Christian Koch, Ganna Krupii, David Hausheer, Proactive Caching of Music Videos based on Audio Features, Mood, and Genre
In Proc. of ACM Multimedia Systems (MMSys), 2017
ISBN: 978-1-4503-5002-0

Proactive Caching of Music Videos based on Audio Features, Mood, and Genre

ABSTRACT

The preferred channel for listening to music is shifting towards the Internet and especially to mobile networks. Here, the overall traffic is predicted to grow by 45% annually till 2021. However, the resulting increase in network traffic challenges mobile operators. As a result, methods are researched to decrease costly transit traffic and the traffic load inside operator networks using in-network and client-side caching. Additionally to traditional reactive caching, recent works show that proactive caching increases cache efficiency. Thus, in this work, a mobile network using proactive caching is assumed. As music represents the most popular content category on YouTube, this work focuses on studying the potential of proactively caching content of this particular category using a YouTube trace containing over 4 million music video user sessions. The contribution of this work is threefold:

First, music content-specific user behavior is derived and audio features of the content are analyzed. Second, using these audio features, genre and mood classifiers are compared in order to guide the design of new proactive caching policies. Third, a novel trace-based evaluation methodology for music-specific proactive in-network caching is proposed and used to evaluate novel proactive caching policies to serve either an aggregate of users or individual clients.

1. INTRODUCTION

The global mobile data traffic is predicted to grow by 45% annually till 2021 [1]. At this time, video will account for about 70% of the overall mobile traffic. This trend is supported by an increasing use of data-intensive multimedia services like YouTube, Spotify and Netflix, as well as increasing 4G network coverage. The increasing demand of users requesting videos, e.g., on their smartphones creates a gap between the available bandwidth and user demand [21]. For example, more than half of YouTube' s videos are watched mobile[1] and about 82% of all YouTube users watch music videos [10]. Content Delivery Networks (CDNs) like Google Global Cache or Akamai help delivering content from closeby caches and, thereby alleviate the content provider from traffic. A recent development, saving network traffic, is content caching by mobile apps, e.g., Spotify or Google Music which reactively store requested music tracks on user devices. Proactive caching is a novel advancement in the area of network content caching. Thereby, content is prefetched based on, e.g., social information or content properties in a dedicated share of the cache, while the remaining share is managed reactively, e.g., by traditional LRU. In a mobile network scenario it is likely to have many in-network caches deployed at different locations, e.g., at base stations or per metropolitan area of a country. Hence, in comparison with CDNs, it is likely to have more but smaller caches serving fewer users. As music represents the most popular content category on YouTube and 37% of video requests [16] in mobile network, in this work, novel proactive caching policies using music features for efficient music content placement are proposed and evaluated for in-network as well as client-side caches. Therefore, low-level features, e.g., tempo, timbre, pitch as well as high-level features, e.g., mood and genre are used. This information can be easily determined and offered by the content provider or CDN, as it is common practice for them to already analyze their offered contents for recommendation, e.g., per individual user or per geographic region. This work states three contributions. First, requests and audio features of music videos are derived and analyzed for their potential on proactive caching. Second, novel proactive caching policies using the aforementioned audio features are proposed. Third, a thorough trace-based simulative evaluation of recommendation strategies for proactive caching policies on in-network as well as client-side caches is conducted using a real network trace covering two weeks.

*since May 2017: Multimedia Communications Lab, Email: Christian.Koch@kom.tu-darmstadt.de

[1]https://www.youtube.com/yt/press/statistics.html

The remainder of this paper is structured as follows: Sec. 2 discusses related work. Sec. 3 analyzes the dataset used for evaluation. Sec. 4 presents the methodology used and explains how mood and genre of a music video are classified. In Sec. 5, proactive caching policies are proposed. Sec. 6, presents the results of the proactive caching policies for in-network as well as client-side caches. Finally, the paper is concluded and potential future work is described in Sec. 7.

2. RELATED WORK

The related work consists of three parts. First, an emotion model which is commonly used for mood classification is introduced. Second, relevant mood and genre classification approaches are discussed. Third, related papers in the areas of caching, proactive caching, and prefetching are presented.

2.1 Music classification

Music classification aims for automatic assignment of a certain class out of a pre-defined set of classes, e.g., genre or mood to a given music audio sample. Thereby, they derive high-level features such as genre or mood from characteristic low-level audio features. A well-known group of such features is named MFCC (Mel Frequency Cepstral Coefficients) [17]. In most of the related works discussed, feature extraction frameworks are used to determine low-level features. Most commonly used frameworks in scientific literature are depicted in Table 1 together with the number of audio descriptors, and the year of their last update.

To classify the emotion or genre of a music track, the following procedure is widely used: First, a representative sample of the music file is selected, e.g., seconds 30-60 to avoid the often not representative, intro. Second, the loudness is normalized in order to make loudness-sensitive metrics comparable between different tracks. Third, music features are derived using one of the feature extraction frameworks shown in Table 1. Fourth, a machine learning model, e.g., a Support Vector Machine (SVM), regression, clustering, or nearest-neighbor search is trained which is able to classify a mood state based on music features.

Genres are well-defined classes to which a music track can belong to. In contrast to this, mood states are more complex to represent. One widely accepted model representing mood states is Thayer's mood model [23]. As depicted in Fig. 1, the model defines a mood by a point in a 2D coordinate system with a certain intensity of valence on the x-axis and arousal on the y-axis. The arousal value is defined as the intensity of the emotion, while the valence value refers to how positive or negative the emotion is perceived. For different domains, variants of this model have been derived. A common representation is depicted by Fig. 1.

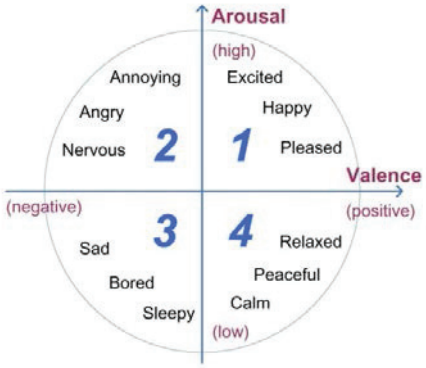


Figure 1: Thayer's mood model[27]

Mood Classification

Yang et al. propose a fuzzy approach for music emotion recognition [28]. Fuzzy classifiers are characterized by not only computing the most probable class but returning a fuzzy vector containing the probability of the music sample belonging to each of the classes. They evaluate two classifiers: fuzzy k-nearest neighbors and fuzzy nearest mean. Therefore, a dataset containing 195 popular music samples of 25 seconds length is used. Each sample is manually annotated with a mood. The four mood classes considered represent the four quadrants of Thayer's mood model. Additionally, to track mood changes within a song, a track is split in 10 seconds samples overlapping by $\frac{1}{3}$ of the previous sample and the mood for each sample is evaluated. The results show that a fuzzy nearest mean classifier with an accuracy of 78.33% is superior to fuzzy k-nearest neighbors.

Trohidis et al. [24] extend Thayer's mood model by two dimensions: pleasant/unpleasant and engaging/disengaging. The authors select a multi-label classification approach, using 6 labels: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-fearful. The dataset used consists of 593 expert-annotated songs from the genres Classical, Reggae, Rock, Pop, Hip-Hop, Techno, and Jazz. These genres serve as features for their multi-label classifier evaluation. As classifiers, they choose binary relevance (BR), label powerset (LP), random k-labelsets (RAKEL), and multilabel k-nearest neighbor (MLkNN). Amongst these classifiers, RAKEL provides the highest prediction accuracy with 80% but different accuracies for each class.

Laurier et al. [15] categorize music into one of Thayer's mood model quadrants by using SVM, Decision Trees and Random Forests, k-nearest neighbors, logistic regression, and Gaussian mixture models. Out of the chosen models, a SVM with polynomial kernel achieved highest mean accuracy of 90.44% for the categories angry, relaxed, and sad.

The highest accuracy is achieved by [15] compared with [19, 6, 27, 22], even though they also use SVMs. Table 2 summarizes mood classification works showing their best performing algorithm and the corresponding accuracy. Most works achieve a high accuracy by manually annotating music tracks and using a small training set in the range of a few hundred tracks only. However, [4] achieves only 25% accuracy by using on average ten mood classes. Therefore, the classification performance is assumed to vary strongly depending on the data and algorithm used as well as on the number of classes chosen.

Table 1: Music feature extraction frameworks

Framework Name	#Descriptors	Last update
MPEG-7 descriptors [11]	17	2004
Marsyas [25]	30	2015
jAudio [18]	40	2009
MIRtoolbox [14]	55	2014

Table 2: Music mood classification approaches showing the best performing algorithm of each paper

Paper	Algorithm	Accur.
Laurier et al. [15]	SVM with polynomial kernel	90.44%
Trohidis et al. [24]	Random k-label set	76-90%
Rho et al. [19]	SVM with radial kernel	87.8%
Han et al. [6]	SVM with polynomial kernel	87.78%
Eerola et al. [3]	Partial least squares regression	75-85%
Yang et al. [28]	Fuzzy nearest mean clustering	78.33%
Yang et al. [27]	SVR	58.3%
Song et al. [22]	SVM with polynomial kernel	54%
Gillhofer et al. [4]	Random Forest	25%

Genre Classification

Gillhofer et al. [4] collect a dataset of 7,628 listening events to 4,149 music tracks, obtained through a mobile app provided in the scope of a user study. For each event, time, location, weather, device, network, and motion are logged. Additionally, they acquired the genre and mood information for each track from last.fm. For genre classification, k-NN, decision tree and random forest, rule learner, and ZeroR are evaluated. They achieve an accuracy of about 60% for the genre prediction using a decision tree approach.

Huang et al. [9] propose a genre classification system using separate feature-selection for each genre class. The features used are intensity, pitch, timbre, tonality, and rhythm. For each pair of two genres, a local feature set is derived by their self-adapting harmony search (SAHS) algorithm. To get accurate results even for ambiguous genres, multiple one-against-one SVM classifiers are trained. The final classification is computed by a classifier ensemble containing the aforementioned SVMs. Evaluations of multiple strategies are conducted on the GZTAN dataset[2] published 2002. They achieved an accuracy of 97.2% for ten different music genre classes. This is an 13% increase compared with just using the original feature set.

To the best of our knowledge the work presented in this paper is the first deriving audio features from a recent dataset containing video requests from a mobile network to YouTube. The data used for genre classification is more recent than, e.g., the GTZAN dataset used in [9] and considerably larger with over 4 million requests, than the datasets used in the related work for mood classification.

2.2 Proactive Caching

Filling caches proactively, i.e., by prefetching objects, has been thoroughly investigated in the area of CPU caching. In the domain of network caching, recently papers have been published on the effects of considering, e.g., content age or the specific popularity distribution of the requested objects. It has been shown that proactive caching policies can be superior to reactive policies such as LRU.

An announcement-based caching approach for on-demand videos is presented by Claeys et al. [2]. By respecting the temporal structure of video segment requests, as well as the chance that a user watches multiple episodes of a series consecutively, announcements are created to inform the caching policy in advance. The evaluation is based on a dataset from 2010 containing 108,392 requests to 5,644 unique videos, which are assumed to be 50 minutes long, using 1 Mbit/s. Simulations are conducted using a realistic network topology and assuming an exponential distribution of video session lengths. Thereby, the authors respect that most videos are only watched partially [12]. By considering announcements of the videos a user is going to watch in the near feature, the cache hit ratio is increased by 11% compared with LRU. In contrast to the paper proposed, the authors of [2] consider episodes of a VoD portal dedicated to TV series which results in a small and homogeneous content catalog compared with YouTube which is used in this paper.

Hasslinger et al. [7] compare LRU with statistic-based caching strategies for Zipf-distributed popularity. They propose Score-gated LRU, defining a score for each object by its popularity. Thereby, the items in the cache are kept constant as long as content popularities do not change. This is beneficial over LRU which always loads every newly requested object into the cache, if not present already. Using score-gated LRU, an object is only inserted in the cache if its score surpasses the lowest score of all objects in the cache. By implementing a variant of score-gated LRU, the authors achieve about 10% hit rate increase compared with LRU.

Several works have investigated the potential of prefetching videos on mobile devices based on social network information, e.g., SonNet [26], CPSys [5], and O^{2}SM [29]. CPSys is designed for mobile video prefetching of YouTube videos. The system consists of two main modules, a prefetcher agent running on smartphones and a central predictor which informs the agent which videos should be prefetched. The central predictor keeps track of all user requests and determines the most similar users to a given user, i.e., the nearest neighbors by using the Jaccard index as a similarity measure. The number of videos to prefetch is determined by the number of videos requested over the last 10 days for each user separately. The videos to prefetch are selected, per user, by a queue containing all the videos requested by a user's neighbors. This queue is ordered firstly by popularity and secondly by recency if there are multiple items with the same popularity. Downloading of the videos is only conducted when a Wi-Fi connection is available. Overall 18-20% of correct prediction ratio are achieved by CPSys. However, in the analysis of CPSys, music videos are explicitly excluded as they show a different and more persistent popularity pattern compared with other video categories. Since the requests to music videos constitute the major share of YouTube requests in mobile networks, ignoring music leaves a gap which the paper proposed fills.

O^{2}SM is a middleware for smartphones predicting promising videos using a machine learning approach taking the user's Facebook feed as an input. O^{2}SM uses commenting, sharing, liking of posts, the number of private messages exchanged, the number of viewed videos from friends or pages, and the global post popularity of Facebook to determine promising prefetch candidates. To derive the user engagement, their own Facebook app needs to be used, which introduces a bias, since the post ordering as well as the look-and-feel differs from the native Facebook client. Furthermore, most of the videos on OSNs are only watched partially [12] which is not considered by their approach.

In contrast to related works using social relationships, this work analyzes the potential of content-based and user-based recommendation for proactive caching. As this information is more likely to be available at content providers and CDNs compared with social information, it is more practical to use.

[2]http://marsyasweb.appspot.com/download/data_sets/

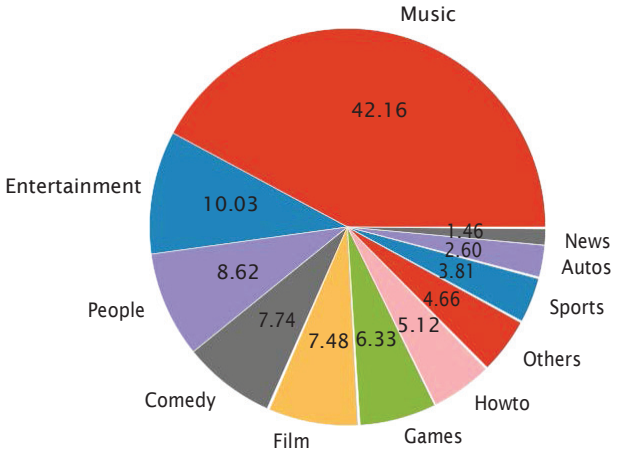


Figure 2: Requests to different YouTube categories

3. DATASET ANALYSIS

The trace used was collected between Monday 14th and Sunday 27th of April 2014 at the GGSNs (GPRS Support Nodes) of a large European mobile network operator covering a whole country. It contains over 10 million requests to YouTube caused by 700k users while being connected to a mobile network. Overall, 1.6 million different videos have been requested. The captured requests are anonymized at an early processing step and contain only unencrypted HTTP GET requests to YouTube. At this time, this is assumed to be about half of all requests to YouTube.

3.1 Content Analysis

The trace contains YouTube videoIDs which are used to enrich the dataset by meta data provided by the YouTube Data API [3]. This meta data provides information, e.g., about the video upload date, the category assigned by the uploader, as well as the video title. In a first step, based on the category, the requests belonging to videos with the category music and to other categories are determined. Music is the most popular category in the trace used causing about 42% of all requests. In a previous study [16] on YouTube category popularity, just 37% of all requests belonged to the category music. Therefore, an increasing trend towards music video watching on YouTube is likely.

The share of requests belonging to the ten most popular categories is shown in Fig. 2. Categories of minor popularity, i.e., with a request share smaller than 1% are summarized in the category *Others*, which contains: Movies, Trailers, Shows, Nonprofit, Animals, Travel, Tech, and Education. Music is the largest category w.r.t video views, more than four times larger than the second largest category Entertainment with just 10.03%. Another interesting finding is that about 35% of the YouTube channels appearing in the dataset have uploaded videos belonging to the category music.

In the following, only music videos which have been requested at least ten times within the two week trace are considered as they are most relevant for caching systems. Thereby, videos with a lower popularity, belonging to the video popularity distribution's outer short tail are removed. This results in 44,704 different remaining videos being used.

[3]https://developers.google.com/youtube/v3/

3.2 User Analysis

The network load measured by the number of user requests is shown in Fig. 3. For each day of the first week contained in the trace, the number of requests per ten-minute time intervals is depicted. It can be clearly seen, that on weekdays, the traffic peeks short after noon, probably because of people watching videos during their lunch break. After end of work, at around 5pm, the load shows a second peak, probably while commuting in public transportation. The load stays high but slowly decreases until around 5am. Weekend days show a different pattern compared with weekdays. Here, the users tend to start watching later at the day and request more videos overall. No dedicated peaks can be observed, instead the traffic stays high between 11am and midnight. Overall, the user activities seem to be shifted in time about 2 to 3 hours, as they start requesting later and stay active for later hours.

Recommending content for users watching only a few videos is hard and less effective in a network scenario where bandwidth reduction is the major goal. Therefore, so-called heavy users are selected out of all users. They are defined by watching at least 2 videos per day for at least 7 days within 2 weeks. This results in 5,351 heavy users representing 1.64% of all users but cause 15.56% of all requests to music videos. On average, they watch 7 videos per day.

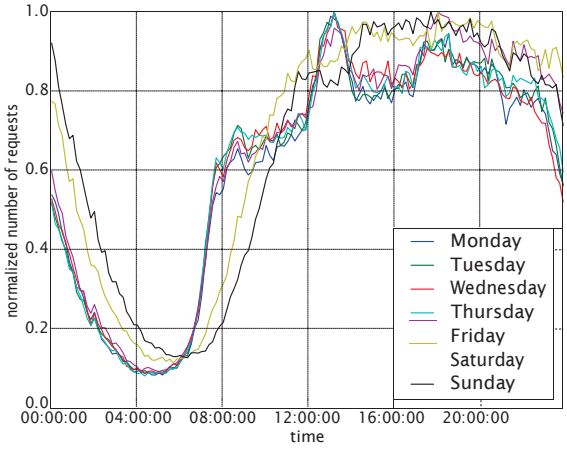


Figure 3: Normalized number of requests observed for each weekday and the hours of the day

4. METHODOLOGY

In this section, an overview of the proactive caching methodology is given and the approach used to derive genre and mood labels for each music track is explained. The methodology's main components are depicted in Fig. 4. The user video queries extracted from the YouTube trace serve as an input. In a first step, for each music track, the corresponding tags from last.fm are requested (ref. Sec. 4.1) and audio features are extracted from the video (ref. Sec. 4.2). Next, mood and genre classifiers are trained on the low-level audio features (ref. Sec. 4.3). The classifiers with the highest accuracy are used and allow labeling each music track with a genre and a mood, based on the low-level audio features only. Therefore, even for tracks that are unknown on last.fm genre and mood information can be determined. The video's

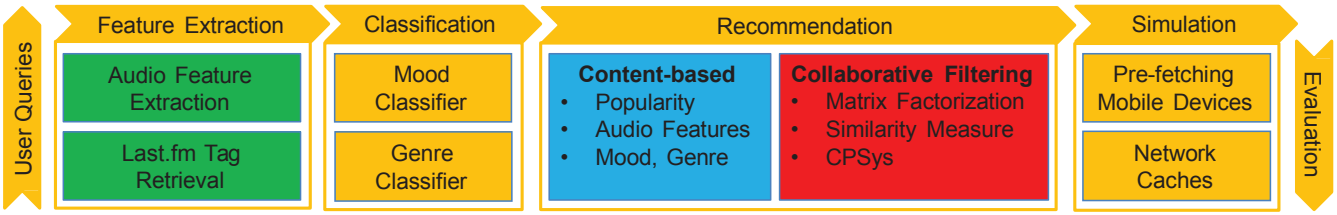


Figure 4: Methodology of the proactive caching design workflow

audio features, as well as classified genre and mood serve as an input to the recommendation component. The idea is to fill a certain share of the cache actively according to the videos recommended by one of the recommendation approaches, which is described in detail in Sec. 5. Furthermore, two variants of caching systems are considered: in-network caching, e.g., at a reverse proxy cache and caching on the user premises, e.g., at a smartphone.

4.1 Genre and Mood Retrieval from last.fm

The performance of the proactive caching approach depends on the correct information for the music videos. In order to determine genre and mood for each music video, the correct labels have to be determined. Therefore, the title of each video is used to request the tags annotated to this title by last.fm users. On last.fm, users can assign labels to a music track indicating, e.g., mood and genre of a song, but may also refer to the song's topic as they are free to choose a label. In a pre-processing step, strings like "official clip", "officiel", and similar strings are removed from the track titles. Furthermore, in case a " - " surrounded by spaces occurs in the title, the preceding part is assumed as the artist, while the latter one is considered as the song title. With this cleaned titles, all tags associated to them are retrieved from the last.fm API [4]. Thereby, for 13,553 tracks, tags are retrieved. Overall, these are 30% of of the 44,704 tracks considered. For very new or unpopular songs, it is less likely to find information on last.fm. To assign a mood and genre label for the other tracks as well, low-level audio features are derived and a genre as well as a mood classifier are trained on this features in conjunction with the dominant genre or mood information obtained from last.fm. This allows determining genre and mood also for tracks for which no information could be retrieved for.

Deriving Mood

The platform last.fm allows users to freely assign tags to songs. This results in a wide variety of tags. Therefore, only tags with a last.fm-specific weight of at least 50 are considered in the mood classification, in order to avoid rarely used and less representative tags. The information how this weight is computed, is not made public by last.fm. One category per quadrant of Thayer's mood model, namely: happy, sad, angry, and relaxed are used as classes, following the works of [15, 20, 28]. For each of the most often used tags, a quadrant of the Thayer mood model is assigned manually. Associated tags are used to group the tracks, as described in Table 3. For example the tags: angry, aggressive, and banger are assigned to the class: ***angry***. This results in a labeled dataset with the low-level audio features and their corresponding genre and mood category. The classifier is trained and tested with this dataset, which is considered as the ground truth.

Table 3: Subset of associations between the quadrants of Thayer's mood model and last.fm tags

Happy	Sad	Angry	Relaxed
happy	sad	angry	relaxed
energetic	nostalgia	aggressive	calm
positive	depressive	banger	downtempo
fun	bittersweet	passion	chillout
cheerful	sentimantal	quirky	dreamy
humorous	melancholic	annoying	longing
feel good	dramatic	gangsta rap	spiritual

Genre Classification

Following the approach in [9], this work uses the following genre classes: rock, classical, pop, blues, jazz, country, disco, hip hop, metal, and reggae. Additionally, the categories chanson, dance, electronic, and soul are considered as last.fm reveals that a large amount of tracks in the trace used belong to this categories which are not reflected in the aforementioned set of genre classes. In the following, a list of similar genres that are aggregated to one meta-genre is given. For example, tracks with the dominant labels hip-hop, hiphop, or rap are assigned to the meta-genre rap. For each track, the genre label assigned by most last.fm users is chosen. Overall, following this approach, the genre could be determined for 9,029 tracks. The number of samples per category are highly heterogeneous, e.g., Pop with 2,004 and Blues with 55 samples, as shown in Table 4.

- **Metal**: metal, heavy metal
- **Rap**: hip-hop, hiphop, rap
- **Reggae**: reggae, reggaeton
- **Rock**: rock, classic rock
- **Soul**: soul, rnb

4.2 Audio Feature Extraction

In the following, the method for low-level audio feature extraction is described. To obtain these features from each music video, the Matlab package MIRtoolbox [14] (ref. Sec. 2.1) is used, as it allows deriving a wide range of audio features from an audio signal, e.g., the MFCC values, tempo, spectral entropy, timbre, and pitch. Furthermore, statistics such

[4] http://www.last.fm/de/api

Table 4: Absolute and relative occurrence of samples per genre in the dataset

Pop	Rock	Rap	Electronic	Soul	Chanson	Reggae	Dance	Metal	Jazz	Disco	Classic	Country	Blues
2,004	1,633	1,397	970	784	543	520	467	197	159	142	96	62	55
22%	18%	15%	11%	9%	6%	6%	5%	2%	2%	2%	1%	1%	1%

as mean and standard deviation are derived from these features, resulting in 392 features overall. For the feature extraction, a representative sample of 30 seconds of each music video is used, which is a common procedure [15, 24, 22]. The sample is taken from second 30 to 60 for tracks with a duration greater than 60 seconds, to avoid the often not representative intro. In case the video length is 60 seconds or shorter, the first 30 seconds are used. For 37,732 videos, the features are derived this way. To reduce the number of features carrying similar information, sets of highly correlating features are determined. For each of this set, only the feature with the lowest entropy is kept as it carries most information. This excludes features that have for most tracks the same or equal values. Thereby, 317 out of 392 features remained.

4.3 Genre and Mood Classification

As the mood and genre information is not available for all songs, a mood and a genre classifier are developed. Classifiers need to be trained on a labeled dataset. Therefore, the low-level audio features derived by MIRtoolBox in conjunction with the determined genre or mood obtained from last.fm containing correctly annotated audio samples are used for classifier training and testing. In a first preprocessing step, all features are normalized to a number between 0 and 1, which is a common requirement for most machine learning algorithms. With the goal to avoid using less predictive and unnecessary many features, the classifier is trained on an iteratively increasing number of features, thereby following a common subset heuristic. If an audio feature can increase the classification accuracy, it is added to the feature set used, otherwise it is discarded. However, the classification accuracy did not increase significantly by doing so. Therefore, all of the 317 features are kept.

Based on the literature presented (ref. Sec. 2.1), a SVM is chosen as the classification model. Many combinations of parameters C(penalty parameter of the error term), γ(kernel coefficient), and different kernels (linear, radial, polynomial, and sigmoid) are tested to find the optimal SVM configuration. To achieve a robust measure of accuracy, a 10-fold cross-validation is performed. The python library scikit-learn [5] is used for training, cross-validation, and grid-search for hyper-parameter optimization as well as class balancing. It is important to consider that the number of samples per class in the dataset vary, i.e., the dataset is unbalanced. Therefore, classes are weighted inversely proportional to their occurrence in the dataset to balance their influence and, thereby achieve high classification accuracy. Following this procedure, a mood and a genre classifier are trained which can determine a track' s mood and genre based on its low-level audio features. It has to be noted that the dataset used in this paper is significantly larger than the ones stated in the related work. The achieved accuracy of the mood classifier is 64% by using a radial basis function (rbf) kernel and a test set size of 10%. As depicted in Fig. 5, the accuracy differs for each class. This figure shows a confusion matrix stating how many percent of the true labels are classified correctly. For example, the category angry is easily mistaken for happy, while happy and relaxed music can be identified with a high accuracy of about 70%.

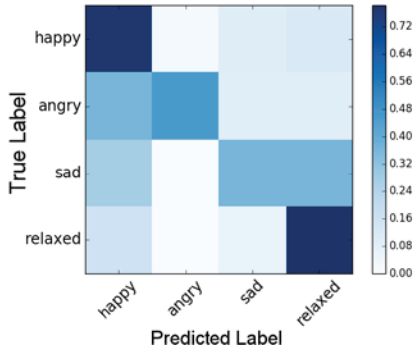


Figure 5: Confusion matrix of the mood classifier

Surprisingly, the rbf SVM used, similar to the approach of Rho et al.[19], outperforms the best reported results of Laurier et al. [15]. However, the reason might be that the training data available is larger than in the related work. Furthermore, the approach in [15] performed only 1.6 - 6.7% worse, depending on parameter configuration chosen for the SVM. Still, this is a high accuracy compared against a random classifier (25%) and a majority class classifier which would always predict happy (40%).

For the genre classification, on the one hand, hip hop, pop and electronic showed a high precision with up to 70%. On the other hand, blues and country, the two smallest categories in the training data, performed with 0% and are, therefore, usually misclassified. However, overall the genre classifier showed an average precision of 50%, which is high for that many classes, compared with a majority class classifier which would always predict pop (22%) or a random classifier(7%). The confusion matrix for all genres is depicted by Fig. 6.

5. PROACTIVE CACHE POLICY DESIGN

The proactive caching policies proposed determine which videos are cached into a dedicated share of the cache. In this storage area, videos are placed on a regular basis, e.g., once per day or 4 times per day. The rest of the cache space is managed by LRU it is the most popular caching policy and, therefore, allows high comparability with other works. For proactive caching, different recommendation approaches are used to determine suitable caching candidates. Two general types of recommenders can be distinguished: content-based and user group-based also known as collaborative filtering. While the content-based approaches use just information related to the content requested by one user, the user group-based approaches require a detailed history of many users to work properly. Therefore, user behavior-based recom-

[5]http://scikit-learn.org

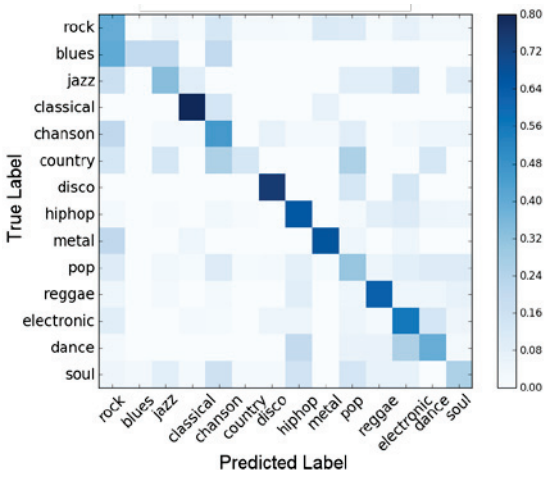


Figure 6: Confusion matrix of the genre classifier

menders require much more resources and detailed information of users. Hence, they are in contrast to content-based approaches not privacy-preserving.

Independent of the approach used, it outputs an ordered list of music video IDs, which are likely to be efficient candidates for proactive caching. This list is taken as an input for the evaluation simulation, where the cache size and the share of the cache used for proactive caching can be configured. Depending on the size of the proactively managed cache share, the top entries of the list recommended by the policy used are taken and the respective videos placed within the proactively managed part of the cache.

5.1 Content-based Caching Policies

Content-based approaches use only information from the contents requested as an input. The underlying assumption is that if a user or a group of users have requested a video, it is likely to request similar videos.

5.1.1 Popularity

A naive benchmark for proactive caching is to select the most popular videos from the recent past. This approach can be implemented easily and is likely to result in a good performance for the near future. However, depending on the dynamics of new videos added to the content catalog this information can soon become stale. In addition, the recent past used to derive the popular candidates might not be representative for the near future, e.g., the watching behavior correlates with a certain time or hour of the day.

5.1.2 Time-aware Caching

The drawback of the ***popularity*** policy, i.e., to consider only potentially stale information from the recent past, is avoided by leveraging seasonal patterns. To this end, time-aware caching is proposed. Music taste is observed to shift over the hours of the day, e.g., activating music during sport activities and relaxing music in the evening [4]. Hence, a caching approach that gives a higher priority to music videos that match the most popular genre or mood of the current hour of the day is likely to increase the cache's performance. In order to investigate this hypothesis, the composition of genre and mood categories is analyzed for each hour of the day. The results are shown in Fig. 7. Surprisingly, only small variations can be observed for different hours of the day. This is observed for all 14 days captured in the trace. While most categories' popularity is relatively static, hiphop and pop music are showing an interesting behavior. As both categories vary in popularity, their summed share stays stable around 60%. One explanation for this might be that the same users tend to request content from both categories.

Genre and Mood.

The time-aware caching policy proposed uses for a dedicated feature, e.g., genre or mood its mean popularity for each hour of the day. Next, for all requests within the current hour of the day the dominating genre or mood is determined, i.e., from which most videos have been watched. The recommendation includes only items belonging to the respective dominating genre or mood. Within this list, the items are ordered by their global popularity. As an extension, not only one but many categories can be considered, e.g., two categories where the space for pop and hiphop music is divided proportionally to these genres' popularity share for the current hour of the day. The policies described above are referenced as ***genre*** and ***mood*** in the following.

5.1.3 Audio Features

The following two policies are only applicable to the scenario where the cache is installed on the user's premise, e.g., on a smartphone. They do not require information from other users but a list of recently watched videos for the user for which the proactive caching is performed.

Feature Vector.

For each user, music videos that are watched more than once by the user are selected. Then, the cosine similarity of the feature vector is calculated between the audio of the videos watched by a user more than once and all previous videos watched by other users. Afterwards, the list containing videos watched by other users is sorted according to the computed similarity. Number of features is variable. However, in this paper, 100 features are used. Thereby, only the features which do not correlate w.r.t. the Spearman rank-order correlation are selected.

Feature.

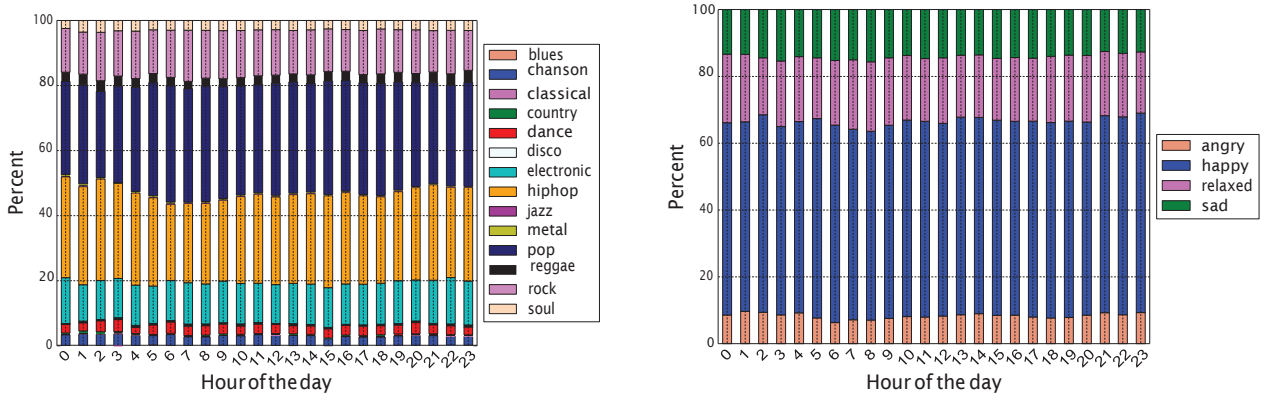
This policy works similar as the ***feature vector*** policy, except that just one low-level feature is used instead of all. Thereby, the average value of this feature for all previously watched videos is used to order the music video list. This allows to evaluate which of the features is most relevant and not to use less relevant features.

5.2 User Behavior-based Caching Policies

The user behavior-based approaches require a user-item matrix that contains information about all requested items from all users. Here, the idea is that users or group of users which are similar with respect to their requesting behavior are also likely to request videos one of them has requested but the other has not yet requested.

5.2.1 Feature Range

This recommender is a combination of user-based collaborative filtering and content filtering. User similarities are defined by their average value for a certain low-level audio feature and their range of its variance using previously



(a) Genre (b) Mood

Figure 7: Genre and mood popularity per hour of the day

watched videos. The average reflects the suitability of this feature for the users while the variance reflects how diverse their music taste is w.r.t. this feature.

5.2.2 Matrix Factorization

It has been shown that for recommendation tasks collaborative filtering, i.e., implemented by ***matrix factorization*** has a higher accuracy than nearest-neighbor approaches[13]. In this work, the ALS[8] (Alternating Least Squares) implementation of Apache Spark[6] is used.

The input data consists of triples (userID, videoID, rating) where the rating can be chosen by two variants. First, implicit rating is used which represents how often a user watches a video. Second, explicit rating is used which refers to the percentage of the video watched. Thereby, jumping over uninteresting videos can be distinguished from watching a video fully. Additionally, the explicit rating is weighted by the number of times a user watches the video, e.g., if it is watched 50% twice, the explicit rating is $2 \times 0.5 = 1$. This is expected to significantly increase the policy' s performance as most videos on YouTube are not watched fully [12].

5.2.3 CPSys

CPSys [5] is a mobile video prefetching system. It uses caches residing on mobile user devices. Videos for proactive caching are selected based on collaborative filtering, i.e., finding closest neighbors of a user and suggest videos for prefetching which are consumed by his neighbors previously but not yet by the user. Thereby, preference is given to most recent and most popular content. CPSys is evaluated on a YouTube video request trace, but explicitly excludes music videos as they show a different popularity pattern compared with other videos' categories. Furthermore, music videos are more likely being consumed repeatedly, while other types of content, for example news and sport events are watched mostly once. It is expected that CPSys does not perform well when contents of different categories are used together. However, as this work is dedicated to music videos only, since the majority of mobile video requests belongs to this category (ref. Sec. 3), in this paper, CPSys is implemented and evaluated.

5.2.4 User Similarity

[6]http://spark.apache.org/

Similarity Measure.

This policy uses a user-item-matrix containing an entry for every video a user has watched. If a video was watched more than once, the number of watches is entered into the matrix. Using this matrix, for each user, similar users are computed, which are called neighbors in the following. The similarity between a user and its neighbors is defined by the Jaccard similarity coefficient. Thereby, neighbors that have watched many of the videos also the user has watched are selected. In a further step, the videos that have been watched by the neighbors but not yet by the user are determined. Their watch count, i.e., the matrix value, is weighted by the similarity between the user and the neighbor that has watched this video. Finally, this value is taken as a score and the videos with the highest score are selected for being proactively cached.

Modified Similarity Measure & Feature.

The similarity policy does not work for all users, as it requires at least one other user who has watched the same video as the user for which proactive caching is applied. For a few users this leads to no results, as no neighbors exist. This policy compensates for this by applying recommendation by ***feature*** policy if no neighbors can be determined by the ***similarity measure*** policy.

Similarity Measure & Feature.

This policy is a hybrid policy between ***similarity measure*** and ***feature***; hence the ***feature*** policy does not serve as a backup like in the previous policy but is always used. First, both policies output list are computed separately. Next, the union of both lists is build and the values are summed up in case a video occurs in both lists. In case the ***similarity measure*** cannot determine a neighbor, this policy results in the same videos being recommended as the ***feature*** policy does.

Aggregated Similarity.

In order to apply the ***similarity measure*** policy for in-network caches, the recommendations for each user is considered as a tuple of (videoID, score). For all users, these

lists are concatenated and in case an entry occurs multiple times in the resulting list, their scores are summed. The resulting list can be perceived as an aggregated similarity measure over all users. Thereby, this metric can be used not just for one user, but also for a group of users in contrast to the other similarity measure policies proposed before.

6. EVALUATION

This section evaluates the proposed proactive caching policies. Thereby, the size of the whole cache and the share of the cache that is managed proactively are varied. The remaining part of the cache, i.e., the reactively managed share, is always managed by LRU as it provides a well-known benchmark often used in the related work and enables a direct and quantitative comparison to the related work. Each policy evaluated results in a list of videos from which the top items are chosen to fill the share of the cache which is manages proactively. By default, this is done once per day, which is assumed reasonable for a typical provider network.

User Selection.

For the evaluation, only users with a constantly high demand for music videos are selected. These users can be easily determined, e.g., by a mobile network operator or content provider. Therefore, in the following, only users having requested at least two videos per day for seven days within the two weeks trace are considered. Thereby, 5,351 users, constituting 1,64% of all users within the trace are used which are responsible for 15.6% of the total video requests. On average, each of these users watched seven videos per day.

6.1 In-network Cache Evaluation

This section evaluates caches covering many users, e.g., within the ISP, CDN, or content provider. In order to show the influence of the cache size, it is varied from 100 to 1,500 videos. Additionally, the share of these caches managed by the proactive caching policy is varied between 5% and 25% to investigate the influence of the ratio between proactive and reactive caching.

In the following, the evaluation of the policies: ***popularity***, ***genre***, and ***aggregated similarity*** are described, as they are three of the most different polices. In case all 700k users are served by the same cache, each of the policies performs comparably well. Therefore, only the cache hit rate (CHR) of the ***popularity*** policy is chosen to be further evaluated. It requires the lowest computational complexity and, therefore, is the most reasonable choice. Fig. 9 shows the resulting CHR if the ***popularity*** policy is used.

Furthermore, the cache size and the proactively managed share of the cache are varied. The blue line represents 0% space managed proactively, i.e., a pure LRU cache. As depicted in the figure, the greater the share of proactively managed cache share, the greater is the resulting CHR. Increasing the proactive share in 5% steps shows a small positive effect after each step, converging at a CHR of 28% and a cache size of 1,500 video items.

In a further setup, five distributed caches are investigated, e.g., placed in the metropolitan areas of a country. Thereby, $\frac{1}{5} = 140k$ of the users are randomly assigned to one of the five caches. For each of the three policies, the CHR differences compared to pure LRU are simulated to investigate the effect of proactive caching using the average CHR of all five caches. The results are depicted in Fig. 8. An interesting finding is that independent of the policy, proactive caching can notably increase the CHR for small (100-400 videos) and for large cache sizes (1,300-1,500 videos) but less for sizes in between. Especially for small cache sizes, proactive caching can increase the CHR by up to 4% using the ***popularity*** policy. For cache sizes between 600 and 1,100, this policy has a positive effect of up to 0.8% if smaller proactive cache sizes are chosen. The policies ***genre*** and ***aggregated similarity*** show a comparable performance for small cache sizes and genre achieves slightly better results for small cache sizes. For mid-sized caches, all policies are also able to decrease the CHR when the proactively managed share of the cache is chosen too large, e.g., 25%. However, a proactive cache share of 5% always increases the performance, even slightly for mid-sized caches. Comparing the three policies, it can be seen that the ***popularity*** policy is superior to genre and ***aggregated similarity***. Overall, the maximum CHR with 55.1% is achieved by the ***popularity*** policy with a proactive cache share of 20% and a cache size of 1,500 items. Proactive caching by popularity achieves the highest gain measured by CHR with a cache size of 200 and a proactive cache share of 25%. Traditional LRU achieves just 8.9% CHR and, combined with proactive caching, achieves 12.8% CHR. Summarizing, the performance of proactive caching depends on the cache size, the number of users served by the cache, and the proactively managed cache share. Furthermore, the number of videos in the content catalog is likely to influence the results. However, as a real trace is used, this parameter is not further evaluated.

6.2 User Cache Evaluation

In the following simulations, a fixed number of cache entries per user is assumed. This number is defined by the average number of videos watched by a user over the two weeks period captured by the trace. However, the number of entries is limited by a maximum of ten to respect the limited client storage capacities. Assuming the videos are cached in 720p resolution, ten videos require 1.3 GB memory or 0.7GB for a resolution of 480p. Botch calculations assume an average video length of 3.5 minutes and a bitrate of 5 and 2.5 Mbps, respectively. For caching on mobile devices, such as smartphones or tablets, a series of simulations using different proactive caching policies are performed. As a common evaluation metric for the different policies, the F1 score also known as F-measure is chosen, as it considers both precision and recall. Precision is defined as the number of videos placed in the cache that are later watched by the users. Recall defines the share of videos that are watched and previously placed in the cache and, therefore correlates with the CHR. For recommender systems, like the policies recommending videos for proactive caching are, this is a common metric.

Policy Comparison

Fig. 10 depicts the F1 score of the proposed policies and their 95% confidence intervals. Policies using the information of all users tend do achieve a higher F1 score, except the ***feature range*** policy which scores insignificantly higher than the ***popularity*** policy. The policies feature vector, popularity, and feature range archived the lowest performance with a value smaller or equal than 0.019. Mood and genre are both significantly superior compared with popularity, which is the default benchmark when it comes to recommendation

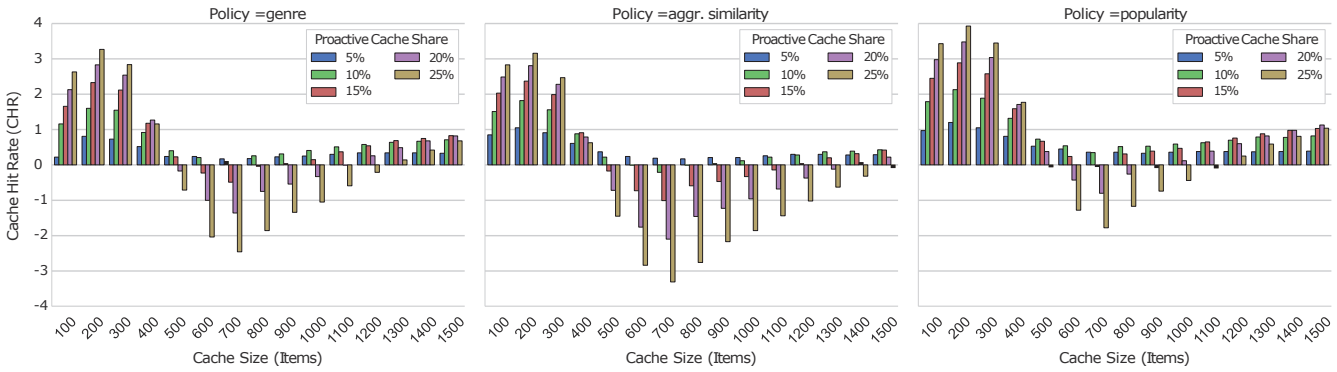


Figure 8: Relative cache performance compared with proactive caching being deactivated

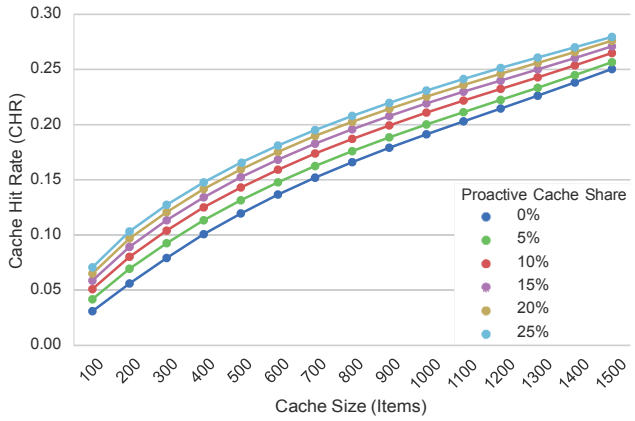


Figure 9: Cache performance with different sizes of caches and proactively managed cache shares

systems. Even though the ***genre*** policy uses a genre classifier with many genre classes and, therefore, a relatively low per-class accuracy compared with the ***mood*** policy's classifier, the ***genre*** policy archives a significantly higher F1 score. As the ***genre*** policy is able to work with either one or many genres, different numbers of genres are evaluated but do not have an impact on the policy's performance.

For the ***matrix factorization*** policies, both proposed variants are evaluated using a training dataset of seven days for optimal performance. First, vanilla ***matrix factorization*** which uses the number of video watches as the video's rating is simulated. Second, ***matrix factorization*** with explicit rating, i.e., how many percent of a video have been watched is evaluated. The second variant scores significantly higher than the first variant but shows a larger confidence interval, however not overlapping with the first variant's confidence interval. CPSys shows better results than ***matrix factorization*** but with a large confidence interval. The best performance in the category of privacy-preserving policies is achieved by the ***feature*** policy with an F1 score of 0.1. For this policy, the feature ***spectrum mean*** is used, as it shows to achieve the highest F1 score. Surprisingly, thereby, it archives a better performance than the previously described policies. Overall, the highest F1 score is achieved by the ***similarity measure*** policy, with a value of 0.197. The number of neighbors is set to 20 for all similarity measures as well as for CPSys to guarantee comparability. Further hybrid policies which try to enhance the ***user similarity*** performance by combining it with the best content-based policy, i.e., ***feature***, cannot increase the performance of the pure ***similarity measure*** policy.

So far, in one simulation a single policy is applied for all users. However, choosing a policy on a per-user basis may further increase the performance. To test this hypothesis, for each user and day captured by the trace, it is evaluated which of the two policies: ***feature*** or ***similarity*** is superior. Surprisingly, for 21.5% of the users, ***feature*** is superior to ***similarity*** for at most one day. For 6.8% of the users the policy is superior for two days and for 2.7% of the users, ***feature*** is superior for three days. Finally for ≤0.5%, which are less than 10 users, ***feature*** is superior over ***similarity*** for 7 days and more. The number of users and days where ***feature*** results in a better performance than ***similarity*** are quite limited. Therefore, the performance gain of a per-user policy selection is assumed quite limited as well and is not further investigated.

Summarizing, the feature policy achieves a high F1 score of 0.1 and is thereby even superior to ***matrix factorization***, the state-of-the-art recommender approach used, e.g., by Apache Spark. However, at the cost of privacy-loss, similarity measure policies achieve F1 scores twice as high as the privacy-preserving ***feature*** policy. Yet, due to legal restrictions, they may never be applied in reality in certain countries. The ***feature*** policy, in contrast to this, leverages only public information of the content and the user that uses this policy. As a result, the policy can be implemented at the user device, e.g., as part of a locally operating Android or iOS app. Thereby, the privacy-sensitive data never has to leave the user's device.

Time Windows

For the results presented, proactive caching is performed once a day, which is assumed reasonable for a typical provider network. In the following, the frequency of proactive caching as well as the size of data used as an input for a policy is further investigated. Therefore, different time windows are evaluated. At the beginning of each time window, the cache is filled proactively and is not changed till the next time window starts. The data used for recommendations by proactive policies is important to gain a good performance. A time window contains the user requests between start and end of the time window. It has two properties, the starting hour,

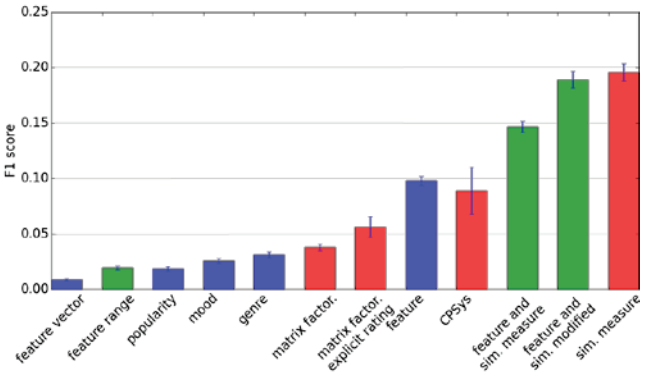


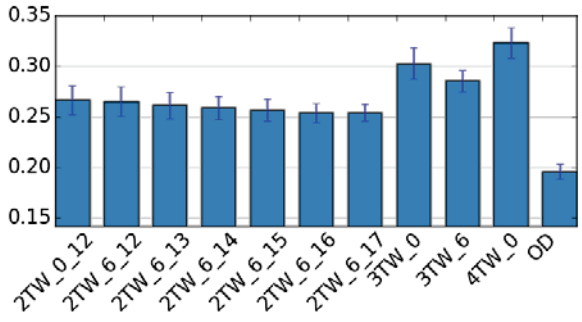
Figure 10: F1 score of policies, blue: content-based (privacy preserving), red: user group-based, green: hybrid

i.e., midnight or noon and the length, i.e., 12, 6, 8 hours, defining how many time windows exist per day, i.e., 1, 2, 3, or 4. In the following, a configuration of two time windows starting at midnight and noon are denoted as 2TW_0_12. The default approach of recommending once a day is denoted as 0D. If more than two time windows are used, e.g., 3_TW_6, the last number represents the starting time of a time window while its length is the remaining hours of the day divided by the number of time windows, e.g., from 6am till midnight (18 hours) results in three six-hour-long time windows in the previous example. As shown in Fig. 3 there are a few requests in the time between 1-6am. Therefore, additionally to midnight, time windows can also start at 6am. While for policies like genre and other content-based policies the time window does not significantly affect the performance, it does so for the ***user similarity*** policy. As this policy archives the best performance, its time window analysis is described in the following. The results are depicted by Fig. 11.

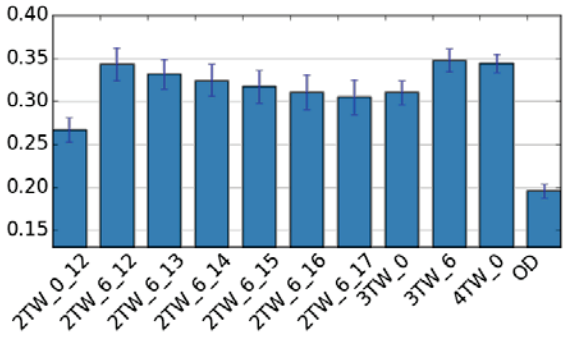
Additionally to number and beginning of time windows, in (a) all time windows' data for the current day is used, starting at midnight, while in (b) the results for just using the last passed time window is shown. It is important to notice that using more data from more time windows does not lead to a higher performance in general. Overall, using the last time window's data only leads overall to higher performance values. Configuration 3_TW_6 achieves the highest mean of F1 measures with 0.35, thereby being significantly better than filling the proactive cache once a day (0D), as well as configurations with two and three time windows starting at midnight. As the configurations starting at 6am achieve much higher F1 scores than the same configurations starting at midnight, it is demonstrated that using the video requests from the current day and filling the cache at 6pm in the morning achieves better performance.

7. CONCLUSION AND FUTURE WORK

This work presents proactive caching policies based on a novel set of content features, namely music-specific features. A methodology for feature extraction using a request trace is proposed and classification approaches for genre and mood as promising new content features are compared. The proposed caching policies are evaluated using a real network trace. For the in-network cache scenario, different configurations of cache size and its share used for proactive caching



(a) all of the day's passed time windows



(b) last time window only

Figure 11: F1 score of different time window configurations for the ***user similarity*** policy with 95% confidence intervals

are evaluated. For the scenario applying caching on the user devices, twelve policies are proposed and evaluated as well as relevant aspects, e.g., how much they respect the user's need for privacy. Additionally, the frequency and the amount of data used for proactive caching is investigated. The key findings of this work are: First, proactive caching is beneficial for large caches with many users being served. However, also caches smaller than 500 can benefit from proactive caching. For cache sizes between 500 - 1,000, proactive caching has no positive effect. Second, for proactive caching on mobile user devices, the privacy-preserving ***feature*** policy achieves a performance more than twice as high as ***matrix factorization***, a state-of-the-art approach used, e.g., by Apache Spark. Overall, the ***similarity measure*** policy achieves the highest performance with an F1 score of 0.2. However, this performance comes at the cost of user privacy, as the requests of all users have to be known. Third, the frequency and the amount of data used as an input for proactive caching policies are analyzed. Proactive caching applied three times a day, every six hours starting from 6am further increases the cache performance with an F1 measure of 0.35 compared with 0.2 if performed only once a day. Summarizing it can be said that proactive caching using music features is a valuable method to work together with reactive caching policies to increase the overall cache performance.

In future work, the effect of different video qualities on proactive caching algorithms is planned to be evaluated, as well as considering video segments instead of whole videos. Additionally, larger cache sizes and different content catalog configurations will be considered. Furthermore, policies using deep learning and audio features are planned to be developed and evaluated.

Acknowledgment

This work has been funded in parts by the German Science Foundation (DFG) as part of the Collaborative Research Center 1053 MAKI.

8. REFERENCES

[1] Ericsson Mobility Report, Feb 2016.

[2] M. Claeys, N. Bouten, D. D. Vleeschauwer, W. V. Leekwijck, S. Latré, and F. D. Turck. An Announcement-based Caching Approach for Video-on-Demand Streaming. In *Proceedings of the 11th ACM International Conference on Network and Service Management (CNSM)*, pages 310 – 317, 2015.

[3] T. Eerola, O. Lartillot, and P. Toiviainen. Prediction of Multidimensional Emotional Ratings in Music from Audio Using Multivariate Regression Models. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[4] M. Gillhofer and M. Schedl. Iron Maiden while Jogging, Debussy for Dinner? In *Proceedings of the MultiMedia Modeling (MMM)*, 2015.

[5] A. Gouta, D. Hausheer, A. Kermarrec, C. Koch, Y. Lelouedec, and J. Rückert. CPSys: A System for Mobile Video Prefetching. In *Proceedings of the 23rd IEEE International Symposium on Modelling Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2015.

[6] B.-j. Han, S. Rho, S. Jun, and E. Hwang. Music Emotion Classification and Context-based Music Recommendation. *Multimedia Tools and Applications*, 47(3):433 – 460, 2010.

[7] G. Hasslinger, K. Ntougias, and F. Hasslinger. A New Class of Web Caching Strategies for Content Delivery. In *Proceedings of the 16th IEEE International Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 1 – 7, 2014.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263 – 272, 2008.

[9] Y. Huang, S. Lin, H. Wu, and Y. Li. Music Genre Classification based on Local Feature Selection using a Self-adaptive Harmony Search Algorithm. *Data & Knowledge Engineering*, 92:60 – 76, 2014.

[10] Ipsos Connect. Music Consumer Insight Report, 2016.

[11] H.-G. Kim, N. Moreau, and T. Sikora. *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*. John Wiley & Sons, 2005.

[12] C. Koch and D. Hausheer. Optimizing Mobile Prefetching by Leveraging Usage Patterns and Social Information. In *Proceedings of the 22nd IEEE International Conference on Network Protocols (ICNP)*, pages 293 – 295, 2014.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, (8):30 – 37, 2009.

[14] O. Lartillot, P. Toiviainen, and T. Eerola. *A Matlab Toolbox for Music Information Retrieval*, pages 261 – 268. Springer, 2008.

[15] C. Laurier, O. Meyers, J. Serrà, M. Blech, P. Herrera, and X. Serra. Indexing Music by Mood: Design and Integration of an Automatic Content-based Annotator. *Multimedia Tools and Applications*, 48(1):161 – 184, 2010.

[16] J. Li et al. YouTube Traffic Content Analysis in the Perspective of Clip Category and Duration. In *Proceedings of the 4th IEEE International Conference on the Network of the Future (NOF)*, 2013.

[17] B. Logan. Mel Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[18] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. jAudio: An Feature Extraction Library. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 600 – 603, 2005.

[19] S. Rho, B.-j. Han, and E. Hwang. SVR-based Music Mood Classification and Context-based Music Recommendation. In *Proceedings of the 17th ACM International Conference on Multimedia (MM)*, 2009.

[20] M. Schedl, G. Breitschopf, and B. Ionescu. Mobile Music Genius: Reggae at the Beach, Metal on a Friday Night? In *Proceedings of ACM International Conference on Multimedia Retrieval*, 2014.

[21] A. Solheim. Microwave Backhaul Radios Meet The Evolving Traffic Challenge. Mobile Dev & Design, Feb 2013.

[22] Y. Song, S. Dixon, and M. Pearce. Evaluation of Musical Features for Emotion Classification. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[23] R. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989.

[24] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas. Multi-Label Classification of Music into Emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[25] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293 – 302, Jul 2002.

[26] S. Wilk, J. Rückert, T. Thräm, C. Koch, W. Effelsberg, and D. Hausheer. The Potential of Social-aware Multimedia Prefetching on Mobile Devices. In *Proceedings of the International Conference on Networked Systems (NetSys)*, pages 1 – 5, March 2015.

[27] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen. A Regression Approach to Music Emotion Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):448 – 457, 2008.

[28] Y.-H. Yang, C.-C. Liu, and H. H. Chen. Music Emotion Classification: a Fuzzy Approach. In *Proceedings of the 14th ACM International Conference on Multimedia (MM)*, 2006.

[29] Y. Zhao, N. Do, S.-T. Wang, C.-H. Hsu, and N. Venkatasubramanian. O2SM: Enabling Efficient Offline Access to Online Social Media and Social Networks. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 445 – 465. Springer, 2013.