The VLDB Journal (2019) 28:197–219
https://doi.org/10.1007/s00778-018-0524-7

REGULAR PAPER

Real-time context-aware social media recommendation

Xiangmin Zhou[1] · Dong Qin[1] · Lei Chen[2] · Yanchun Zhang[3,4]

Received: 24 February 2018 / Revised: 30 August 2018 / Accepted: 10 October 2018 / Published online: 26 October 2018

Abstract
Social media recommendation has attracted great attention due to its wide applications in online advertisement and entertainment, etc. Since contexts highly affect social user preferences, great effort has been put into context-aware recommendation in recent years. However, existing techniques cannot capture the optimal context information that is most discriminative and compact from a large number of available features flexibly for effective and efficient context-aware social recommendation. To address this issue, we propose a generic framework for context-aware recommendation in shared communities, which exploits the characteristics of media content and contexts. Specifically, we first propose a novel approach based on the correlation between a feature and a group of other ones for selecting the optimal features used in recommendation, which fully removes the redundancy. Then, we propose a graph-based model called *content–context interaction graph*, by analysing the metadata content and social contexts, and the interaction between attributes. Finally, we design hash-based index over Apache Storm for organizing and searching the media database in real time. Extensive experiments have been conducted over large real media collections to prove the high effectiveness and efficiency of our proposed framework.

Keywords Social media recommendation · Feature selection · Content–context interaction · Real-time

1 Introduction

The creation of online shared communities has resulted in the astonishing increase in digital media data, and their applications in many domains like entertainment and advertisement. A shared community is a media sharing service, such as YouTube and Netflix, in which users can upload, download, comment, view, and rate the media. In shared communities, user behaviours are highly affected by system recommendations, which has raised the demand of advanced social recommendation. Social media has contexts such as time and location, which can enhance the recommendation quality. Users from various areas may have different preferences to media or advertised products. Users' interest may be affected by their viewing history, and change over time. The media popularity may change over time, locations, and social user communities. Consider a social media recommendation application in advertisement. A media on the Great Ocean Road may have greater effect on users in Melbourne than those in Sydney. A commercial on surfing in Gold Coast may attract more interests in Summer. A concert clip may be popular in a beginning period and become less popular after years. Some users share similar preferences on media, while do not have common interests with others. In practice, applications are typically context sensitive, which demands the development of context-aware recommender systems.

We study effective solutions for real-time context-aware media recommendation in shared communities. For online recommender systems, three key issues need to be addressed. First, we need to construct a robust and extendible model that captures the media content, contexts, and their interactions,

✉ Xiangmin Zhou
xiangmin.zhou@rmit.edu.au

Dong Qin
dong.qin@rmit.edu.au

Lei Chen
leichen@cse.ust.hk

Yanchun Zhang
yanchun.zhang@vu.edu.au

1 School of Science, RMIT University, Melbourne, Australia

2 Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hongkong, China

3 Centre for Applied Informatics, Victoria University, Footscray, Australia

4 Cyberspace Institute of Advanced Technology (CIAT), Guangzhou University, Guangzhou, China

and can be extended to various contexts. This is vital, as media data contain much uncertain textual content and social contexts that greatly affect users' interests, while interactions exist between content and contexts, and among various types of contexts. Failing to capture these characteristics may lead to a low recommendation quality. Take the time context as an example. Ignoring it may mislead online systems to recommend an item that only meets a user's interest years ago, which does not reflect her current interest and her interest changes over time. Meanwhile, without extendibility, a recommender system will have to be redeveloped when it is applied to other online communities. Second, we need to design novel solutions for selecting the best features used in recommendation. As many contexts are associated with media, some of them are actually redundant. For instance, we can infer that a social media on Federer vs. Nadal in Australian Open 2017 FINAL belongs to sport category. In this case, the type context is obviously redundant. A good feature selection solution will guarantee the quality of recommendation and greatly ascertain the efficiency of system. Finally, we need to design efficient indexes for organizing and searching the media database over distributed environment. According to the statistics of Tubular Intelligence, 434.4 million videos were uploaded to YouTube in 2015, generating 1.1 trillion views [1]. Sequential database scan over a single processor is obviously infeasible for the real-time recommendation.

The previous studies have various definitions of contexts catered for individual applications and domains, like behavioural categories [2], article popularity [4], social trust [5], user activities, and interactions [6–9]. However, none of them can systematically analyse what the compact and discriminative context set is and how to select it from features. While interactions exist among features, existing models lack mechanisms to adaptively capture this information and scale to different contexts. To overcome these problems, we propose a generic framework for context-aware recommendation with respect to personalized users by exploring media contexts such as location and time, together with content like title and description. We first design two novel algorithms to select the optimal feature set that is both compact and discriminative based on the correlation of a feature and a group of other ones. Then, we build a novel model called *content–context interaction graph* (CCIG) for media with a node pair-based CCIG similarity by analysing the contents and contexts, and their interactions. CCIG advances the feature interaction graph (FIG) [10] in threefold: (1) FIG was customized only for the fixed image features, while CCIG is extendible for any media with any number of contexts; (2) the representation for each content or context in CCIG is compact and robust to the uncertainty of media attributes; (3) unlike FIG that describes the text information of an image as a set of words and treats them equally, CCIG's tensor model for metadata is based on fixed tokens with the metadata source importance, thus more practical for big word sets. Our contributions in this work are summarized as follows.

1. We propose a correlation-based feature selection that exploits the correlation between a feature and a set of other ones for finding the most effective content and contexts. We propose two algorithms, FCRR and GFCRR, to achieve the optimal feature set, while saving the time cost of feature selection process.
2. We propose a novel uniform CCIG model and a graph similarity over it. While CCIG captures the media content, contexts, and their interactions, the graph similarity adopts a pair-node comparison, thus extendible to any number and types of features.
3. We propose a novel multi-level index on CCIGs over Apache Storm. On the first level, multiple hash-based indexes are exploited to organize the CCIG nodes to various attributes. On the second level, a set of buckets are used to store the media *ids* to the corresponding hash keys of the indexes.
4. We propose a data partition method that balances the workload of each processor in recommendation. Algorithms are proposed to compute top K media under Apache Storm. We conduct extensive tests to prove the high performance of our framework.

The rest of this paper is organized as follows. Section 2 reviews the related work on feature selection and context-aware media recommendation. Section 3 formulates context-aware recommendation problem. Section 4 presents our new feature selection method. Section 5 presents our new data model and similarity matching optimization. We report our experimental results in Sect. 6 and conclude our paper in Sect. 7.

2 Related work

We review existing literature on two topics closely related to our work, including feature selection and social media recommendation.

2.1 Feature selection

Feature selection aims to identify a subset of relevant features and remove redundancy in model construction. Normally, it includes subset selection and evaluation. Based on the subset evaluation, existing approaches for feature selection can be classified into two types: wrapper methods [11–13] and filter methods [14–19]. In [11], the subset is identified by a stochastic search algorithm, which randomly selects features from the whole feature set, and evaluated based on a large-scale linear regression model. Broadhursta et al. [12]

found the optimal subset based on the root-mean-square prediction error in a multiple linear regression model with a variable parameter and a fixed one, respectively. Based on them, two algorithms are proposed by replacing the regression model with a partial least squares model. Kabir et al. [13] proposed a neural network-based method called CAFS. First, it divides the input features into two groups based on their correlation. It then uses a constructive approach which starts with a small feature subset and incrementally adds more features to the hidden neurons. As such, it selects features from two groups and decides the neural network architecture. All wrapper methods are based on customized models, producing the feature subsets with higher accuracy. However, they may cause overfitting and high time cost on training the models for updated datasets.

Traditional filter methods [14,15] select subsets by feature weighting, which assigns feature weights based on the linear correlation of features and target concepts and ranks them based on their relevance to a target. Though they avoid the time for dynamic updates, the redundancy to a single concept can not be removed. Advanced filter methods adopt nonlinear correlation between features for subset selection. Yu et al. [18] used information entropy to analyse the redundancy. Wang et al. [16] maximized the weight of selected feature subset and minimized its redundancy. In [19], a feature forming a Markov blanket in a subset is considered as redundancy and determined by the correlation of a feature and a target concept. Song et al. [17] proposed a clustering-based method, which constructs a graph over features based on their correlation, divides the graph into clusters, and selects the most representative one from each cluster to form a subset. These methods remove the redundancy of pairwise features, but cannot identify the redundancy of a feature group.

2.2 Context-aware media recommendation

Traditional media recommendation exploits contents for finding user interests. Examples include topic network based [20], multimodal based [21], VideoTopic based [22], and characteristics based [23]. However, as social user behaviours are affected by the contexts in communities, simply using media content is not enough for effective recommendation. To overcome this issue, many context-aware systems have been proposed. Akther et al. [7] used a user's activity on various social networks to build a centralized profile, considering the physical contexts like temperature and the situational contexts like location. In [2], a probabilistic matrix factorization fuses the user preference and interpersonal influence based on the historical user–item and user–user interaction for recommendation. Yang et al. [5] inferred the category-specific friend circles and the trust on social links based on user rating. The interested categories are recommended to various friend circles by computing the ranking scores for all items. In [6], a temporal context-aware mixture model (TCAM) models the topics related to users' interests and those to time context for user behaviour modelling. In [4], contexts like time are integrated with a social popularity-based model, which first divides the original rating matrix based on different contexts and further predicts the ratings of an item by matrix factorization. Zhou et al. [8] fused video content and users for social recommendation in shared communities. Huang et al. [3] exploited the matrix factorization results, the types, and time, for recommendation. These methods exploit the preset contexts, which produces the high-quality recommendation for certain media. However, they are not extendible to various contexts in recommendation. Rendle proposed factorization machines that use SVM and factorization to capture feature interactions [24]. This model only exploits the social interactions, while the effects of features are not considered. Debnath et al. [42] proposed a feature weighting in content-based recommender system (FWC), which treats various contexts as attributes and assigns a weight to each context based on its importance to users. However, it neglects the correlation among contexts. In this work, we focus on adaptive feature selection, extendible data modelling, and real-time big data processing for recommendation. The notations used in this paper are listed in Table 1.

Table 1 Notation

Notation	Meaning	Defined in (section)
u	A social user	3
$p(u)$	The profile of a social user	3
M	Content of a media	3
$\mathcal{C}$	Context set of a media	3
ω	Title weight	4.1.1
$\mathcal{D}$	Tensor	4.1.1
v	Media data	4.1.1
N	Gram length	4.1.1
T	Time length	4.1.2
f	A feature	4.2.2
F	A feature set	4.2.2
c	Comments of a media	5.1.1
G	Content–context interaction graph	5.1.1
γ	Node weight	5.1.2
λ	Clique weight	5.1.2

3 Problem formulation

This section provides a formal problem definition on context-aware social recommendation and describes our proposed approach briefly. Our solution is a personalized recommendation with respect to specific social users. Thus, it is important to define a *user profile*, which is a preliminary component of our system.

Definition 1 Given a social user u, its *user profile*, denoted as $p(u)$, is described by the content and contexts of its viewing history. Formally, a user profile is a set of pairs $\{(M, \mathcal{C})\}$, where M is the content of a media in its viewing history and $\mathcal{C}$ denotes the contexts including the location, time, and user interactions.

In our context-aware social media recommendation, the system input is a social user described by its profile, and its output is a list of media data with top relevance scores to this profile.

Definition 2 Given a social user u, a media relevance function r, our context-aware recommendation constructs a user profile $p(u)$ and detects a list of most relevant media data, S_v, such that for any media data $v_i \in S_v$ and $v_j \notin S_v$, the following condition holds:

$$r(p(u), v_i) \geq r(p(u), v_j)$$

Figure 1 shows our system framework that includes two parts: (1) media database processing; (2) media recommendation. In media database processing, given a media set of all its content and contexts, we select features by context analysis. Then, we model each data as a content–context interaction graph (CCIG) over the selected features, which reflects the information of features and the correlations between them. The dataset is divided into n subsets, and each is indexed and allocated to a bolt under Apache Storm. For recommendation, given a user, we construct her profile as a CCIG set by modelling each media in her viewing history as a CCIG. This profile is passed to all bolts, and each conducts a KNN-based recommendation, in which multiple features contribute to the final results interactively. Compared with the collaborative filtering-based methods, our KNN-based recommendation is more robust to the sparsity of social media. The results from all bolts are ranked to form the final recommendation.

4 Media data feature selection

We first present the candidate content and contexts and then select the media features over the candidates for recommendation.

4.1 Candidate social data features

We will describe the candidate features that exist in most media shared communities, including metadata, time, location, category, user connection, and their representations.

4.1.1 Tensor model over media metadata

The content of a social media can be described as a set of keywords from its title and description. Given a keyword set, various models, such as pLSI, LDA, and TF/IDF, can be applied for representation. However, while pLSI and LDA require costly learning process over training dataset, original TF/IDF model tends to produce big and sparse vector representations. To solve their problems, we propose tensor model over textual metadata. Tensor space model (TSM) has been successfully applied for text analysis [25,26]. However, as the metadata are from two sources, title and description, we need to consider their importance difference in our model, so the representation reflects the importance of two sources. Motivated by this, we propose weighted tensor model (WTM) that allows the metadata from two sources to contribute to the model differently.

Weighted tensor construction Given a set of keywords from the title and description of a social media, we divide them into a number of N-grams and decide their weights before tensor construction. Intuitively, a big N leads to a high tensor dimensionality, high time, and space costs, while a small N may fail to capture discriminative information. Thus, we select good string length N and weight ω for our tensor construction by conducting preliminary experiments on dataset D_s that will be used for effectiveness evaluation in Sect. 6.3. We give each N-gram from title a weight ω and each from description a weight $1 - \omega$. All weighted m N-grams form a corpus. Applying TF/IDF over these N-grams, we represent

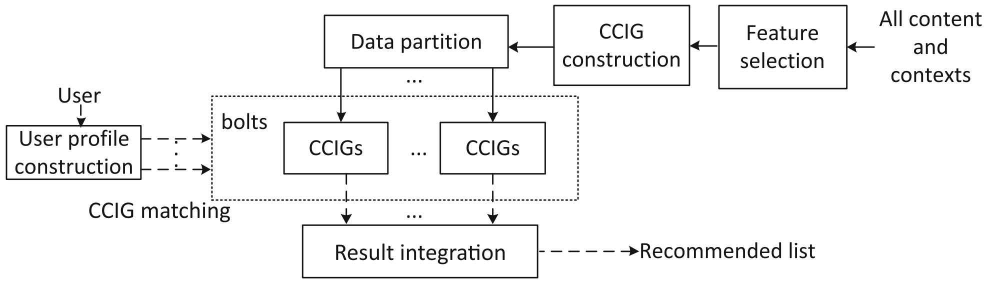


Fig. 1 Framework of our context-aware social recommendation

Table 2 Davies–Bouldin index to different N and ω

	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$
$\omega=0.1$	2.782	1.743	1.484	1.132	1.054
$\omega=0.2$	2.765	1.682	1.347	0.728	0.792
$\omega=0.3$	2.530	1.420	0.992	0.824	0.822
$\omega=0.4$	2.318	0.896	0.832	0.805	0.789
$\omega=0.5$	2.424	0.801	0.793	0.782	0.780
$\omega=0.6$	2.226	0.769	0.703	0.702	0.700
$\omega=0.7$	3.376	1.154	0.835	0.788	0.782
$\omega=0.8$	4.465	1.487	1.173	1.049	0.921
$\omega=0.9$	4.596	2.578	1.405	1.051	1.049
$\omega=1.0$	4.723	3.420	2.293	1.432	1.419

the metadata of each media as a m-dimensional vector. Given a number of vectors in D_s, we perform k-means clustering based on L_2 distance with respect to N and ω by varying N from 2 to 6 and ω from 0.1 to 1 for each N. We assess the quality of clustering by a standard metric Davies–Bouldin index [27]. As reported in Table 2, the DB index drops with ω increasing for a fixed N, reaches to an optimal value at $\omega = 0.6$, and increases with the further increasing ω. Thus, title takes more important role in recommendation. Meanwhile, with the increase in N, the optimal DB index drops and keeps steady after $N = 4$. We set 4 as default N to balance the tensor dimensionality and effectiveness.

With the optimal N and ω, we can construct weighted tensor model by embedding the weight in TSM [25,26]. Given the keywords of a media v, we apply the weight 0.6 to those in title and 0.4 to those in description. We index each 4-gram using 26 letters and "_" to all the non-letter characters. We construct 2-order weighted tensors where each dimension is to a character pair and has 729 (i.e. 27^2) pairs by assigning each character pair, "aa" ∼ "_ _", an index $0 \sim 728$, mapping each N-gram into one position of the matrix $\mathcal{D}$, and calculating its position value using weighted TF/IDF. An element is mapped into a value in a space $\mathcal{D} = \{a_{ij}\} \in R^{729\times 729}$. Given two subsets S_1 from title and S_2 from description, let $\langle r_1, l_1\rangle$ be a tensor position to $\hat{c}_1\hat{c}_2$ and $\hat{c}_3\hat{c}_4$, s denote string $\hat{c}_1\hat{c}_2\hat{c}_3\hat{c}_4$, and its weighted TF/IDF value is computed by Eqs. 1–3.

$$\mathrm{TF}(r_1, l_1) = \frac{\omega * \mathcal{F}(s, S_1) + (1-\omega) * \mathcal{F}(s, S_2)}{\omega * \bar{n}_1 + (1-\omega) * \bar{n}_2} \quad (1)$$

$$\mathrm{IDF}(r_1, l_1) = \log \bar{n}/(|v_i \in D : s \in v_i|) \quad (2)$$

$$T(r_1, l_1) = \mathrm{TF}(r_1, l_1) * \mathrm{IDF}(r_1, l_1) \quad (3)$$

where $\bar{n}_1$ and $\bar{n}_2$ are the number of N-grams in the title of v and that in its description, respectively, $\bar{n}$ the media number in training dataset, D the training dataset, and v_i a media whose metadata contains s. Using our WTM, the keyword set of each media is converted into a 729×729 matrix. We apply higher-order singular value decomposition (HOSVD), which has been effective for tag and user recommendation [28,29], to extract the main components of a tensor [26,30]. Based on our statistical analysis over D_s, we keep the first 50 singular values that take up to 90% information.

Tensor similarity measure We apply tensor similarity to our WTM for metadata analysis, considering its success in visual content analysis [31]. Suppose $\mathcal{D}_1$ and $\mathcal{D}_2$ are two weighted tensors constructed from two social media data, the similarity between them can be measured by their J-divergence. Let $p(x)$ and $q(x)$ be the N-gram distributions of two tensors, their J-divergence is:

$$J(p||q) = \frac{1}{2} * \left(\int p(x) \log \frac{p(x)}{m(x)} \mathrm{d}x + \int q(x) \log \frac{q(x)}{m(x)} \mathrm{d}x \right) \quad (4)$$

where $m(x) = (p(x) + q(x))/2$.

4.1.2 Context candidates

This part presents social contexts in details, including their models and similarity measures.

Time Social media may become popular in different time periods, which may affect the user interests and behaviour. Thus, we take the time of media data as a candidate feature, which is the period between the media upload and the last comment. We capture the period in which a social media received most interests, and describe this period T as its start and end time, i.e. $\langle t_s, t_e\rangle$. We set the length of T a fixed value for all media data and will evaluate it as a parameter in Sect. 6.3. Given two media time periods, T_1 and T_2, the similarity between them is measured by the probability of the overlapping between their most popular periods, which is defined as: $\mathrm{Sim}_T = |T_1 \cap T_2|/|T_1 \cup T_2|$. Based on Sim_T, we derive the distance between two periods:

$$\mathrm{Dis}_T = (1 - \mathrm{Sim}_T)/(1 + \mathrm{Sim}_T). \quad (5)$$

Location The location of a media indicates a space point where it is captured. Intuitively, users may have special interests to media from certain places. Thus, we consider location as a feature candidate and model it as a pair of its latitude and longitude $L : \langle la, lo\rangle$. Given locations $L_1 : \langle la_1, lo_1\rangle$ and $L_2 : \langle la_2, lo_2\rangle$, we measure their great-circle distance [32] on the surface of a spherical Earth, which is computed by: $\mathrm{GD} = R \cdot arccos(sinla_1 \cdot sinla_2 + cosla_1 \cdot cosla_2 \cdot cos(lo_1 - lo_2))$, where R is the Earth radius (6371 km). As GD is a physical distance, we normalize it by:

$$\mathrm{Dis}_L = (\mathrm{GD} - \mathrm{min}_{\mathrm{GD}}) \,/\, (\mathrm{max}_{\mathrm{GD}} - \mathrm{min}_{\mathrm{GD}}) \,. \quad (6)$$

Category Users may be interested in social media to certain categories. Category feature can be simply described as its name. The similarity of two categories is decided based on whether their names are same or not. Given two category features, if they have the same name, they are similar with the dissimilarity value 0. Otherwise, their dissimilarity value is 1.

User connection The user context of a media is described as an id set of its users, denoted as U, who have shared, liked, and commented it. Borrowing the idea in [8], we measure the similarity between the user connections of media data based on the probability of common users shared by their user id sets, which is: $\mathrm{Sim}_U = |U_1 \cap U_2|/|U_1 \cup U_2|$. Based on Sim_U, we derive the distance between two user sets as:

$$\mathrm{Dis}_U = (1 - \mathrm{Sim}_U)/(1 + \mathrm{Sim}_U) \tag{7}$$

We compute the user similarity over sub-communities to accelerate the similarity calculation as in [8]. Given a media collection, the social users are grouped into a number of sub-communities based on their interests on media in the whole dataset. A sub-community consists of a group of users who have highly similar preferences with respect to media. Turning the similarity over users into that over sub-communities, the complexity of user similarity computation becomes linear [8].

4.2 Correlation-based feature selection

We will first discretize each feature into categorical labels and then present our redundancy removal algorithms.

4.2.1 Feature discretization

In terms of the correlation of two random variables, there are two lines: classical linear correlation based and information theory based. We exploit the information theory-based measure since it captures the nonlinear correlations which exist in our application. However, due to the characteristics of social media features, the information theory-based measure cannot be directly applied. For one thing, social media are continuous data uploaded to shared communities at more than 300 items per minutes [33]. For another, the information theory-based measures like *joint entropy* only process discrete data. Thus, we will discretize features to transform each value into a series of categorical labels based on their representations. Since category features are naturally discrete strings, we only discrete other candidates.

To limit the label number and capture the common characteristics of metadata or context to a label, we conduct grouping operation over each feature. Given a set of 2-order tensors in the training set D_s, we cluster them based on their tensor distances. Tensors with high similarities are grouped together and described as a *metadata symbol label*. Given a set of time features in D_s dataset, we extract the middle point of a time as its representation and apply the *equal width binning* technology [34] to features. The bin intervals of a feature are taken as its *time categorical labels*. Given a set of locations in D_s dataset, we group the locations based on their great-circle distances and describe each group as *a location symbol*. We group the user connection features of the training dataset based on the distance D_u between user sets. Each group of user connections is labelled as a *user connection symbol*.

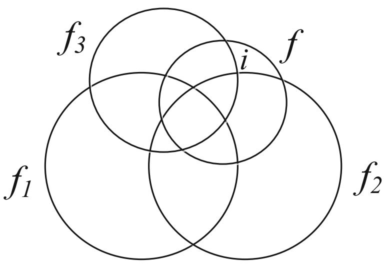


Fig. 2 Joint entropy of four features

4.2.2 Redundancy removal

The average pairwise correlation [35] and total amount of pairwise mutual information [36] for each feature have been exploited to alleviate the pairwise redundancy problem. In practice, one feature which is not redundant with respect to each single feature in a set is likely to be redundant with respect to a combination of several ones. As in Fig. 2, each circle is the information contribution of a feature in a set $\langle f, f_1, f_2, f_3 \rangle$. f is redundant with respect to the set $\langle f_1, f_2, f_3 \rangle$, but cannot be removed using existing methods. To solve this problem, we propose a new joint entropy-based concept *feature contribution* FC that shows the contribution of a feature to a set as follows:

$$\mathrm{FC}(F, f) = H(F, f) - H(F) \tag{8}$$

where F is a feature set. $H(F, f)$ $(f \notin F)$ is the *joint entropy* that measures the uncertainty of set $F \cup \{f\}$, and can be calculated by:

$$H(F, f) = H(F|f) + H(f). \tag{9}$$

Given features X and Y, $H(X)$ is the entropy of X, computed by:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{10}$$

where $p(x)$ is the probability that $X = x$. $H(X|Y)$ is the entropy of X under condition Y, computed by:

$$H(X|Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log_2 p(x|y) \tag{11}$$

where $p(x|y)$ is the probability that $X = x$ under the condition $Y = y$.

Given $F = \{f_i | i = 1, \ldots, |F|\}$ and a threshold τ, F has no redundant feature, if for $\forall f_i \in F$, there is no subset $F' \subset F$, such that $\text{FC}(F', f_i) < \tau$ holds. We set τ to 10% to avoid extreme information loss as in [19]. In [19], authors proposed a pairwise feature redundancy reduction approach, in which a set of tests have been conducted to obtain the optimal feature redundancy threshold. It has been proved that, for a given feature pair, one feature can be removed as redundancy safely if its entropy contribution to the pair is smaller than 10% of the overall entropy of the whole pair. In our work, τ is a redundancy threshold for a group of features, which is also defined based on entropy contribution. The optimal feature redundancy threshold in [19] is actually that for a special case of our feature redundancy solution, where the size of the set is 2. Thus, we can safely follow the setting in [19] and fix the optimal τ value to 10%.

Given a set F of all candidate features and τ, we aim to select an optimal subset F', such that $\forall F'' \subset F$, $F'' \neq F'$ and $|F''| = |F'|$, the following inequality holds: $H(F') \geq H(F'')$. Here, $|F'|$ the number of features in F' and $|F''|$ that in F''. Unfortunately, finding the optimal feature subset is an NP-hard problem.

Theorem 1 *The problem of Optimal Subset Selection is NP-complete.*

Proof We will prove that *Optimal Subset Selection* is an NP-complete problem via a two-step reduction from the *Subset Sum* problem.

Let S be a collection containing all the subsets of *Subset Sum* problem. In the first step, we construct $\mathbb{F}$, a collection containing all the subsets of *Optimal Subset Selection* problem based on S, in which $|S| = |\mathbb{F}|$ and there exists a one-to-one subset mapping χ between S and $\mathbb{F}$, i.e. $F_i = \chi(s_j)$, where $F_i \in \mathbb{F}$ and $s_j \in S$. Thus, the transformation from $sum(s_j)$ to $H(F_i)$ can be finished in $O(|S|)$, which is polynomial. In the second step, given a subset $F_i \in \mathbb{F}$, *Optimal Subset Selection* needs to compare $H(F_i)$ with that of all subsets with the size of $|F_i|$ to select the optimal one. This can be transferred from *Subset Sum* problem by selecting the largest sum among subsets of size $|F_i|$. Its time complexity is $O(|F_i|)$, which is polynomial. Thus, the problem of *Optimal Subset Selection* can be converted from *Subset Sum* problem by two steps with polynomial time cost. Thus, the *Optimal Subset Selection* is NP-complete. □

As the optimum subset selection cannot be achieved, we propose an algorithm called *feature contribution-based redundancy removal* (FCRR), which obtains the features by sequential backward selection globally in a greedy way. Given a set F of all candidate features, FCRR performs the operations over F by two steps: (1) compute the *feature contribution* of each $f_i \in F_r$ with respect to the remaining of the set $F_r \backslash \{f_i\}$; (2) remove the feature with the least contribution value smaller than τ. The two steps are conducted recursively until no feature in F is redundant. Given a set of $|F|$ features, the time complexity of FCRR is $O(|F|^2)$, which will be costly if $|F|$ is big.

We propose an algorithm, *group feature contribution-based redundancy removal* (GFCRR), which divides the whole set F into κ groups by one-pass clustering and performs FCRR over each group to reduce the time cost of GFCRR. The partition over F is conducted by selecting κ seed groups that have lowest correlations between each other, calculating the FC of the rest of features with respect to the κ groups, and assigning each to its container group that has the lowest FC value from it. Turning the optimal subset selection in the whole feature set into those in a number of feature clusters, GFCRR algorithm may not guarantee the quality of selected feature subset. To make the selected feature subset close to the one obtained globally, we postprocess the clusters by recursively grouping the clusters with the highest correlation until the variance between the overall entropy of all the clusters and that of the whole feature candidate set is no more than the preset information loss threshold τ.

Theorem 2 *Given a feature set F and a threshold τ, let F' be the feature subset selected by FCRR, and F'_g that generated by GFCRR. For $\forall f_i \notin F'$, let its entropy be H, c_j its cluster generated in GFCRR, and A' its feature contribution to F. Then, the probability of $f_i \in F'_g$ is equal to $\frac{\Phi(\frac{\tau-\mu}{\sigma})-\Phi(\frac{\tau-A'-\mu}{\sigma})}{\sigma(\Phi(\frac{H-\mu}{\sigma})-\Phi(\frac{-\mu}{\sigma}))}$, where $\Phi(\cdot)$ is the cumulative distribution function of a normal distribution, $\mu \approx H$ and $\sigma \approx \sqrt{H}$*

Proof Let $F = \{c_1, \ldots, c_\kappa\}$ be the κ clusters generated in GFCRR and $f_i \in c_j$, and H_i the joint entropy of cluster c_i. Let y be the whole correlation of f_i with respect to $C \backslash c_j$. We have

$$0 \leq y < H \tag{12}$$

In statistics, y is a normal distribution. Then, y conditional on Eq. 12 obeys a truncated normal distribution with a probability density function[1] f given by Eq. 13.

$$f(y; \mu, \sigma, 0, H) = \frac{\frac{1}{\sqrt{2\pi}} \mathrm{e}^{-0.5((y-\mu)/\sigma)^2}}{\sigma\left(\Phi\left(\frac{H-\mu}{\sigma}\right) - \Phi\left(\frac{-\mu}{\sigma}\right)\right)} \tag{13}$$

where $\Phi(\cdot)$ is its cumulative distribution function, $\mu = \frac{H(\sum H_i - H)}{\sum H_i} \approx H$ and $\sigma = \sqrt{\frac{H^2(\sum H_i - H)^2}{(\sum H_i)^2(\sum H_i - 1)}} \approx \sqrt{H}$.
Recall A' be the feature contribution of f_i with respect to F. Let the feature contribution of f_i to c_j be A. As there may be

[1] https://en.wikipedia.org/wiki/Truncated_normal_distribution.

correlation between c_j and $c_{j'}$, the feature contribution A' of f_i with respect to F is no smaller than $A-y$, i.e. $A' \geq A-y$. Thus, we have

$$y \geq A - A' \tag{14}$$

As the overall correlation of all the clusters is up to τ and y is a part of the overall correlation, we have:

$$y \leq \tau \tag{15}$$

Based on Eqs. 14–15, we have

$$A - A' \leq y \leq \tau \tag{16}$$

For $f_i \notin F'$, we have $A' < \tau$. The probability of $f_i \in F'_g$ is equivalent to that of $A > \tau$. Under the condition of Eq. 16, the probability of $A > \tau$ is equivalent to that of $\tau - A' < y \leq \tau$, i.e.

$$\begin{aligned} Pr(A > \tau) &= Pr(\tau - A' < y \leq \tau) \\ &= \int_{\tau-A'}^{\tau} f(y; \mu, \sigma, 0, H) \\ &= \frac{\int_{\tau-A'}^{\tau} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-0.5((y-\mu)/\sigma)^2}}{\sigma\left(\Phi\left(\frac{H-\mu}{\sigma}\right) - \Phi\left(\frac{-\mu}{\sigma}\right)\right)} \\ &= \frac{\Phi\left(\frac{\tau-\mu}{\sigma}\right) - \Phi\left(\frac{\tau-A'-\mu}{\sigma}\right)}{\sigma\left(\Phi\left(\frac{H-\mu}{\sigma}\right) - \Phi\left(\frac{-\mu}{\sigma}\right)\right)} \end{aligned} \tag{17}$$

□

We randomly generated the features of 1000 media data over the whole feature candidate set F to measure the value of $Pr(A > \tau)$. The probability value over the randomly selected set is $Pr(A > \tau) = 0.096$. This indicates our GFCRR can guarantee a low feature selection error rate compared with FCRR.

The time complexity of GFCRR is $O(|F|^2/\kappa)$, which is κ times faster than FCRR. Note that we conduct tests over datasets having a small number of contexts. For datasets with large number of contexts, our GFCRR can save more time over the FCRR as analysed by the time complexities of these two approaches. We conduct FCRR and GFCRR over the D_s by computing the FC between each f_i and the set $F \backslash f_i$. The FC values and the percentages of all features using two methods are reported in Tables 3 and 4. Clearly, type can be removed from the candidate set. Note that in this work, the contribution of a feature to a feature set is only the non-overlapping part between them; thus, the sum of all feature contributions over the whole feature set is smaller than 100%.

Table 3 FC of features in FCRR

	Metadata	Location	Time	User	Type
FC	16.66	11.29	12.90	9.67	4.19
Perc (%)	24.02	16.28	18.60	13.94	6.04

Table 4 FC of features in GFCRR

	Location	Time	Metadata	User	Type
FC	16.77	11.93	21.61	11.29	4.17
Perc (%)	52.00	36.99	50.75	26.51	9.84

We compare our FCRR and GFCRR with the CAFS [13] to prove the superiority of our proposed algorithms. Following the parameter settings in [13], we compare the quality and efficiency of feature selection algorithms. CAFS outputs the same features as FCRR and GFCRR do, but takes about 40 times higher time cost due to its extremely high time complexity $O(|F|^3)$. Meanwhile, GFRCC costs 42.9% less time than FRCC due to its less feature contribution calculations over groups.

We also prove the superiority of our methods over pairwise feature redundancy reduction called FAST [17] by empirical analysis. In [17], a graph is constructed over the whole feature set, where each node is a feature and the weight of each edge is the correlation of its linked features. Based on the graph, a minimal span tree is built and divided into a number of clusters by cutting the edges whose weights are smaller than their correlations with respect to their class labels. Here, a class label corresponds to a query that collects data from social platforms. The most representative feature of each cluster is selected, and all the representatives for all clusters form the result set of the redundancy removal process. Table 5 reports the correlations of different feature pairs over D_s, where $R(F_i, F_j)$ is the correlation between two features F_i and F_j, $R(F_i, C)$ that between F_i and the class label C, and $R(F_j, C)$ that between F_j and the class label C. From the results, we can see that, for each feature pair, the correlation between its features is smaller than those between its features and the corresponding class label. Based on the clustering algorithm used in [17], all the edges will be removed from the minimal span tree. Therefore, the five features are assigned to five different clusters and selected as representatives in final redundancy removal process. Accordingly, no redundant feature is removed from the whole feature set. As analysed above and demonstrated in Sect. 6.3, type context is redundant. Obviously, this pairwise feature selection approach cannot maximally remove the redundancy.

5 Context-aware social recommendation

This section presents our context-aware recommendation solution. We first present a graph-based data model and then

Table 5 Feature relevance in FAST

Feature pair	$R(F_i, F_j)$	$R(F_i, C)$	$R(F_j, C)$
Metadata–location	0.31	0.82	0.75
Metadata–time	0.29	0.82	0.32
Metadata–user	0.47	0.82	0.56
Metadata–type	0.22	0.82	0.42
Location–time	0.28	0.75	0.32
Location–user	0.26	0.75	0.56
Location–type	0.18	0.75	0.42
Time–user	0.30	0.32	0.56
Time–type	0.16	0.32	0.42
User–type	0.23	0.56	0.42

propose a recommendation approach by matching a given user profile with the media data. We finally present our optimization for real-time processing.

5.1 Graph-based social data model

We will first present how to construct a uniform graph-based model over the selected feature set and then discuss the similarity measure between these graphs.

5.1.1 Content–context interaction graph

We aim to effectively and efficiently identify the relevance of a media data in shared community with respect to a user profile. To achieve this, we need to construct a uniform model that captures the content and contexts of media and the inherent interactions among them, and can adapt to various media types for real applications. Clearly, simply representing each media as a set of its features fails to capture the interaction between these features. We propose a *content–context interaction graph* (CCIG) model over the feature representations described in Sect. 4.1. CCIG is an undirected complete graph, in which each node is one of media content and contexts, and an edge between vertices indicates the correlation of two features. To mine this correlation, we construct a number of *m*-dimensional vectors. Each describes the statistic information of this feature over 4-grams from a metadata. We extract the comments of a media to reveal the distribution of its contexts such as time, location, and users. Each comment is described as a quadruple $c = \langle c_t, c_l, c_u, c_c \rangle$, where c_t is its time, c_l its location, c_u the user who posted it, and c_c the 4-gram form of its content.

Given a media v, we construct its content vector v_c by counting the frequency of each 4-gram appearing in the metadata. The time period of media v starts from its uploading and ends to the present. Given a time slot of v, we construct its time vector v_t by dividing the whole period into equal width bins and counting the number of bins containing each 4-gram in all c_c as a dimension value. Given a location set $\{c_l\}$, its location vector v_l is constructed by clustering the comments into a number of groups based on their geographical distance and then counting the number of groups containing each 4-gram. Based on the distance between locations, the location collection c_l of v is clustered into different geographical groups. The value of each dimension is the geographical group frequency of corresponding 4-gram in all c_l, which reveals the popularity of v in various areas. Given a user set $\{c_u\}$, its user vector v_u reflects the popularity of v among users and is obtained by counting the user frequency of corresponding 4-grams in all c_u. Likewise, we can construct the statistic vectors of any other features in case more contexts are available for media recommendation. After obtaining each feature vector, we exploit Pearson's correlation coefficient as the edge value between two arbitrary vertices in CCIG. Take the edge connecting metadata and time as an example. Given two vectors v_c and v_t, their Pearson's correlation coefficient is computed by:

$$\rho_{v_t, v_c} = \frac{\text{cov}(v_t, v_c)}{\delta_{v_t}\delta_{v_c}} \tag{18}$$

where cov is the covariance and δ the standard deviation. Once the feature interaction is constructed, we can model each media as a CCIG as shown in Fig. 3.

We use user browsing history for profile construction. Given a user, its profile includes his interested media, described as a CCIG set. The recommendation is conducted by matching a user profile with each media data in database.

5.1.2 CCIG graph similarity

We design a graph dissimilarity measure for matching a media data (CCIG) to a user profile (CCIG set). If a media data match any of media in a profile, this media data are relevant to this user. We propose pair-node clique model-based dissimilarity over two CCIGs, where two compared cliques correspond to two node pairs with same attributes. In CCIG, a

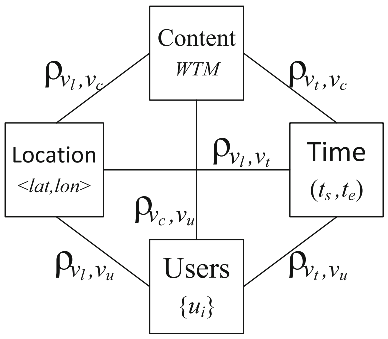


Fig. 3 CCIG

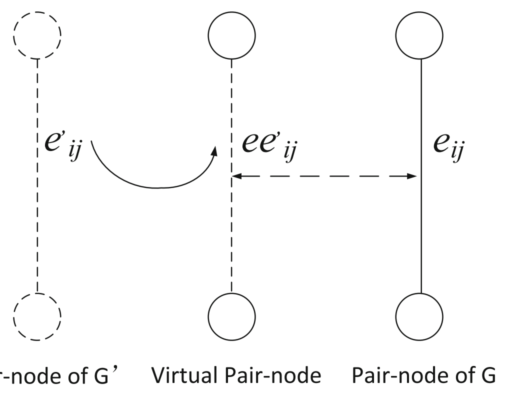


Fig. 4 Metadata-time clique

pair-node clique is comprised of two nodes and the edge connecting them. Suppose we have two CCIGs, G and G', the dissimilarity between two pair-node cliques $\phi(n_i, n_j, e_{ij})$ and $\phi(n'_i, n'_j, e'_{ij})$ is defined as:

$$cs(\phi, \phi') = \gamma\left(\left(\mathrm{Dis}_i + \mathrm{Dis}_j\right)/2\right) + (1-\gamma)\rho(e_{ij}, e'_{ij}) \quad (19)$$

where Dis is the distance between two nodes as defined in Sect. 4.1, ρ is the distance between e_{ij} and e'_{ij}, and γ denotes the weights of nodes in clique similarity. To measure the distance between e_{ij} and e'_{ij}, we migrate all the comments of G' to G and recalculate the correlation between nodes i and j, represented as a virtual edge ee'_{ij}. If the difference between e_{ij} and ee'_{ij} is small, the two media data have been discussed over two features in a similar pattern in G and G'.

Figure 4 shows how to decide the distance between metadata-time cliques of G and G'. The comments of G' are migrated to G to form a virtual pair node. The distance between e_{ij} and e'_{ij} is calculated by the difference between virtual edge ee'_{ij} and e_{ij}. To make $\rho(e_{ij}, e'_{ij})$ symmetric, we migrate all the comments of G to G' and recalculate the correlation $e'e_{ij}$ between n'_i and n'_j. The distance between e_{ij} and e'_{ij} is computed by:

$$\rho(e_{ij}, e'_{ij}) = (|e_{ij} - ee'_{ij}| + |e'_{ij} - e'e_{ij}|)/2 \quad (20)$$

Let λ_i be a pair-node clique weight, and ϕ a CCIG to a pair-node clique set, the dissimilarity of G and G' is computed by:

$$\mathrm{GS}(G, G') = \sum_{i=1}^{|\phi|} \lambda_i cs_i(\phi_i, \phi'_i) \quad (21)$$

Using GS dissimilarity, we can match a user profile with a media in database. Given a user profile described as a CCIG set, $\{G_{ui}\}$, and a media represented as a CCIG, G_{dj}, their dissimilarity, R_{GS}, is the minimal GS between G_{dj} and G_{ui}.

$$R_{\mathrm{GS}}(\{G_{ui}\}, G_{dj}) = \min_{i=1}^{\|\{G_{ui}\}\|} \mathrm{GS}(G_{ui}, G_{dj}) \quad (22)$$

Suppose that the number of features used in CCIG model is $|F|$, then the number of pair-node cliques in a CCIG graph $|\phi|$ is $|F| \times (|F|-1)/2$. Let ι be the number of removed features in feature selection. The number of pair-node clique comparisons in each GS matching is reduced by $\frac{(2\times\iota\times|F|+\iota^2+\iota)}{2}$, which is exponential to the number of reduced features. Thus, removing redundant features can greatly reduce the cost of recommendation.

5.2 Social data matching optimization

Context-aware media recommendation is performed by the R_{GS}-based matching over a user profile and each media in database. Given a profile $\{G_{ui}\}$ of a CCIG set, a naive recommendation method is to compute the R_{GS} between $\{G_{ui}\}$ and each G_{dj} in database and find K media with the smallest distances. However, this approach incurs high time cost due to the complexity of R_{GS} and large size of big social media. It is demanded that efficient solution is proposed to identify the most relevant CCIGs in media database without many GS-based computations.

5.2.1 CCIG index structure

We design a multi-level index on CCIGs to improve the efficiency of R_{GS}-based user profile matching. Our index consists of three core parts: (1) a hash scheme used for generating multiple hash keys over content, time, and location of CCIGs; (2) a chained hash table that maps each user to a cluster id, and multiple B^+-trees that are used to store the index keys to other attributes; and (3) a number of buckets that store the related media IDs to identify media data. In this work, we only consider the features that can be ordered using hash-based index and leave the extension to nominal values or numerical values for future research from the following aspects. For the contexts with numerical values, like browsing duration, we can simply quantize the values into symbols and store them in an array. With respect to the nominal values that cannot be ordered, such as the cloth colour or style, the concept clustering can be exploited to group the objects into a particular hierarchical categorization structure. Figure 5 shows our index structure.

We apply the locality-sensitive hashing (LSH)-based strategy over the content, time, and location and exploit LSB indexes [37] for the Z-order values of hash keys over these attributes, since LSH can approximately identify the top K relevant attribute values without any costly distance calculations. To use LSH on a metric, this metric must satisfy the

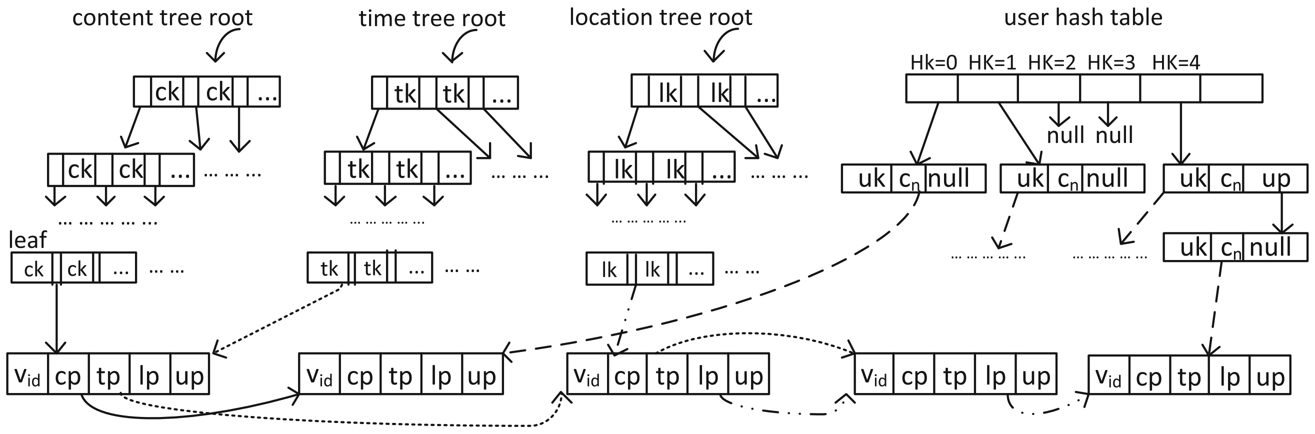


Fig. 5 CCIG-index structure

triangle properties. Thus, we embed the J-divergence-based tensor distance into L_2 space, so the LSH can be applied in the new space. FastMap [38] maps objects into points in $\bar{k}$-dimensional L_2 space, which preserves the similarities between objects. A higher $\bar{k}$ of L_2 space produces more accurate mapping results. Given a dimension $i \in \{1 \ldots \bar{k}\}$, two reference points x_i and x_i' are selected. Then, an object a is embedded into the L_2 space by Eq. 23:

$$F_i(a) = \frac{J(x_i \| a)^2 + J(x_i \| x_i')^2 - J(x_i' \| a)}{2J(x_i \| x_i')} \tag{23}$$

Applying FastMap to our content model, a tensor representation is converted into a $\bar{k}$-dimensional vector, which is further converted into an Z-order value. The Z-order values of all nodes are mapped to hash keys that are organized using an LSB index. Likewise, with respect to location model, we adopt FastMap to embed the location distance into L_2 space. Thus, we can convert each pair of location coordinates into a Z-order value and further into a location hash key. All these location hash keys are stored in a location LSB index. As we fixed time periods for all media data as a uniform value, the neighbouring relationship of media data over time is same as that of their start points in L_1 space. Thus, we can directly map all the time points to the Z-order values of their start points. Similarly, we use a time LSB index for the hash keys of these Z-order values over time. We adopt the chained hash table [8] to organize the user id clusters, by which a user is mapped into a cluster.

At the bottom level of the index, an inverted list including a number of buckets is used to store the linked media data. Each bucket element is a quintuple formed as $\langle v_{\text{id}}, \text{cp}, \text{tp}, \text{lp}, \text{up}\rangle$, where v_{id} denotes the media data id, cp the pointer to the next element having the same content hash code, tp to the next one with the same time hash code, lp to the next one with the same location hash code, and up to the next one with the same user id cluster. The space cost of our index is decided by the size

```
Procedure KTopRelevantMediaQuery.
input:  $\mathcal{H}_u$ - chained hash table to user nodes
        $I$-Inverted file    $\mathcal{LSB}_c$ - content LSB-tree
        $\mathcal{LSB}_t$ - time LSB-tree   $\mathcal{LSB}_l$ - location LSB-tree
        $Q$ : {CCIG$_i$}-A user profile
        $K$ - Number of top score media data
output: $KNN\_list$ - A list of $K$ top score media data
1. for each CCIG$_i^u \in$ {CCIG$_i$}
2.    $\{d_{Qi}\} \leftarrow$ SocialDescriptorVectorization($\mathcal{H}_u$,CCIG$_i^u$)
3.    $\{v_i\} \leftarrow$ GetSocialRelevanceCandidates($I, \{d_{Qi}\}$)
4.    $\{v_{ri}\} \leftarrow$ RankRelevanceCandidates($\{v_i\}$)
5. Repeat /*SearchLSB($\mathcal{LSB}, Q_f$)*/
6.    for each CCIG$_i \in$ { CCIG$_i$}, pick the leaf entry, $e_i^c$,
      from $\mathcal{LSB}_c$ having the next longest common prefix
      with CCIG$_i^c$, pick the leaf entry, $e_i^t$, from $\mathcal{LSB}_t$
      having the next longest common prefix with CCIG$_i^t$,
      pick the leaf entry, $e_i^l$, from $\mathcal{LSB}_l$ having the next
      longest common prefix with CCIG$_i^l$
7.    $\{v_i\} \leftarrow$ GetMediaCandidates($\{e_i^c\}, \{e_i^t\}, \{e_i^l\}$)
8.    $v_n \leftarrow$ GetNextMostRelevantMedia($\{v_{ri}\}$)
9.    for each $v_i \in \{v_i\} \cup \{v_n\}$
10.       ComputeGS($V_i$,CCIG$_i$)
11.       GetUserProfileGS
12.       UpdateKNN_list
13. Until $K$ top score media are found
14. return $KNN\_list$
```

Fig. 6 Computing K top relevant media data

of database and that of users. Given a database of $|D|$ media data, let $|hk|$ be the size of each hash key, $|us|$ the overall size of all user names, |CCIG| the size of each CCIG, where $|hk|$ and $|CCIG|$ are constant. The space cost of our index would be $O(|D| * (|hk| * 4 + |\text{CCIG}|) + |us|)$.

5.2.2 *k*NN query

With CCIG index, we perform the social media recommendation based on *k*NN query. Figure 6 shows the algorithm for computing K top score media of a given user profile {CCIG$_i$}. Our algorithm is performed by three steps: (1) find a list of media candidates and rank this list for each CCIG

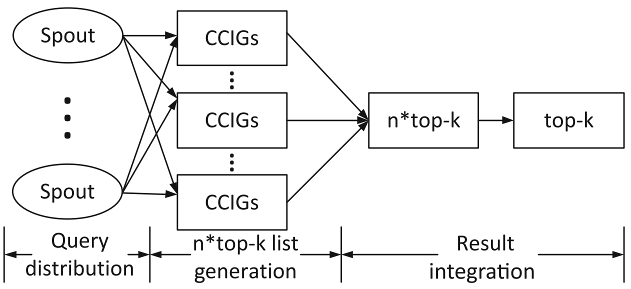


Fig. 7 Storm topology framework

in user profile (lines 1–4); (2) obtain a number of media candidates based on content, time, and location by searching the LSB trees for all CCIGs in user profile (lines 5–7); (3) refine the media candidates by first obtaining the most relevant media set for each CCIG in user profile, then calculating the GS between this CCIG and each candidate, and obtaining the R_{GS} between each candidate and the user profile (lines 8–11). The kNN_list is updated if the GS of a candidate is smaller than that of its top K relevant media (line 12). Steps (2) and (3) are recursively performed, until all K top relevant media are identified, and returned to the users (lines 5–14). Given a social user profile of $|p(u)|$ media and $|U|$ involved users, the search cost of our algorithm is: $O(|p(u)| * |D| * \xi + |U| * \varrho)$, where $|D|$ is the database size, ξ the ratio of candidate size to the database, and ϱ the average cost of user name matching.

We take advantage of Apache Storm to implement a real-time recommender system. Storm is a real-time fault-tolerant and distributed data processing system, which includes two main parts, Spouts and Bolts. Spouts distribute the data from the sources to various bolts, while Bolts receive the inputs and work as processing units accomplishing different functions. Moreover, each Bolt can distribute the intermediate results to other bolts for further processing. For Apache Storm real-time process, a topology including spouts and bolts is necessary, where nodes in a topology stand for data processor and edge illustrate how data are transferred.

Figure 7 shows our topology for real-time processing. Our topology consists of three parts: (1) a number of spouts that distribute query data from a user profile to all bolts; (2) a set of bolts that conduct candidate filtering; and (3) a result integration component that selects the top k relevant media over the candidates from different bolts. Suppose we have n bolts for query processing, the whole media dataset is divided into n subsets, and each is allocated to a bolt. To maximize the utilization of processors, we propose a novel uniform partition strategy called UP, which uniformly distribute the whole dataset to n bolts. Given a dataset {CCIGs}, UP performs the partition by two steps. First, the whole dataset is divided into $\bar{m}$ groups based on the distances between CCIGs by one-pass clustering algorithm [39]. We adopt this algorithm because

```
Procedure MediaUpdateMaintenance.
input: $\{CCIG-index_i\}$ - $n$ CCIG indexes
       $\{v_i\}$ - a set of updated media data
output: $\mathcal{S}_g$ - $n$ updated CCIG-indexes
1.SocialUpdatesMaintenance($\{v_i\}$)
   /* user, sub-communities*/
2.for each $v_i \in \{v_i\}$
3.    ComputeTimeandDiscretization
4.    ComputeLocationDiscretization
5.    ComputeContentContextInteraction
6.    $\{CCIG_i\}$ ←ConstructCCIG
7.    if $v_i$ is in media database
8.        IndexDeletion($v_i$)
9.$\{SG_i\}$ ←UP($\{CCIG_i\}$)
10.for each $SG_i \in \{SG_i\}$
11.   IndexInsertion($\{SG_i\}$)
12. return $S_g$
```

Fig. 8 Maintaining social media updates

of its high clustering speed. Then for each group of CCIGs, we distribute its CCIGs to n subsets uniformly. As such, all the subsets will have similar distributions of CCIGs. Accordingly, the workloads on all bolts will be similar in the process of recommendation.

After obtaining n CCIG subsets, we distribute them on n bolts. The CCIGs of each bolt are organized as a CCIG index. Given a relevant media number K and a user profile, spouts distribute this profile to each bolt, where a KTopRelevantMediaQuery process is conducted. n top K relevant media lists are generated from the bolts and passed to the result integration component that selects the final top K relevant media. In this work, we construct a CCIG index over each data partition and allocate it on a processor in distributed environment. We will leave the problem of allocating one CCIG index to multiple processors for future research.

5.2.3 Dynamic data maintenance

We discuss how to maintain our CCIG model and the index under Apache Storm in dynamic environment. Due to the user activities, media content and contexts may change over time. Thus, except for the dynamic sub-community maintenance, dynamic updates are necessary for time and user representations, the correlation of different attributes, the data allocation, and index as well. We will discuss the details on how to update each of them periodically. The details on our media context maintenance algorithm are shown in Fig. 8. First, we update the user representation and sub-communities by using the social update maintenance method in [8] (line 1). Then, for each data with update, we compute its time representation, time and location discretization, interactions of its attributes, and construct its CCIG (lines 2–6). If the media data are in database already, we remove it from existing database and index (lines 7–8). Following that, we perform partition over all media data with updates and insert the CCIGs of each subset into the CCIG index over its bolt (lines 9–11). Finally, the updated CCIG indexes are returned (line 12). Let the time cost of sub-community maintenance be T_{mc}, $|\{v_i\}|$ the number of the updated media data, the cost of our context update maintenance is: $O(T_{\mathrm{mc}} + |\{v_i\}| * (C_{\mathrm{t}} + C_{\mathrm{td}} + C_{\mathrm{ld}} + C_{\mathrm{m}}))$, where C_{t} and C_{m} are the costs of representing time and metadata, respectively, and C_{td} and C_{ld} are those of time and location discretization.

6 Experimental evaluation

We report our test results to evaluate the effectiveness and efficiency of our proposed recommender system.

6.1 Experimental setup

We conduct experiments over 3,053,000 data from two sources: (1) a large set D_{w} of real media collected from YouTube based on 20 popular queries (mariah carey; miley cyrus; american idol; wwe; lil wayne; chris brown; runescape; naruto; alicia keys; beyonce; leona lewis; nba; obama; shakira; hannah montana; christina aguilera; funny cats; taylor swift; michael jackson; jonas brothers) [40]; (2) two synthetic datasets generated by the synthpop package of R language [41] which provides routines to generate a synthetic version of an original dataset. Using synthpop, we learn the probability distributions of variables from a real dataset, from which the synthetic values are drawn. For each real media data, we treat all its content and contexts as variables that form the input of synthpop for learning variable distributions. The whole real dataset, D_{w}, consists of the media data of 753,000 videos that were uploaded to YouTube from 2012 to 2016. For each video, we kept its contents including title and description and its contexts including time, location, category, and users who commented or liked it. We generate a dataset D_{syw} of 2,000,000 synthetic media data based on D_{w} and use two types of datasets, D_{w} and D_{syw}, for efficiency test. D_{syw} has the same data distribution as D_{w}, but is cleaner than it, since the data distribution learning filtered out the noises input by users.

As Google Trend's statistics analysis on YouTube from 2012 to 2016, the user interests with respect to different items are similar in different years [44]; thus, the recommendation quality of the system over a media subset can reflect that over its whole set. Therefore, to save the overall time for parameter turning and keep the dataset consistency in effectiveness evaluation part, we use three media subsets for effectiveness test: (1) a subset D_{s} selected from D_{w}, consisting of 150,000 videos uploaded to YouTube from January to December 2016; (2) a sparse subset D_{ss} consisting of 120,341 videos whose comment and "liked" number is less than the average number for all media in D_{s}; and (3) a synthetic dataset D_{sys} of 300,000 media data generated based on D_{s} by the synthpop package of R language. For the media from each query, we select 2 top active users as the recommended targets. As such, we obtain a set of 40 target users for each test dataset. We check each media recommended to a user and make sure if it is accepted by this user. If a media has been commented or liked by this user after recommendation, it is successfully recommended.

6.2 Evaluation methodology

We have conducted extensive experiments to evaluate our system from two aspects: (1) effectiveness and (2) efficiency. First, we evaluate the parameters that decide the system effectiveness, including the effect of media popularity time length T, the effect of node weight γ, and the pair-node clique weights λ_i, and obtain their optimal values. We then compare our approach with existing methods using the default parameters in terms of effectiveness and report the recommendation results using six methods, including two state-of-the-art competitors, CTT [3] and FWC[42], and four CCIG-based methods, CCIG-MU, CCIG-MT, CCIG-ML, and CCIG-MTLU. Here, CTT is a collaborative filtering, type, and time fusion-based approach [3], while FWC is a popular feature-based recommender system that assigns weights to features depending on their importance to users [42]. CCIG-MU is our metadata and user-based method, CCIG-MT our metadata and time-based method, CCIG-ML our metadata and location-based method, and CCIG-MTLU our metadata, time, location, user-based method. Our previous studies [8,9] are relevant to this research. However, they are non-personalized recommendation and assume the target user profile is not available, which is unsuitable to the personalized recommendation. Thus, we do not compare this CCIG-based recommendation with our methods proposed in [8,9] for the experimental evaluation.

The system effectiveness is evaluated by a metric *recall*, also known as hit rate, denoted as *recall*@N, which is the percentage of successful recommendations in the user interest list [3]. Let R_N be the number of recommended media and A_N that of accepted media. *recall*@N can be computed by Eq. 24.

$$Recall@N = \frac{R_N \cap A_N}{A_N} \tag{24}$$

To measure the recall of our system, we divide the whole view history based on a time point into two parts: training set and test set. The training set is used by the system to recommend the top- ranked media lists. A recommendation appearing in the test set is a successful one. The recall metric reflects the real interests of target users, which is not affected by subjective user views. We evaluate the system efficiency in terms of the overall time cost of recommendation over the whole dataset using our CCIG index under Apache Storm. All experiments are conducted on a server using an Intel Xeon E5 CPU with 256 GB RAM running RHEL v6.3 Linux.

6.3 Effectiveness

We first evaluate the effect of parameters T, γ, and λ_i and then compare our solution with the state-of-art recommendation methods, followed by the test on the effect of social updates.

6.3.1 Effect of media popularity time length *T*

We test the system effectiveness over D_{s}, D_{ss}, and D_{sys}, using time-based recommendation by varying T from 1 to 10 months to obtain an optimal T in CCIG. Time-based recommendation is a special case of our CCIG-based method, where each CCIG has one node and has no any edge. As time-based recommendation is based on the time context only, we do not need to set γ and λ_i, which are required by multiple feature-based CCIG graph matching. Given a target user, the time-based recommendation can be easily done by performing the *k*NN query over our time LSB index as in traditional LSB index [37]. For each T, we recommend the top 3, 5, 10, 20 media to users. Figure 9 shows the recall of the recommendation over each of three datasets. Clearly, the recall of recommendation increases because more active user information is captured with the increase of T from 1 to 7. With the further increase in T, more inactive time slots are involved, causing the decrease in recall. Since the sparse data D_{ss} are extracted from the dataset D_{s} whose comment and "liked" number is less than the average number of all media, these two datasets are same in terms of the feature number and the feature correlations. The difference between them exists in the density of comments to each media in them, which does not affect the importance of different contexts. Thus, the recommendation quality change trends over them are same. However, the user interest prediction over data with high density comments will be more accurate, because more accurate context information can be applied for the recommendation. Thus, we set the default T to 7.

6.3.2 Effect of node weight *γ*

We test the effect of γ in clique similarity on the recall of the recommendation over each of D_{s}, D_{ss}, and D_{sys}, to find the optimal γ. We set T to its default value. By varying γ from 0 to 1, we measure the optimal recall of our system at each γ value under all combinations of λ_i ($i = 1, \ldots, 6$), where the sum of λ_i is 1. As shown in Fig. 10, for each dataset, the recall increases gradually with the increase in γ from 0 to 0.7,

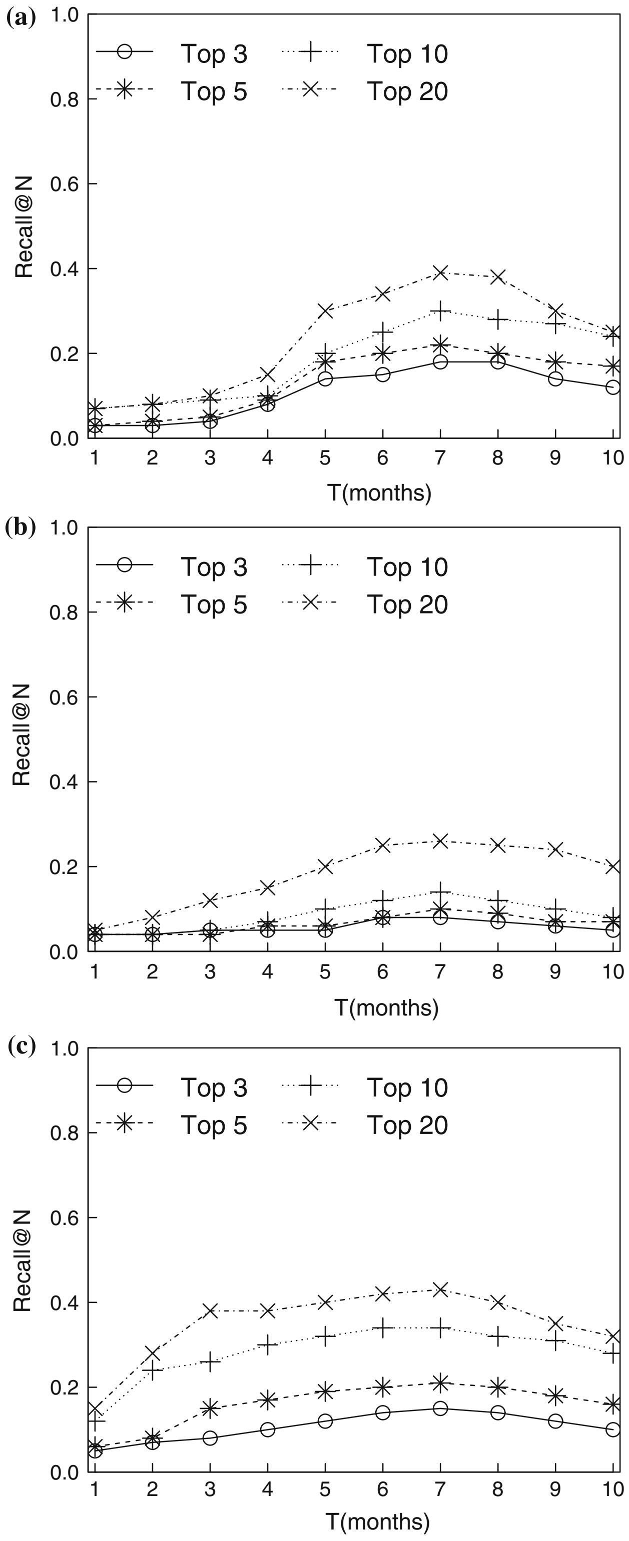


Fig. 9 Effect of time length. **a** D_{s}, **b** D_{ss}, **c** D_{sys}

reaches its best value, and drops after that. This is because extremely small γ cannot capture enough information on the pair-node difference. With the increase in γ, more node

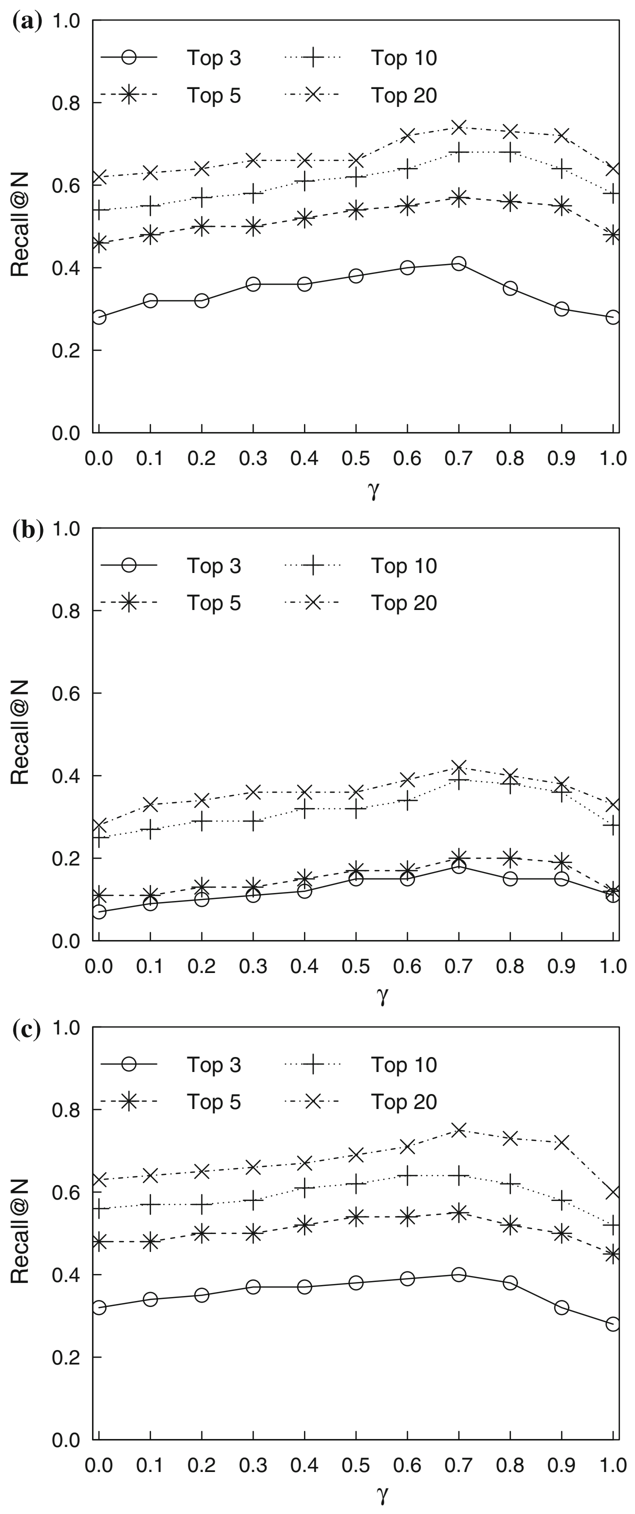


Fig. 10 Effect of node weight. **a** D_{s}, **b** D_{ss}, **c** D_{sys}

information is applied to improve the effectiveness of system. After $\gamma = 0.7$, because the extreme node information is introduced, the edge difference cannot be captured. Thus, we set the default γ as 0.7. As we tested all the combinations of parameter settings, it took long time to complete the whole set of tests, including 4.33 hours over D_{s}, 3.87 hours over D_{ss}, and 9.53 hours over D_{sys}.

6.3.3 Effect of pair-node clique weight $\boldsymbol{\lambda_i}$

We test the effect of λ_i $(i = 1, \ldots, 6)$ on the recall of our system over D_{s}, D_{ss} and D_{sys}, where λ_1 is for metadata and time pair (M–T), λ_2 for metadata and location pair (M–L), λ_3 for metadata and user pair (M–U), λ_4 for time and location pair (T–L), λ_5 for time and user pair (T–U), and λ_6 for location and user pair (L–U). We fix T and γ to their default values. For each combination of λ_i $(i = 1, \ldots, 6)$, where the sum of λ_i is 1, we test our system effectiveness. We first decide the optimal λ_1 by varying it from 0 to 1 and testing the highest recall of recommendation at each λ_1 value. For each λ_i $(i = 2, \ldots, 6)$, we fix λ_j $(j = 1, \ldots, i-1)$ to their default values and test its effect on the highest recall by turning it from 0 to 1 and reporting the optimal recall at each λ_i value. Figures 11, 12 and 13 show the highest recall of our system at each λ_i value for D_{s}, D_{ss}, and D_{sys}. Clearly, the recommendation recall increases for each of three datasets as the increase in each λ_i $(i = 1, \ldots, 5)$ and drops after its optimal value. We can see the optimal values $\lambda_1 = 0.3$, $\lambda_2 = 0.1$, $\lambda_3 = 0.2$, $\lambda_4 = 0.1$, $\lambda_5 = 0.2$. Once the first five λ_i are fixed, we obtain the optimal $\lambda_6 = 0.1$. The optimal λ_1 takes the highest weight due to the high correlations of metadata and time to the feature class as analysed in Sect. 4.2.2, which leads to a much higher joint entropy of this feature pair. To quantize the contributions of different node pair cliques in the measure, we reported the joint entropy of each node pair over three datasets in Table 6. Since location has the lowest joint entropy with other features, the weights to pairs containing the location node are lowest.

6.3.4 Effectiveness comparison

We compare the effectiveness of six methods, CCIG-MU, CCIG-MT, CCIG-ML, CCIG-MTLU, CTT [3], and FWC[42], by performing kNN-based recommendation over D_{s}, D_{ss}, and D_{sys}. Please note the first four options belong to our CCIG-based solution, but applied different features in recommendation. We set all the methods to their optimal parameter settings as shown in Tables 7 and 8 and recommend top- ranked data to each of target users. Here, for FWC, w_o is intercept, w_{m} slop for metadata, w_{l} slop for location, w_{u} slop for user, and w_{t} slop for time. For CTT method, ξ means time decay and β feature weight.

Figure 14 shows the comparison of six methods in terms of recall. Obviously, our CCIG-MTLU achieves the highest recall for recommending different numbers of top relevant media over three datasets. For D_{s} and D_{sys}, CCIG-MTLU is followed by CTT. This is because CCIG-MTLU fully

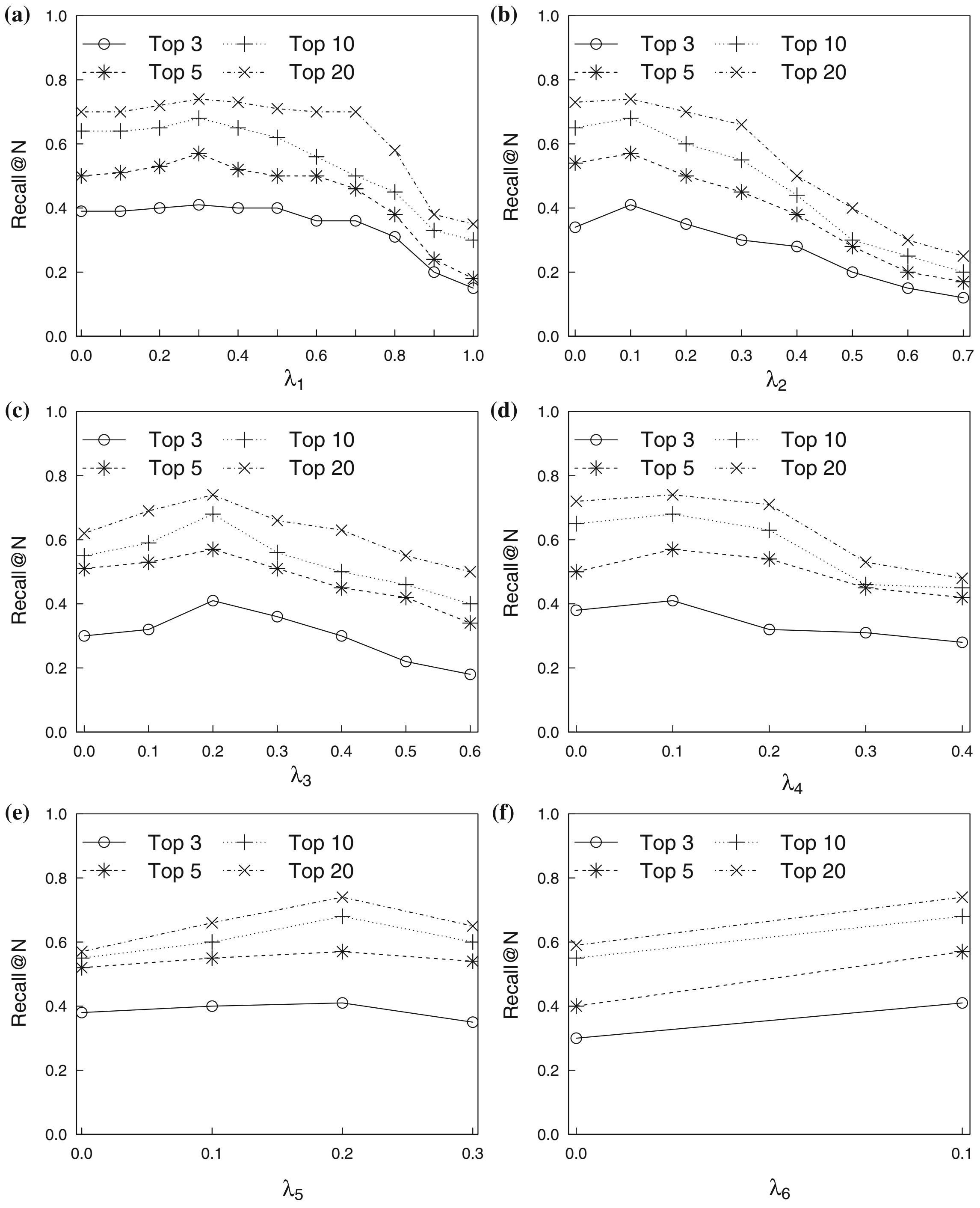


Fig. 11 Effect of pair-node clique weight on D_s

exploits all the context information needed as analysed in Sect. 4 and their interactions for recommendation, leading to the high effectiveness of system. CTT achieves higher recall over D_s and D_{sys} comparing with CCIG-MU, CCIG-MT and CCIG-ML, since it introduces collaborative filtering and considers the instant feedback of user actions, better reflecting the recent user interests. However, without the location and feature interactions, the recall achieved by CTT is much lower than our CCIG-MTLU. FWC performs worse than CTT and CCIG-MU because it only takes advantage of different features. Compared with CTT, FWC ignores the contributions of collaborative filtering in recommendation. While compared with CCIG-MU, FWC does not consider the feature interactions. Thus, our CCIG-MTLU is superior to other competitors in terms of effectiveness.

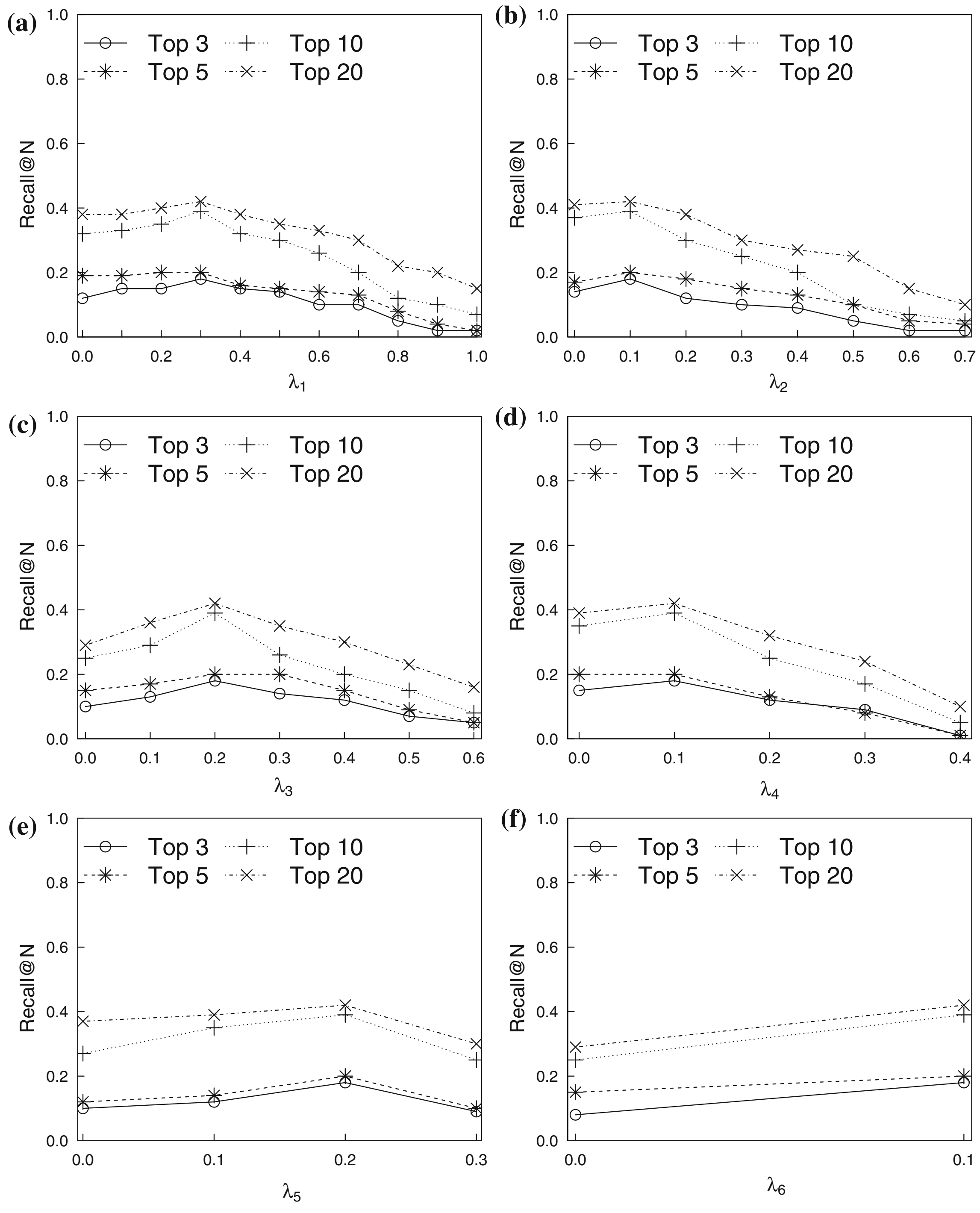


Fig. 12 Effect of pair-node clique weight on D_{ss}

Considering the performance difference of different methods over the sparse dataset, as shown in Fig. 14b, each method achieves lower effectiveness over sparse data due to the less context information in them. Meanwhile, CTT incurs the most serious downgrade over sparse data compared with its performance over D_s and D_{sys}. This is because CTT heavily depends on the ratings among media and users, while sparsity makes it hard to calculate the similarity by fewer ratings.

6.3.5 Effect of media update

We test the effect of media data updates over D_s, D_{ss}, and D_{sys}, on the effectiveness of our system. Each of D_s, D_{ss}, and D_{sys} is divided into two parts: (1) a test set containing the media content and contexts in recent 4 months of it and (2) a source set containing those appearing in first 8 months of it. Figure 15 shows the recall changes of our

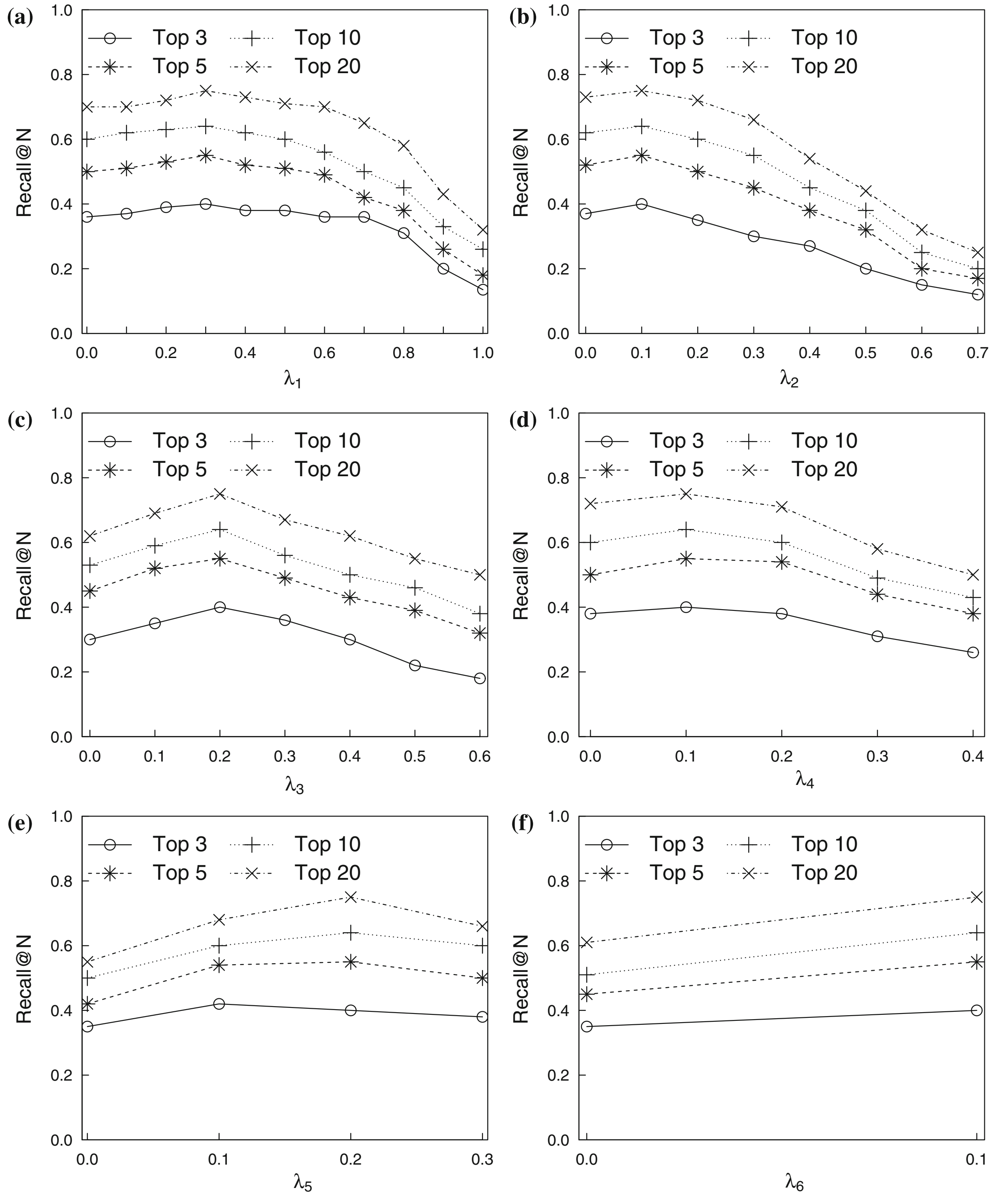


Fig. 13 Effect of pair-node clique weight on D_{sys}

recommender system with different update sizes over three datasets. Obviously, for each dataset, the recall of our system keeps steady with slight increase due to two reasons. For one thing, our method well keeps the media contexts and interactions. For another, as new data come, more relevant media are recommended. This has proved our method can well handle the media updates. The steady system performance over dataset with four months updates indicates that we can do the parameter tuning periodically, while do not need to do that frequently to keep the high-quality recommendation. In practice, as the new items have fewer comments, the recommendation over them will be almost a cold-start processing without social connections, leading to the lower rank of them in terms of relevance score. Accordingly, comparing with existing data, the new items have a lower probability of being

Table 6 Joint entropy of feature pairs

	$H(M,T)$	$H(M,U)$	$H(T,U)$	$H(M,L)$	$H(T,L)$	$H(L,U)$
D_s	28.24	26.32	25.24	17.26	16.28	14.25
D_{ss}	21.58	19.84	16.25	13.08	10.36	9.58
D_{sys}	25.54	24.26	21.89	15.36	11.20	10.85

Table 7 Parameter settings in our alternative approaches

	γ	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
MU	0.6	0	0	1	0	0	0
MT	0.7	1	0	0	0	0	0
ML	0.6	0	1	0	0	0	0
MTLU	0.7	0.3	0.1	0.2	0.1	0.2	0.1

Table 8 Parameter settings in existing approaches

FWC					CTT	
w_o	w_m	w_l	w_u	w_t	ξ	β
0.083	0.272	0.221	0.045	0.035	0.02	10

recommended; thus, the recall should not increase greatly with the increase in new data input in short time period.

6.4 Efficiency

We evaluate our system efficiency by testing the effect of CCIG index over a single processor and that of UP partition over multiple processors, comparing with existing method [3], and testing the effect of media updates.

6.4.1 Effect of CCIG index

We evaluate the effect of CCIG index by conducting the K top relevant media query with the index over D_w and D_{syw}. Here, K is set to 20. We compare the overall response time of recommendation using two methods: the CCIG matching with index (CCIG-index) and without index (CCIG), by varying the size of media collection from 1 to 5 years data. Given two CCIGs, the GS distance between them with respect to any of their pair-node clique subsets is no bigger than the distance over their whole clique set. Thus, we can apply the partial distance-based filtering to both approaches, which considers a pair-node clique subset of CCIGs to exclude the false alarms without the full GS calculation.

As shown in Fig. 16, our approach with CCIG-index is much faster than that without index. Especially, as the dataset size increases, the cost of the CCIG index increases much slower than that for CCIG matching without index; thus, the cost gap becomes bigger. This is because simultaneously searching the LSB-trees over multiple features quickly

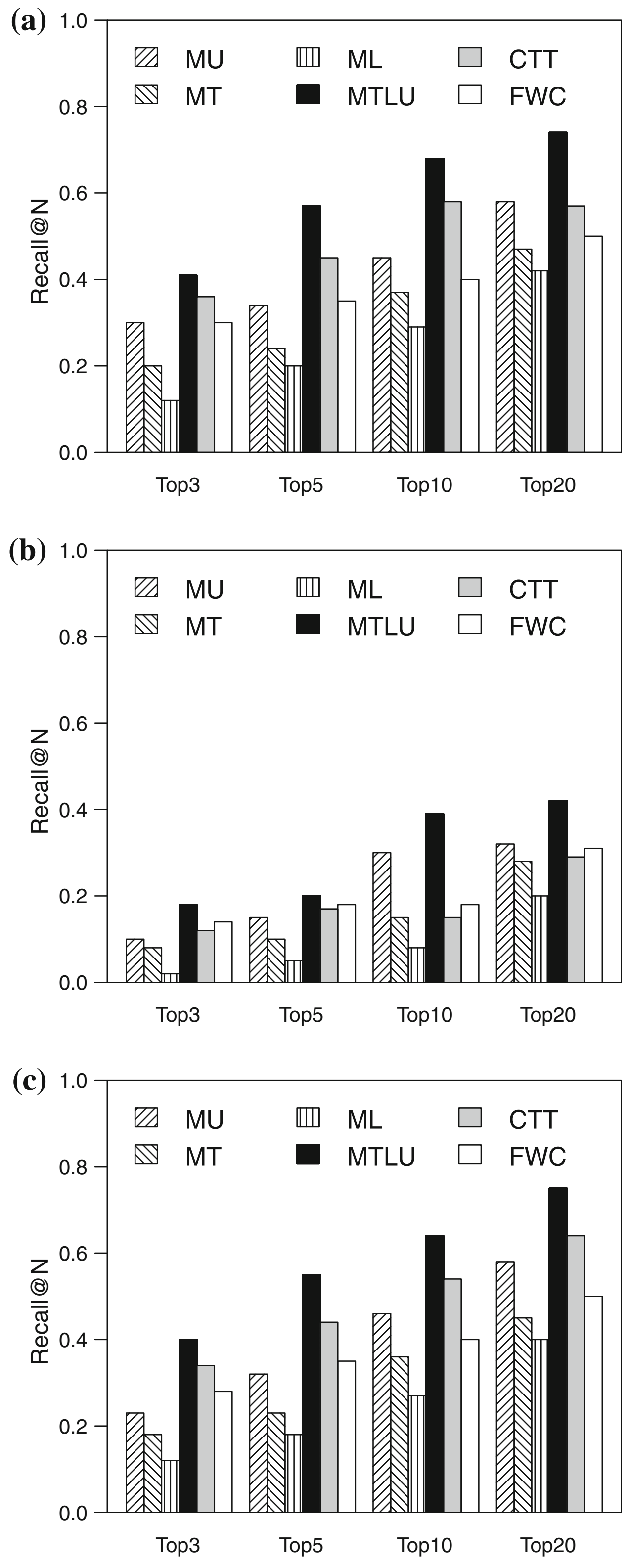


Fig. 14 Effectiveness comparison. **a** D_s, **b** D_{ss}, **c** D_{sys}

excludes the media with low relevance with extremely low filtering cost, which greatly reduces the number of GS calculations. This has proved the high effect of our CCIG-index.

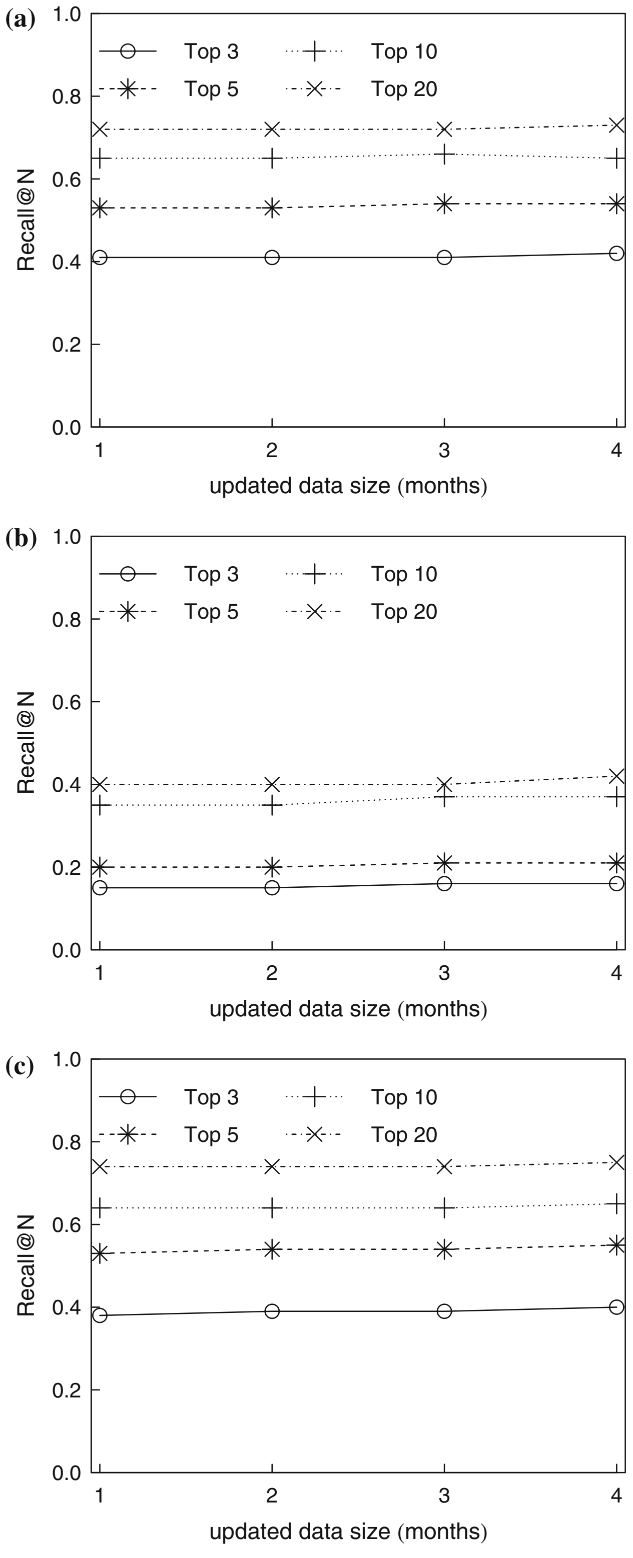


Fig. 15 Effect of context updates on effectiveness. **a** D_s, **b** D_{ss}, **c** D_{sys}

6.4.2 Effect of media dataset partition

We examine the effect of our proposed UP over D_w and D_{syw} for K top relevant media query over Apache Storm,

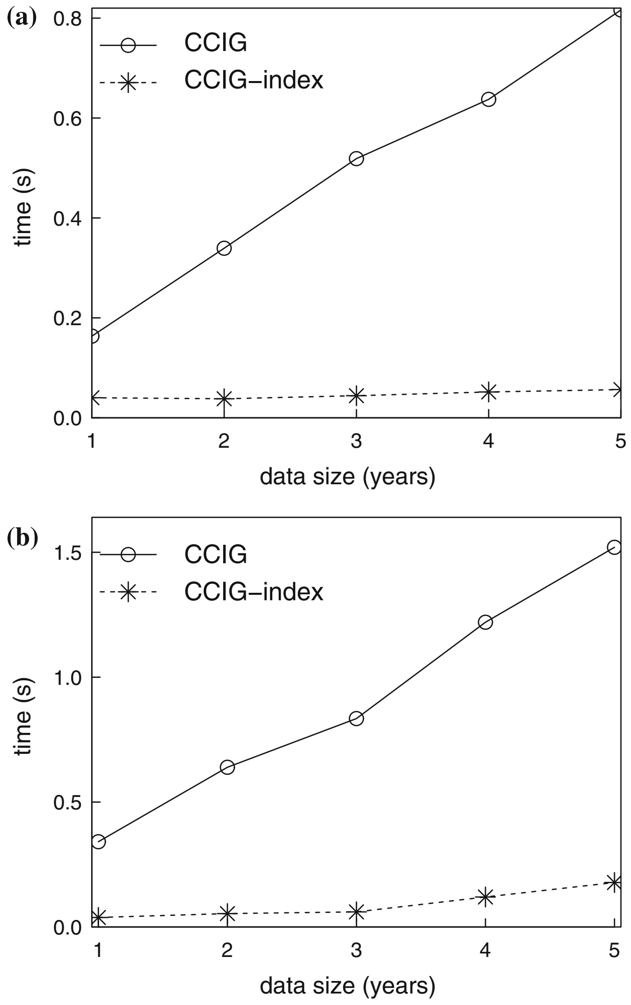


Fig. 16 Effect of index. **a** D_w, **b** D_{syw}

where K is set to 20. We compare our method (UP) with the Apache Storm built-in fields grouping by the time attribute (FG-T) [43]. We set the distance threshold of one-pass clustering to 0.2, 0.4, and 0.6, respectively, and test the time cost changes of UP under different thresholds, UP-0.2, UP-0.4, and UP-0.6. By fields grouping, the whole time range of the dataset is divided into n equal-length periods, and the media in the same period are allocated on a bolt. We test the time cost of our system by varying the bolt number n from 2 to 10. Figure 17a, b shows the cost comparison of two data partition strategies over two datasets.

Obviously, UP under each threshold achieves higher efficiency than the field grouping for different bolt numbers. For one thing, the dataset is uniformly distributed to the bolts with our UP method; thus, the workloads to different bolts are close. All the bolts can be fully exploited, leading to high system efficiency. In terms of our approach with multiple parameter values, the smaller distance threshold leads to more balanced data distribution over each bolt, and fur-

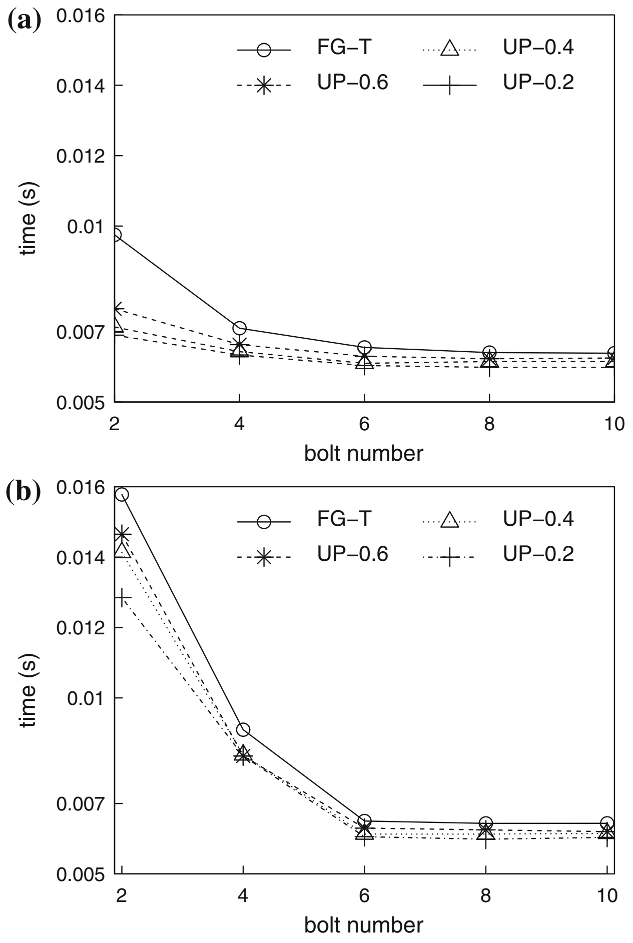


Fig. 17 Effect of partition. **a** D_{w}, **b** D_{syw}

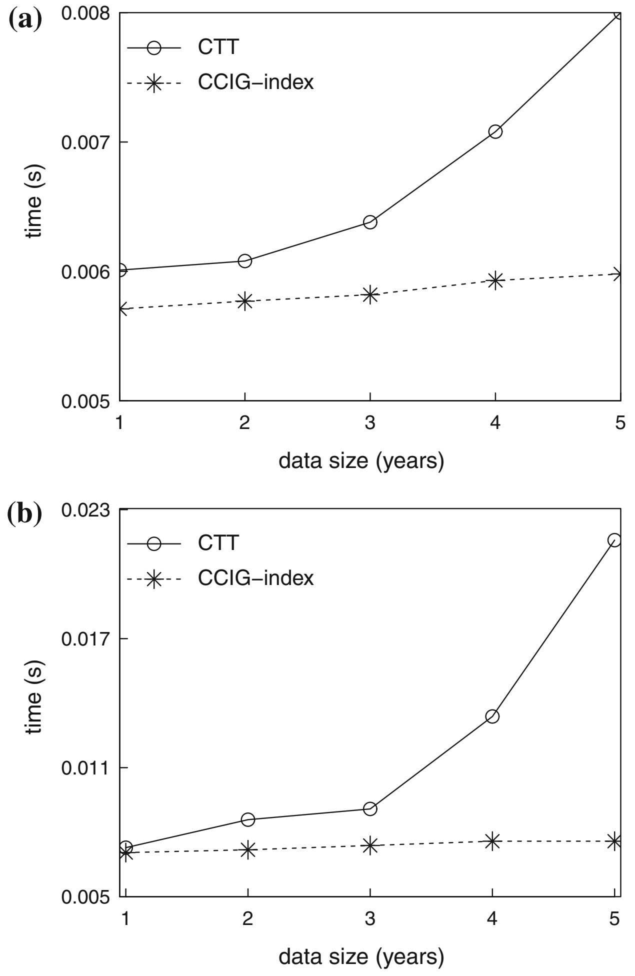


Fig. 18 Efficiency comparison. **a** D_{w}, **b** D_{syw}

ther smaller time cost. For another, with the fields grouping, the data distributed to some bolts may not contain any relevant media to a user, which makes the operations on them invalid. Thus, the processors cannot make full use of their capacity. With the bolt number increasing, the time cost of each approach decreases, because more processors jointly perform the search operations. When we allocate more than 6 bolts, the time cost of each method drops slightly with the further increase in bolts. This is caused by the data sparsity over each bolt, which makes the reduction in distance calculations smaller.

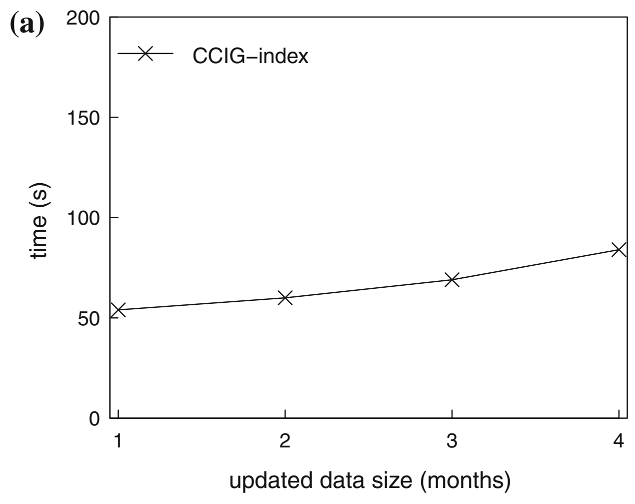
6.4.3 Efficiency comparison

We compare our recommender system with the state-of-the-art method CTT in terms of the average response time of each system over D_{w} and D_{syw}. We use 6 bolts following the setting in [3] and set the distance threshold of one-pass clustering to 0.2. We test the average total time cost of recommendations over 1–5 years time periods. The time costs of two methods are tested under their default settings as in 6.3.4. Their efficiency comparison is shown in Fig. 18.

Clearly, our approach is much faster than CTT, due to the efficient filtering over CCIG-index. With CCIG-index, the media data can be ranked by the Z-order values of their hash keys, and potentially relevant media can be quickly found for recommendation. Though CTT takes advantage of Apache Storm, all the media on each bolt have to be processed sequentially, leading to the slow response of each bolt. With the data size increase, the improvement of our approach is increased in terms of efficiency. This has proved the superiority of our CCIG-index over CTT under Apache Storm.

6.4.4 Efficiency of media updates

We test the cost of media updates over D_{w} and D_{syw} by changing the size of updates. Each of D_{w} and D_{syw} is divided into two parts: (1) a test set containing the media content and contexts appearing in recent 4 months (September–December 2016) and (2) a source set containing those



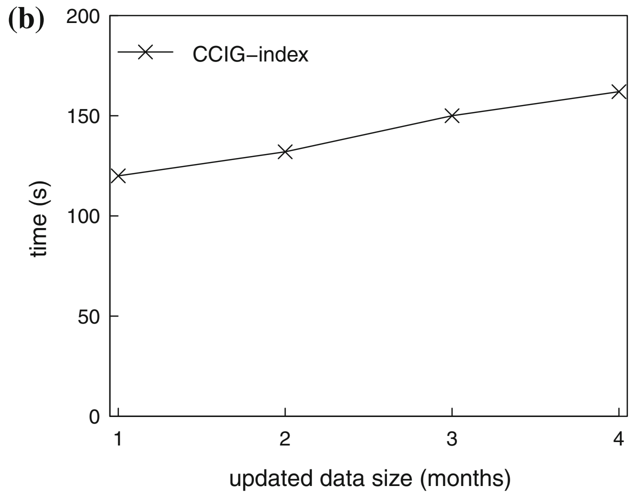


Fig. 19 Effect of updates on efficiency. **a** D_w, **b** D_{syw}

Table 9 CCIG construction time

Dataset	D_w	D_{syw}
Av. time (s)	2.3	2.4

appearing from January 2012 to August 2016. We fix our source sets and vary the test sets from 1 to 4 months updates. The time cost changes over different context updates are reported in Fig. 19. Clearly, the cost increases steadily with the update size increase. This is because we process the media with updates with the support of hash-based index under distributed environment. Thus, the update cost can be well controlled.

To further analyse the speed of media updates, we test the cost of CCIG construction for new incoming media data. Table 9 reports the average CCIG construction costs for each media data over two datasets. Clearly, the CCIG graph can be quickly constructed, thus suitable to online media updates.

7 Conclusion

This paper proposes a context-aware framework that effectively recommends social media to users in real time. First, we propose a new approach for removing the redundancy among a group of features with nonlinear correlations. Then, we propose a novel uniform content–context interaction graph model that fully fuses the content and contexts and captures the interactions between them. Finally, we propose a hash-based index under Apache Storm with advanced algorithms that uniformly allocate media over multiple processors and obtain top relevant media for online recommendation, and well maintain the social updates. The experimental results have verified the superiority of our recommender system in terms of efficacy.

Acknowledgements The work is partially supported by ARC project DP140100841, the Hong Kong RGC GRF Project 16207617, the National Science Foundation of China (NSFC) under Grants (No. 61729201, No.61332013, No.61572139), Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Huawei Co.Ltd Collaboration Project, YBCB2009041-45, Hong Kong ITC ITF grants ITS/391/15FX and ITS/212/16FP, and Microsoft Research Asia Collaborative Research Grant. The authors would like to thank Wenqiang Shao, Yang Li, Sheng Wang, and Liangjun Song for useful discussions.

References

1. http://tubularinsights.com/youtube-changes-33-percent-a-year/. Accessed 30 Nov 2015
2. Jiang, M., Cui, P., Liu, R., Yang, Q., Wang, F., Zhu, W., Yang, S.: Social contextual recommendation. In: CIKM. ACM, pp. 45–54 (2012)
3. Huang, Y., Cui, B., Jiang, J., Hong, K., Zhang, W., Xie, Y.: Real-time video recommendation exploration. In: SIGMOD, pp. 35–46 (2016)
4. Kumar, R., Verma, B.K., Rastogi, S.S.: Context-aware social popularity based recommender system. IJCA **92**(2), 37–42 (2014)
5. Yang, X., Steck, H., Liu, Y.: Circle-based recommendation in online social networks. In: KDD, pp. 1267–1275 (2012)
6. Yin, H., Cui, B., Chen, L., Hu, Z., Huang, Z.: A temporal context-aware model for user behavior modeling in social media systems. In: SIGMOD, pp. 1543–1554 (2014)
7. Akther, A., Alam, K.M., Kim, H.N., Saddik, A.E.: Social network and user context assisted personalization for recommender systems. In: IIT, pp. 95–100 (2012)
8. Zhou, X., Chen, L., Zhang, Y., Cao, L., Huang, G., Wang, C.: Online video recommendation in sharing community. In: SIGMOD. ACM, pp. 1645–1656 (2015)
9. Zhou, X., Chen, L., Zhang, Y., Qin, D., Cao, L., Huang, G., Wang, C.: Enhancing online video recommendation using social user interactions. VLDB J. **26**(5), 637–656 (2017)
10. Cui, B., Tung, A.K.H., Zhang, C., Zhao, Z.: Multiple feature fusion for social media applications. In: SIGMOD, pp. 435–446 (2010)
11. Meiri, R., Zahavi, J.: Using simulated annealing to optimize the feature selection problem in marketing applications. EJOR **171**(3), 842–858 (2006)
12. Broadhurst, D., Goodacre, R., Jones, A., Rowland, J., Kell, D.B.: Genetic algorithms as a method for variable selection in multiple

linear regression and partial least squares regression, with applications to pyrolysis mass spectrometry. Anal. Chim. Acta **348**(1), 71–86 (1997)
13. Kabir, M., Islam, M., Murase, K.: A new wrapper feature selection approach using neural network. Neurocomputing **73**(16), 3273–3283 (2010)
14. Wolf, L., Shashua, A.: Feature selection for unsupervised and supervised inference: the emergence of sparsity in a weight-based approach. JMLR **6**, 1855–1887 (2005)
15. Wang, X., Wang, Y., Wang, L.: Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recogn. Lett. **25**(10), 1123–1132 (2004)
16. Wang, J., Wu, L., Kong, J., Li, Y., Zhang, B.: Maximum weight and minimum redundancy: a novel framework for feature subset selection. Pattern Recogn. **46**(6), 1616–1627 (2013)
17. Song, Q., Ni, J., Wang, G.: A fast clustering-based feature subset selection algorithm for high-dimensional data. TKDE **25**(1), 1–14 (2013)
18. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: ICML, vol. 3, pp. 856–863 (2003)
19. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. JMLR **5**, 1205–1224 (2004)
20. Luo, H., Fan, J., Keim, D.A.: Personalized news video recommendation. In: ACM MM, pp. 1001–1002 (2008)
21. Yang, B., Mei, T., Hua, X., Yang, L., Yang, S., Li, M.: Online video recommendation based on multimodal fusion and relevance feedback. In: CIVR, pp. 73–80 (2007)
22. Zhu, Q., Shyu, M.L., Wang, H.: Videotopic: content-based video recommendation using a topic model. In: ISM, pp. 219–222 (2013)
23. Li, L., Wang, D., Li, T., Knox, D., Padmanabhan, B.: SCENE: a scalable two-stage personalized news recommendation system. In: SIGIR, pp. 125–134 (2011)
24. Rendle, S.: Factorization machines. In: ICDM, pp. 995–1000 (2010)
25. Cai, D., He, X., Han, J.: Tensor space model for document analysis. In: SIGIR. ACM, pp. 625–626 (2006)
26. Liu, N., Zhang, B., Yan, J., Chen, Z., Liu, W., Bai, F., Chien, L.: Text representation: from vector to tensor. In: ICDM, pp. 4–10 (2005)
27. Maulik, U., Bandyopadhyay, S.: Performance evaluation of some clustering algorithms and validity indices. TPAMI **24**(12), 1650–1654 (2002)
28. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: RecSys. ACM, pp. 43–50 (2008)
29. Symeonidis, P.: User recommendations based on tensor dimensionality reduction. In: Artificial Intelligence Applications and Innovations III. Springer, pp. 331–340 (2009)
30. Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. SIMAX **21**(4), 1253–1278 (2000)
31. Zhou, X., Chen, L., Zhou, X.: Structure tensor series-based large scale near-duplicate video retrieval. IEEE Trans. Multimed. **14**(4), 1220–1233 (2012)
32. https://en.wikipedia.org/wiki/Great-circle_distance. Accessed 14 Sept 2018
33. http://www.reelseo.com/youtube-300-hours/. Accessed 21 Nov 2014
34. Marzuki, Z., Ahmad, F.: Data mining discretization methods and performances. Lung **3**(32), 57 (2012)
35. Haindl, M., Somol, P., Ververidis, D., Kotropoulos, C.: Feature selection based on mutual correlation. In: Progress in Pattern Recognition, Image Analysis and Applications. Springer, pp. 569–577 (2006)
36. Chou, T.-S., Yen, K., Luo, J., Pissinou, N., Makki, K.: Correlation-based feature selection for intrusion detection design. In: MILCOM. IEEE, pp. 1–7 (2007)
37. Tao, Y., Yi, K., Sheng, C., Kalnis, P.: Quality and efficiency in high dimensional nearest neighbor search. In: SIGMOD, pp. 563–576 (2009)
38. Faloutsos, C., Lin, K.-I.: FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: ACM, vol. 24(2) (1995)
39. Schweikardt, N.: One-Pass Algorithm, pp. 1948–1949. Springer, Boston, MA (2009)
40. https://docs.google.com/spreadsheets/u/1/d/1s0okSI6Tcj4REgovRshrjULgQU14SF29Xv1TzLPqhU4/pub?gid=1. Accessed 1 Aug 2008
41. Nowok, B., Raab, G.M., Dibben, C.: synthpop: Bespoke creation of synthetic data in R. J. Stat. Softw. **74**, 1–26 (2015)
42. Debnath, S., Ganguly, N., Mitra, P.: Feature weighting in content based recommendation system using social network analysis. In: WWW, pp. 1041–1042 (2008)
43. http://storm.apache.org/releases/1.0.0/Concepts.html. Accessed 12 Apr 2016
44. https://trends.google.com/trends/explore?date=2012-01-01%202016-12-31&gprop=youtube&q=music,TV,art,sport,pet. Accessed 31 Dec 2016