XQM: Interactive Learning on Mobile Phones

Alexandra M. Bagi[1], Kim I. Schild[1], Omar Shahbaz Khan[1],
Jan Zahálka[2][0000−0002−6743−3607], and Björn Þór Jónsson[1][0000−0003−0889−3491]

[1] IT University of Copenhagen, Copenhagen, Denmark
bjth@itu.dk
[2] Czech Technical University, Prague, Czech Republic
jan.zahalka@cvut.cz

Abstract. There is an increasing need for intelligent interaction with media collections, and mobile phones are gaining significant traction as the device of choice for many users. In this paper, we present XQM, a mobile approach for intelligent interaction with the user's media on the phone, tackling the inherent challenges of the highly dynamic nature of mobile media collections and limited computational resources of the mobile device. We employ interactive learning, a method that conducts interaction rounds with the user, each consisting of the system suggesting relevant images based on its current model, the user providing relevance labels, the system's model retraining itself based on these labels, and the system obtaining a new set of suggestions for the next round. This method is suitable for the dynamic nature of mobile media collections and the limited computational resources. We show that XQM, a full-fledged app implemented for Android, operates on 10K image collections in interactive time (less than 1.4 seconds per interaction round), and evaluate user experience in a user study that confirms XQM's effectiveness.

Keywords: Interactive learning · Relevance feedback · Mobile devices

1 Introduction

As media collections have become an increasingly large part of our everyday lives, both professionally and socially, the traditional interaction paradigms of searching and browsing have become less effective. Search requires users to have a clear idea of the outcome of the interaction, while browsing large collections is too inefficient. Many users have significant media collections on their *mobile devices*, often thousands of photos, that they wish to interact with. When considering intelligent interaction with these collections, many questions arise: How should the system be designed to best utilize the limited computing resources of the mobile device? How should users interact with the system to best make use of the limited screen space? How will regular mobile device users react to this novel interaction paradigm? In this paper, we explore these challenges.

The *interactive learning* paradigm has had a revival as a viable approach for intelligent analysis of media collections. In interactive learning, the system incrementally and interactively builds a model of users' information needs, by

continually using the model to suggest media items to users for feedback and then using that feedback to refine the model. In recent years, significant progress has been made towards scalable interactive learning systems, where the thrust of the work has been on handling larger and larger collections with moderate hardware whilst providing relevant results [17, 10]. Interactive learning is a good fit for mobile systems, as it inherently does not rely on heavy data preprocessing, bears limited computational resources in mind, and can work with dynamic datasets.

We present and evaluate XQM, a full-fledged mobile application for interactive learning.[1] Fig. 1 illustrates the overall architecture and processing of XQM. In the *interactive exploration* phase, XQM gradually builds an *interactive classifier* capable of fulfilling the user's information need. This is done by presenting the highest ranked images from the current model in the *user interface* and asking the user to provide feedback on those, labelling them as relevant or not relevant. The labels are then used to retrain the interactive classifier, and the classifier is in turn used to query the *feature database* for a new set of *suggested images* to judge. This interactive process continues until the user is satisfied or decides to start from scratch. To support the model construction, the app also has a *data processing* phase, where state-of-the-art *semantic feature extraction* is deployed on a dedicated server. When installing the app, existing images are analysed in this manner to build XQM's feature database, and subsequently the user can analyse new images to add to the feature database. In a performance study, we show that XQM can interactively explore collections of up to 10K images with response time of less than 1.4 seconds per interaction. Furthermore, we report on user experience in a user study with participants ranging from novice users to experienced interactive learning users.

2   Related work

Processing large image collections makes the collaboration between humans and computers inevitable. Computers have a large memory and can process large amounts of data fast, but they lack the human's ability to extract a great amount of semantic information from visual content. This phenomenon is known as the semantic gap [14]. A large body of research has been devoted to closing the semantic gap and indeed, we are able to automatically extract more semantic information from the data than before. To that end, however, a computer still needs feature representations of the data which are meaningful to the machine, but might not be to a human. Currently, mostly convolutional neural networks (CNNs) are used [11, 15, 16].

The need for feature representations presents two key challenges for a mobile approach. Firstly, the resource-limited mobile phone needs fast access to the feature representation in addition to the raw data. Secondly, CNNs are computationally very intensive, and their performance hinges on access to state-of-the-art

---

[1] XQM is an acronym of Exquisitor Mobile, as the design of XQM relies heavily on Exquisitor, the state-of-the-art interactive learning system [10]. The XQM app is available to the research community at www.github.com/ITU-DASYALab/XQM.
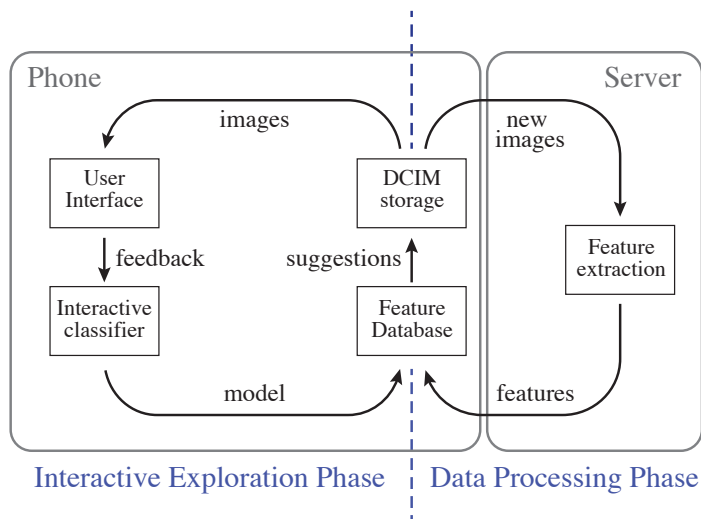
Fig. 1: Architectural overview of the XQM interactive learning mobile app.

GPUs that either mobile phones do not possess or are unsuitable for high long-time usage due to overheating. There are approaches for deep learning on mobile phones, but their performance lags behind the desktop-based state of the art.

Most recent approaches for analyzing multimedia collections are based on retrieval: an index is built on top of the feature representation(s) and users pose queries that retrieve results based on this index. There is a large number of approaches to build the index: to name a few, product quantization [9], clustering-based approaches [8], or hashing-based approaches [2, 5]. Overall, this approach works well, as the search engine is a familiar interface to the users, and the semantic quality of state-of-the-art representations is high enough to surpass human-level performance on some tasks [7]. However, an index-based approach might not be so suited for mobiles. Again, there is the limited resources challenge: an index is yet another data structure the phone needs access to. Also, multimedia collections on mobile phones tend to be highly dynamic, whereas index-based approaches favour static collections.

Interactive learning approaches, in particular user relevance feedback (URF), directly work with the user to obtain the items she finds relevant: in each interaction round, the user selects relevant and not relevant items, the interactive learning model retrains itself based on the judgment, and finally the user is presented with new and potentially relevant items for the next round. The bulk of the algorithmic research on relevance feedback and the closely related active learning techniques has been done in the 2000s [1, 13, 19]. Recent work has improved interactive learning to perform well on modern large-scale datasets: scaling up to interactive performance on 100 million images [17], and improving the performance to 0.29 seconds per interaction round whilst further reducing

computational requirements [10]. A good choice of an interactive classifier is the Linear SVM [4], which combined with modern feature representations yields good performance at a modest computational cost. In terms of applications, interactive learning approaches fall under the umbrella of multimedia analytics, which strives to iteratively bring the user towards insight through an interface tightly coupled with an interactive learning machine model [18]. These approaches do not rely on a static index and computational requirements are a built-in core consideration, namely interactivity (making sure the model is able to retrain itself and produce suggestions in sub-second time). However, it is still not trivial to satisfy those requirements, especially on a mobile device.

This work aims to implement a URF system on mobile phones, therefore both the technical restrictions of these devices and the general user behaviour towards mobile applications need to be considered. There are a number of factors that prevent simple deployment of the state of the art on mobile devices. Efficient interactive learning approaches, such as Exquisitor [10], hinge on a C/C++ implementation and storage solutions not supported by mobile OS (at least Android), and the machine learning (computer vision, information retrieval, etc.) codebase for mobiles is limited. Whilst this is an implementational challenge rather than a scientific one, it is a barrier nonetheless. Moreover, user interface interactions and their convenience are somewhat different than on a computer. For example, mobile UIs make heavy use of swipes and finger gestures which do not necessarily map directly to mouse interactions and typing is more cumbersome on a screen keyboard than on a computer keyboard. Lastly, the core framework components of a native mobile application largely stipulate the way how connection can be established between the user and the system.

3   XQM Architecture

In this section, we describe the architecture of the XQM mobile app, depicted in Fig. 1. We start by presenting the user interface which supports the interactive exploration process outlined in the introduction, and then consider the underlying components in a bottom-up fashion: the semantic feature extraction, the feature database, and the interactive classifier. Note that the actual photos are stored by other applications in DCIM storage folders. When starting XQM for the first time, the user must thus grant the app access to DCIM image storage.

User Interface: Fig. 2 shows the main screens of the user interface of XQM. Each exploration session starts with the *home screen* (Fig. 2(a)), which displays a non-scrollable list of 6 random images from the image collection (the largest number of images that can be displayed on the mobile screen in adequate resolution). The user can tap on any of the six suggestions to open an enlarged image on a new *feedback screen* (Fig. 2(b)) in order to inspect it in more detail, and then potentially judge the image as a positive or negative example by swiping the image right or left, respectively. Once an image has been judged as a positive or negative example, the user returns to the home screen where the image is
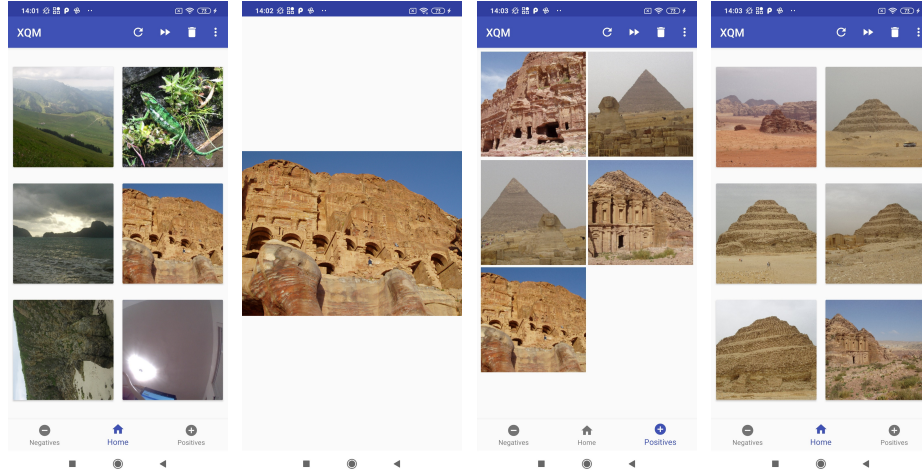
(a) Home Screen   (b) Feedback Screen   (c) Positive Images   (d) Resulting Model

Fig. 2: The interactive learning process as captured in the XQM user interface.

replaced by the image currently considered the most relevant, according to the interactive classifier.[2]

The user can, at any time, revisit the positive and negative examples by tapping on the corresponding buttons at the bottom of the home screen, to open the *positives screen* (Fig. 2(c)) or *negatives screen* (not shown). From those screens, the user can remove images from the positive or negative lists, which in turn impacts the model in the next suggestions round.

Once the interactive classifier seems good enough, the user can tap on the "fast-forward" icon at the top of the screen to fill the home screen with the most relevant images (Fig. 2(d)). In addition to the fast-forward button, the XQM app has three buttons located on the top navigation bar: the "random" icon is used to get a new set of random images;[3] the "trash" icon is used to start a new exploration from scratch; and finally the "overflow menu" icon (three dots) can be used to add new images to the feature database through the data processing phase, or get help as outlined in the user study in Section 5.

Semantic Feature Extraction: In the *data processing phase* of Fig. 1, semantic deep learning features are extracted from images and stored in a feature database. There are two options to facilitate feature extraction on mobile devices: on-device, using the CPU of the mobile phone (or GPU, in the case of

---

[2] In the current implementation, at least one positive and one negative example are needed; until these have been identified, random images replace the judged images.

[3] Loading random images is useful when the model is missing positive examples with concepts that have not yet been seen; in a future version we plan to implement search functionality to further help find positive examples.

high-end devices); or off-device, sending the images to be processed to a remote server. As outlined in Section 2, on-device processing suffers from lower semantic performance, low availability of tools/libraries, and risk of overheating causing damage to the device. The off-device approach, on the other hand, requires access to a mobile or wireless network, which may result in latency and/or usage charges [6], and can also raise security questions.

We have chosen the off-device approach for the XQM app, to (a) make use of state-of-the-art semantic features, and (b) avoid complex resource management issues on the device. We have wrapped a pretrained ResNext101 model with 12,988 classes with a web-API, which allows submitting a ZIP file with multiple images and returns a JSON file with information on the semantic features. Following [17], however, the server only returns information about the top s semantic features associated with each image (by default, $s = 6$).

During the data processing phase, the collection of images to analyse is split into batches of 100 images, which are compressed and sent synchronously to the server for processing. Upon receiving the JSON file from the server, it is parsed and the information is stored in the feature database described in the next section. When the app is first run, this process is applied to the entire collection of images on the device; subsequently, only newly added images are analysed when the user chooses to update the database.

Processing each batch of 100 images takes little over a minute with a Xiaomi Redmi Note 8 Pro smartphone and a laptop for running the extraction. Initialising the app on a mobile device with thousands of images would thus take significant time. For a production app more care must be taken to implement the data processing phase, including asynchronous and secure communication. The current process, however, is sufficient for the purposes of understanding the performance of the interactive exploration phase, which is the main emphasis of the paper.

Feature Database: The semantic feature data resulting from the analysis described above must be stored persistently on the mobile device in a format that allows efficient access in the interactive phase. However, neither the traditional multimedia approach of storing a sparse NumPy matrix in RAM nor the advanced compression mechanism of [17] are applicable in the limited environment of the Android OS. Instead, the standard approach to data storage is using the SQLite relational database, which requires careful normalisation and choice of data types to work well.

Fig. 3(a) shows the data that would typically be stored in (compressed) binary format in RAM. In SQLite, the IDs of the features and corresponding probabilities can be stored as a TEXT string that is parsed when reading the data to rank the images. While this approach has modest space requirements, requiring 179 kB for a database of 1,000 feature vectors, parsing the TEXT string resulted in computational overhead.

Instead, Figs. 3(b)-(d) show the final normalised SQLite database, where three tables represent the feature database, one for storing all relevant folder

ImageName	FeatureID	Probability
storage/emulated/0/DCIM/Camera/103501.jpg	[5344, 4309, 6746, 5060, 2874, 2705]	[0.102834791, 0.063476876, 0.05930854, 0.05230371, 0.03741937, 0.0310730178]
storage/emulated/0/DCIM/Camera/101502.jpg	[6248, 5326, 2851, 5324, 5325, 5326]	[0.122722, 0.116210177, 0.059408964, 0.045039124, 0.044916617, 0.0203401245]

(a) Original Table

FolderPathID	FolderPath
0	storage/emulated/0/DCIM/Camera
1	storage/emulated/0/DCIM/Screenshots

ImageID	ImageName	FolderPathID
0	103501.jpg	0
1	101502.jpg	0
2	103200.jpg	0
3	102200.jpg	0
4	101501.jpg	0
5	103301.jpg	0
6	103511.jpg	0

ImageID	FeatureID	Probability
0	5344	102
0	4309	63
0	6746	59
0	5060	52
0	2874	37
0	2705	31
1	6248	123

(b) Path Table   (c) Image Table   (d) Feature Table

Fig. 3: Comparison of storage alternatives: (a) the original unnormalised table; and (b)-(d) final normalised tables.

paths and their identifiers, one for storing image identifiers, image names and path identifiers, and the third for storing the feature identifiers and probabilities, one per row. For additional space savings, the probabilities have been converted to integers using multiplication, since the INTEGER data type requires half the storage of a FLOAT data type. With this implementation, a collection of 1,000 feature vectors requires only 116 kB.

When applying the interactive classifier to the feature vectors to rank images, only the feature table is required, as it contains the image identifiers that can be used to identify the most relevant suggestions. Once the 6 most relevant suggestions have been identified, the image table and the path table must be accessed to build the path of the image for presentation in the user interface. Since the access is based on the primary keys of both tables, reading the required data is very efficient.

Note that deletion of images is handled by detecting missing images as they are suggested to the user, and subsequently removing their information from the feature database. If an image is moved, it will be handled as a deletion and an insertion; the order will depend on when the user updates the database and when the moved image first appears as a relevant image.

Interactive Classifier: XQM uses the LIBSVM [3] implementation of linear SVM, which is considered the state of the art classifier in interactive learning [10, 17]. As the relevance judgments generally constitute a very small training set, we used generic parameter settings to avoid over-fitting. Once the model has been trained, the feature vectors are retrieved from the SQLite database, as outlined above, and fed to the model to produce a score for each image. The 6 images with the highest score, or farthest from the decision boundary on the positive side, are then returned as suggestions.

4   System Performance Evaluation

This section evaluates the efficiency of the mobile application. Due to space concerns, we focus primarily on the interactive phase, measuring the time required to retrieve new suggestions in each interaction round. We have also measured the one-time process of analysing image contents, only about one-third of the time is spent on the phone.

Experimental Setup: For the evaluation, we used a Xiaomi Redmi Note 8 Pro smartphone with 128 GB memory and 6 GB RAM, MediaTek Helio G90T 2.05 GHz (8-core) processor and Android 9.0 (Pie) OS. To obtain results that are not influenced by other design decisions, we constructed two stand-alone apps specifically for the experiments.

The first app randomly chooses positive and negative examples, retrieves their feature vectors from the SQLite database, computes the SVM model, retrieves all the features from the SQLite database, and computes the next six suggested images. The app takes three input parameters: number of feature vectors, ranging from 100 to 10,000; number of rated examples, ranging from 3 to 48; and number of suggestions to retrieve, set to 6. For each parameter combination we ran 300 iterations and report the average.

The second app repeatedly selects random images from the database and presents them on screen, as would be done in the XQM app. It takes as input a database of images, ranging in size from 100 to 10,000. To avoid warm-up effects, we first loaded 200 images, and then measured 1,000 images for each database.

We separated the two processes as the latter app is independent of many of the parameters of the first app. To simplify the presentation of results, however, we (a) only report results with 48 rated images, as computing the SVM model is efficient and is only more efficient with fewer suggestions, and (b) incorporate the time for presenting 6 images into the results from the first app.

Experiment Results: Fig. 4 presents the details of the time required for each iteration of the relevance feedback process. The x-axis displays the number of images and feature vectors in the database, while the y-axis shows the time that was used for completing these tasks in seconds. As the figure shows, the time required to build the model and show the final suggestions is negligible (only visible for the smallest collection) and the time to rank images is also very small, while the majority of the time is used to retrieve feature vectors, which is linearly dependent on database size, and display the images on screen which is independent of database size. As Fig. 4 furthermore shows, the process is very efficient for small databases, requiring less than 250 ms for 1,000 or fewer images, and that even for 10,000 images the total time per iteration is only about 1.4 seconds. Overall, we can conclude that for the vast majority of mobile phone users, the app will perform interactively.
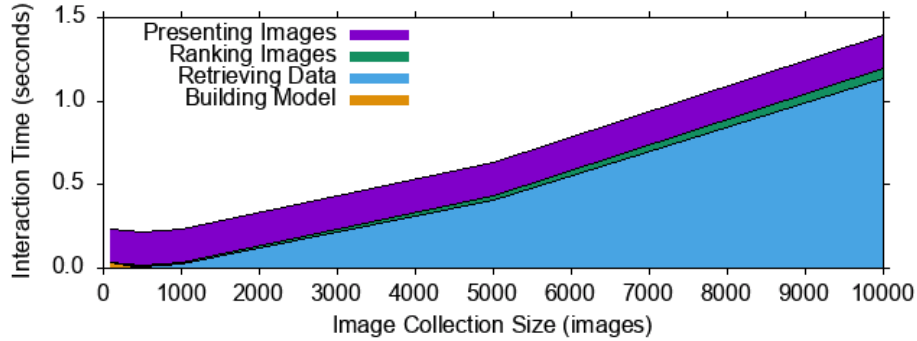
Fig. 4: The average time for subtasks in the interactive exploration phase.

5   User Interface Evaluation

XQM's design is intended to make the app easy to use for novice users who are not familiar with interactive learning. To evaluate the UI, we conducted a user study focusing on the app's usability, learnability and functionality.

Evaluation Setup: We recruited 8 users for the user study, 3 female and 5 male, all university students between 23 and 27 years of age. Three testers have limited technical knowledge, while the remaining 5 are CS students; 2 of the latter have worked with interactive learning in their thesis. All 8 users use mobile phone applications on a daily basis, but only 3 use an Android phone.

The users were provided with a Xiaomi Redmi Note 8 Pro with 2,883 images from the INRIA holiday data set[4] and the Lifelog Search Challenge.[5] The users were asked to conduct 3 sessions to find each target presented in Table 1: TV screen; exotic bay; and skyscrapers. Note that this setup is different from the intended use case scenario, as users have not created the collection themselves and thus had no information about its contents. However, to establish a consistent environment, it was necessary to use the same image collection for all users. To further guarantee comparability, an identical random seed was used to begin each interaction with the same set of random images.

The users received a brief verbal introduction to XQM. An on-boarding modal within the app was then used to describe its functionality and usage; users could return to this information at any time using the help menu. The sessions were audio-recorded. Users were able to complete the task in 22 out of the 24 sessions; the two unsuccessful sessions are excluded from the analysis.

Quantitative Analysis: Table 2 shows how many times users took each of the main actions to accomplish the task. Due to the high variation in results,

---

[4] http://lear.inrialpes.fr/people/jegou/data.php
[5] http://lsc.dcu.ie/

Table 1: Example target images and their occurrences in the collection.

Target	TV screen	Exotic bay	Skyscrapers
Occurrences in collection	~ 100	~ 30	~ 15
Example image			

both the mean and median are provided in the table. On average, the users gave between 7 and 20 images ratings (positive or negative) for the three tasks. Showing random images was often used to accelerate the sessions, although the value is heavily skewed from one session where a user hit random 84 times; essentially simulating scrolling through the collection. Table 2 also shows that fast-forward and correcting previous ratings were rarely used, while the trash icon for starting a new session was entirely disregarded.

An analysis of the session logs indicates that novices encountered some difficulties understanding the principle of interactive learning and thus XQM's purpose. Overall, though, the results indicate that the users were largely successful in solving the tasks; in particular the fact that users never started from scratch indicates that they generally felt they were making progress in the sessions. However, the fewer instances of the target class there are the longer the sessions take, and the (sometimes extensive) use of the random button indicates that adding search functionality to find positive examples would be useful.

Qualitative Analysis: We analysed the session logs to understand and classify the concerns and suggestions of users, and report on this analysis below.

Understandability and Learnability: Most users refrained from using the buttons, mostly because they did not remember or understand their functionality. Three users had difficulties understanding which images were replaced and said they would like to understand how the model evaluates images in order to make better rating decisions. The two users with prior interactive learning experience, however, did not report such problems. One of them said: "I think XQM's strong suit is its simplistic UI. That makes it a lot easier for novices to learn." And: "The modal gave very good information of how the app works." Both experienced users said that they would like to have a separate screen containing a history of their search.

App Functionality: A number of suggestions were raised regarding the functionality of the app. Five users desired an "undo" function to be able to reverse their actions; such user control is desirable to allow the user to explore the app without fear of making a mistake [12]. To accelerate the sessions, two users expressed a desire to be able to scroll through the collection, since 6 images are

Table 2: Mean/median of actions per user per motive

Target	Feedback	Random	Fast-forward	Start over	Rating change
TV	7,0 / 6,0	1,0 / 0,6	0,0 / 0,0	0,0 / 0,0	0,6 / 0,0
Bay	13,0 / 11,5	1,0 / 1,0	1,7 / 1,0	0.0 / 0,0	0,7 / 0,0
Skyscrapers	19,5 / 11,0	19,8 / 8,0	3,8 / 0,5	0,0 / 0.0	1,3 / 0,0

only a small fraction of most mobile collections, and three users suggested to incorporate a search function for finding relatively rare image motives. Furthermore, two users said that they would like to swipe directly in the home screen: "It would be faster than opening the image every time you want to rate it. This would be very useful for eliminating irrelevant results." Nevertheless, after the introduction, users quickly learned how to rate: "I like the swiping feature to rate the image. That is very intuitive."

Image Features: Users noted that some images score constantly higher in the suggestion list than others without being perceived as relevant to the task. Traditional techniques, such as TF-IDF, have been reported to improve the relevance of the returned items [17] and could be relevant here. Nevertheless, overall users could find relevant results despite this issue: "It is exciting to see that the model caught up on what I am looking for."

Discussion: Overall, the results of the user study indicate that XQM succeeds in its goal of implementing interactive learning on a mobile device. Despite variation in the approach and performance of users, they were overwhelmingly able to complete the assigned tasks and find examples of the desired target items. The results also point out a number of improvements to make, including search and browsing to find positive examples, an undo button to easily take back an action, rating directly in the home screen, and improving the representative features to avoid some images occurring in every interaction session.

6   Conclusion

In this paper we presented XQM, a full-fledged mobile app for user interaction with media collections on the user's mobile device. Our interactive learning based approach is demonstrated to operate well on the dynamic mobile collections and use modest computational resources, which are at a premium on mobile devices, whilst providing relevant results in interactive time of less than 1.4 seconds on a 10K collection. The user study confirms that XQM is a useful tool not only to the experienced users, but also for the novice or casual users unfamiliar with interactive learning, with clear potential for further improvement. With XQM, we hope to have opened new avenues for research on advanced, intelligent approaches for media collection analytics on mobile devices.

References

1. Aggarwal, C., Kong, X., Gu, Q., Han, J., Yu, P.: Active learning: A survey. In: Data Classification, pp. 571–605. CRC Press (2014)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (Jan 2008)
3. Chang, C., Lin, C.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011)
4. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
5. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. SCG. pp. 253–262 (2004)
6. Ensor, A., Hall, S.: GPU-based image analysis on mobile devices. CoRR **abs/1112.3110** (2011), http://arxiv.org/abs/1112.3110
7. Geirhos, R., Temme, C.R.M., Rauber, J., Schütt, H.H., Bethge, M., Wichmann, F.A.: Generalisation in humans and deep neural networks. In: Proc. NIPS. pp. 7538–7550 (2018)
8. Guðmundsson, G.Þ., Amsaleg, L., Jónsson, B.Þ.: Impact of storage technology on the efficiency of cluster-based high-dimensional index creation. In: Proc. DASFAA. pp. 53–64 (2012)
9. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE PAMI **33**(1), 117–128 (Jan 2010)
10. Khan, O., Jónsson, B.Þ., Rudinac, S., Zahálka, J., Ragnarsdóttir, H., Þorleiksdóttir, Þ., Guðmundsson, G.Þ., Amsaleg, L., Worring, M.: Interactive learning for multimedia at large. Proc. ECIR pp. 495–510 (2020)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proc. NIPS. pp. 1097–1105 (2012)
12. Nielsen, J.: 10 usability heuristics for user interface design (1995), https://www.nngroup.com/articles/ten-usability-heuristics/, Last accessed on 25.03.2020
13. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
14. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE PAMI **22**(12), 1349–1380 (2000)
15. Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. CVPR. pp. 1–9 (2015)
16. Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proc. CVPR (2017)
17. Zahálka, J., Rudinac, S., Jónsson, B., Koelma, D., Worring, M.: Blackthorn: Large-scale interactive multimodal learning. IEEE TMM pp. 687–698 (2018)
18. Zahálka, J., Worring, M.: Towards interactive, intelligent, and integrated multimedia analytics. Proc. IEEE VAST pp. 3–12 (2014)
19. Zhou, X., Huang, T.: Relevance feedback in image retrieval: A comprehensive review. Multimedia Syst. **8**, 536–544 (2003)