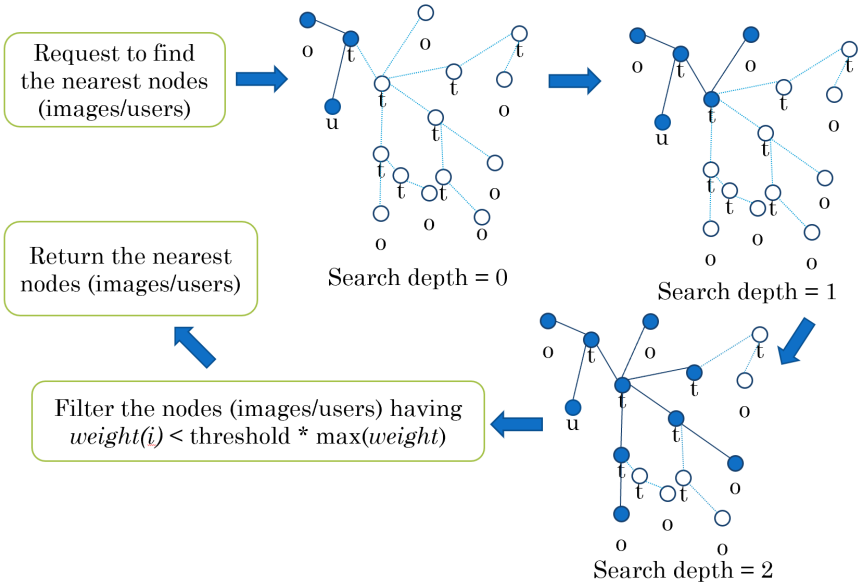


Fig. 5. Search of the nearest nodes

and the nearest user nodes in line 2 to the current user node. The principle of search is the same in these two lines and an example of search is schematically described in Fig.5.

$threshold$ and $search\ depth$ are two parameters specified in the configuration of the recommender system, and these parameters take part in the search of the nearest nodes.

We want to find the nodes in limits specified in the parameter $search\ depth$. As presented in Fig.5 the recommender system finds the nodes in limits of $search\ depth$ value equals to 2. In the beginning recommender system marks the node of the current user (signed with a char "u"). Then the recommender system marks the next nodes (topics, signed with a char "t") and neighbor images or users (signed with a char "o") to these nodes. The current search depth value equals to 0 in this iteration. Then the recommender system repeats this procedure and starts to move from the marked nodes. In each iteration recommender system increments the current search depth value while it will not reach the specified limit of $search\ depth$ parameter.

Then the recommender system filters the found nearest nodes $\beta_i$ (images for content filtering or users for collaborative filtering). For each of the found nodes $\beta_i$ the recommender system calculates $weight^i_{User}$ value by expression (5):

$$\begin{aligned} weight^i_{User} &= weight^i_{User\to Topic} + weight^i_{Topic\to Topic} + \\ &+ weight^i_{Topic\to\beta} = weight^i_{User\to Topic} + \\ &+ \gamma \times \sum_{j=1}^{k} weight^i_{Topic_j\to Topic_{j+1}} + weight^i_{Topic\to\beta} \end{aligned} \quad (5)$$

In this expression $weight^i_{User\to Topic}$ is a weight of edge from user to the neighbour topic, $weight^i_{Topic_j\to Topic_{j+1}}$ is a weight of edge from one topic to other topic, that consist a part of path from the current user to the found node $\beta_i$, $weight^i_{Topic\to\beta}$ is a weight of edge from topic to node $\beta_i$, $\gamma$ is a coefficient involving to sum $weight^i_{Topic\to Topic}$ terms share in the common sum (meaning how important are weights of edges between topics for the common sum $weight^i_{User}$). This sum considers three factors:

1) User interests ($weight^i_{User\to Topic}$).
2) Similarity of topics in the path ($weight^i_{Topic\to Topic}$).
3) Value of $weight^i_{Topic\to\beta}$, that will be a probability of the class (topic) in the image ($weight^i_{Topic\to\beta}$) in the case of content filtering and the weight of interest in the case on collaborative filtering.

Then the recommender system finds the maximum $weight^i_{User}$ and returns the nodes related with path from user, that has weight less than multiplying of $threshold$ parameter and maximum of $weight^i_{User}$. In other words, the recommender system filters the found in limits of $search\ depth$ nodes by $threshold$ parameter.

## VI. Used Technologies

A prototype containing content filtering was developed and it is planned to implement the collaborative filtering also. The prototype was developed on Java and was built on the Spring Boot platform. Medium GloVe model [15] containing 50000 natural language words and 100 coordinates in semantic space for each of words was converted to the graph of topics $G_T$. The graph was stored in Neo4j database. The recommender system system finds the nearest images for content filtering by execution of Cypher query. Google Vision AI service was used to recognize classes on images. It is planned to use Inception CNN model [18] locally to exclude network load related with sending requests to Google Vision AI service.

## VII. Recommender System Parameters and Obtained Metrics

Consider, how recommender system parameter values were defined and how the recommender system was tested.

A dataset containing 3 users, 1500 images and lists of user positive ratings was prepared manually, because there are no published datasets containing images of different topics and user histories. For example, the dataset "Pinterest Fashion Compability" [19] contains only fashion items. A set of images were collected from Google Images service with using of Selenium Webdriver framework. 100 topics were specified in csv file and for each of these topics 15 images were downloaded. There are 50 images defined for each of 3 users, that users already positively rated and that related with the one abstract topic for one user ("wild" for the first user, "art" for the second, "cities" for the third). There are recommended images manually marked also, that contain objects corresponding their interests. It is planned to collect dataset from the real deployed photo hosting.

Values of recommender system parameters were defined to achieve as more accurate and effective recommendations as possible for now, when we have only test data prepared manually. So, the found parameter values will be an initial approximation for more powerful optimization methods. When the recommender system will be deployed on the production environment, then we will have more reliable data, that will be used to find the best parameter values with using of optimization methods.

The $\gamma$ parameter was defined by the following requirement: it was preferrable to have values of $weight^{i}_{Topic \to Topic}$ between 0 and 1 (the most part of values). This requirement lets to to achieve the same influence of terms of the summing expression (5). The found value of $\gamma$ parameter that satisfied the requirement is 0.01 (initial approximation).

The recommender system classifies all images to two classes: recommended and not recommended. We can evaluate the quality of the proposed algorithm with using of methods for classifiers. Values for parameters $threshold$ and $search\ depth$ (that are used in the algorithm of the search of the nearest nodes) were defined by the method of complete enumeration to reach the reach the maximum values of precision and recall metrics (to reach the maximum of these metrics).

The source metrics are:

- TP - images that were recommended right (true positives).
- TN - images that weren't recommended, but should be recommended (true negatives).
- FP - images that were recommended, but shouldn't be recommended (false positives).
- FN - images that weren't recommended right (false negatives).

The precision metric was calculated by expression (6).

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

The chart of precision is presented in Fig.6.

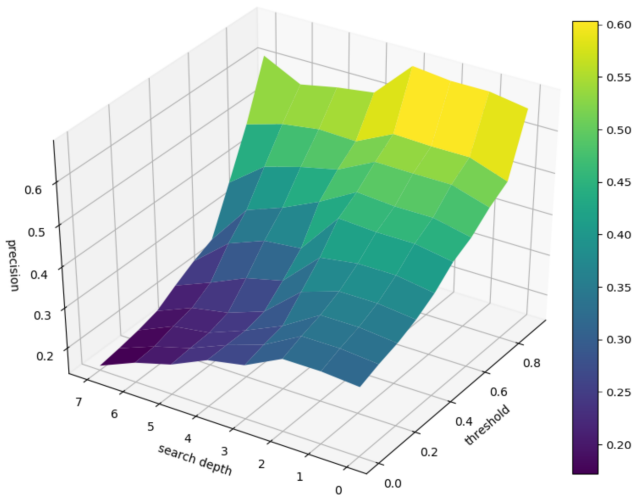


Fig. 6. Precision chart

The precision metric increases with increase of $threshhold$ value and decreases with $search\ depth$ more than 3.

The recall metric was calculated by expression (7).

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

The chart of recall is presented in Fig.7.

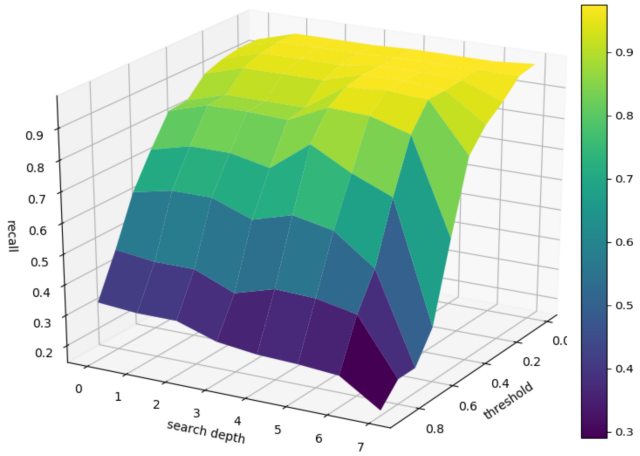


Fig. 7. Recall chart

The recall metric increases with decrease of $threshhold$ value and decrease of $search\ depth$.

The best point, where expected precision and recall were reached is: $threshold$ is equal to 0.8 and $search\ depth$ is equal to 4. Precision is equal to 0.61 and recall is equal to 0.72.

The time measurement was done on two computing nodes connected to the same local network. The recommender system was deployed on nodes having configuration of typical modern PC (see Table II).

The time measurement results are presented in Fig.8. The relatively rapid growth of time is observed since $search\ depth$

TABLE II
COMPUTING NODES OF THE RECOMMENDER SYSTEM

| | **Node 1** | **Node 2** |
|---|---|---|
| What was deployed | Application part | Database (Neo4j) |
| CPU | Intel Core i5-8250U | Intel Core i7-4702MQ |
| RAM | 8Gb | 8Gb |

is equal to 5. The almost same delay (around 300 ms) from 0 to 4 values of *search depth* is caused by the network load of connection to the graph database. So, when *search depth* is equal to 2, the recommender system isn't loaded significantly.

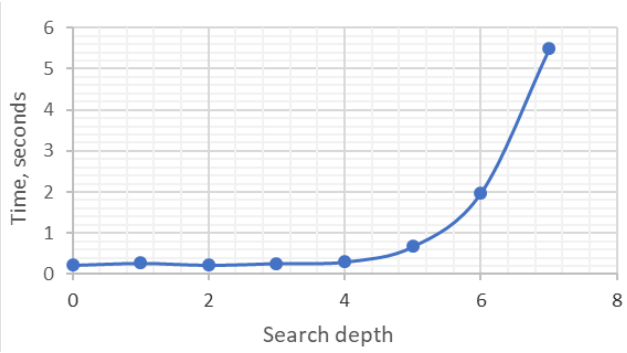


Fig. 8. Execution time chart

Thus the recommender system showed good results, but the one potential problem wasn't considered yet. In the future we planned to study the case when one topic contains significantly more images then other topics. If this topic will be close to the user, the recommender system might recommend almost all images of this topic and it isn't preferable. We want to modify the algorithm in a way, that recommendations were quite different. For example, we can take the first recommended image of this topic with the same weight, the second with less, the third even less and e.t.c.

## VIII. CONCLUSION

In this paper we analyzed existing solutions related to the automation of image recommendations. The proposed approach of automatic image recommendations addresses the following shortcomings of existing solutions: the necessity of manual configuration of metadata by users, lack of user interests consideration. The proposed algorithm eliminates shortcomings by recognizing images by locating users and images in the semantic space, and storing users, topics, and images in the graph. The proposed algorithm was implemented in the form of a correctly working prototype, that has satisfying precision, recall and execution time. It is planned to calibrate parameters of the algorithm to improve accuracy and execution time with using of real production data.

## REFERENCES


[1] F.O.Isinkayea, Y.O.Folajimib, B.A.Ojokoh, "Recommendation systems: Principles, methods and evaluation", *Egyptian Informatics Journal*, November 2015, pp.261-273.

[2] Houtao Deng, "Recommender Systems in Practice", *Towards Data Science*, February 2019.

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

[4] A.G. Gomzin, A.V. Korshunov, "Recommender Systems: modern methods review", in *Proc. of the Institute for System Programming of the RAS*, vol.22, 2012.

[5] Kemal Ozkan, Zuhal Kurt, Erol Seke, "Image-based recommender system based on feature extraction techniques", Sarajevo, Bosnia-Herzegovina: *International Conference on Computer Science and Engineering (UBMK)*, October 2017.

[6] Hessel Tuinhof, Clemens Pirker, Markus Haltmeier, "Image Based Fashion Product Recommendation with Deep Learning", Volterra, Italy: *International Conference on Machine Learning, Optimization, and Data Science Machine Learning, Optimization, and Data Science 4th International Conference (KDD)*, July 2018.

[7] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems", Gr. Britain: London: *24TH ACM SIGKDD conference on knowledge discovery and data mining*, June 2018.

[8] Stamatios Giannoulakis, Nicolas Tsapatsoulis, Klimis Ntalianis "Identifying Image Tags from Instagram Hashtags Using the HITS Algorithm", Orlando, FL, USA: *IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing*, April 2018.

[9] Oren Barkan, Noam Koenigstein, "Item2Vec: Neural Item Embedding for Collaborative Filtering", *arXiv*, March 2016.

[10] Thomas A. Trost, Dietrich Klakow, "Parameter Free Hierarchical Graph-Based Clustering for Analyzing Continuous Word Embeddings", Vancouver, Canada: *Association for Computational Linguistics*, pp.30-38, August 2018.

[11] Makbule Gulcin Ozsoy, "From Word Embeddings to Item Recommendation", *arXiv*, June 2016.

[12] Ramzi Karam, "Using Word2vec for Music Recommendations", *Towards Data Science*, December 2017.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, "Distributed Representations of Words and Phrases and their Compositionality", *arXiv*, October 2013.

[14] Nikiforov Igor, Drobintsev Pavel, Voinov Nikita, "A System Prototype for Real Time Automatic Fraud Detection in Text Data", Proc. of *XXI International Conference of Soft Computing and Measurement*, May 2018.

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

[16] Justin J. Miller, "Graph Database Applications and Concepts with Neo4j", *Proc. of SAIS*, 24, May 2013.

[17] Angira Amit Patel, Jyotindra N. Dharwa, "An integrated hybrid recommendation model using graph database", Indore, India: Proc. of *International Conference on ICT in Business Industry and Government (ICTBIG)*, November 2016.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions", Proc. of *The Computer Vision Foundation*, CVPR 2015.

[19] Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, Julian McAuley, "Complete the Look: Scene-based complementary product recommendation", *Proc. of CVPR*, 2019.