Mignon: A Fast Decentralized Content Consumption Estimation in Large-Scale Distributed Systems

Stéphane Delbruel, Davide Frey, François Taïani

Mignon: a Fast Decentralized Content Consumption Estimation in Large-Scale Distributed Systems

Stéphane Delbruel[1], Davide Frey[2], and François Taïani[1]

[1] Université de Rennes 1, IRISA – ESIR, France
[2] Inria, Rennes, France
{stephane.delbruel, francois.taiani}@irisa.fr, {davide.frey}@inria.fr

Abstract. Although many fully decentralized content distribution systems have been proposed, they often lack key capabilities that make them difficult to deploy and use in practice. In this paper, we look at the particular problem of content consumption prediction, a crucial mechanism in many such systems. We propose a novel, fully decentralized protocol that uses the tags attached by users to on-line content, and exploits the properties of self-organizing kNN overlays to rapidly estimate the potential of a particular content without explicit aggregation.

Keywords: Decentralized systems, Content consumption, Estimation

1 Introduction

User-generated content (UGC) services have grown extremely fast over the last few years [1,37]. In order to support this growth, current services typically exploit private data centers owned by large companies such as Google, Sony and Amazon. These data centers are further augmented with Content Distribution Networks (CDNs) and caching servers positioned at points-of-presence (PoP) within the infrastructure of Internet Service Providers (ISPs) [16].

This approach tends to favor big players, and to concentrate the industry in the hands of a few powerful actors. For several years now, both academia and practitioners have therefore sought to explore alternative designs to implement social online services in general, and UGC video services in particular. One strategy espouses a fully decentralized organization [2,5,18,24,28], in which each individual user (through her computer or set-top box) provides resources to implement the system's overall services, including storage [24,29,30], indexing [9], queries [3,19], recommendation [4,5,6], caching [13], and streaming [8,14].

To ensure their scalability, most of these services primarily rely on *limited interactions* (e.g. with a small set of neighboring nodes) and *local information* (e.g. users profiles, bandwidth, latency, tags). The use of local information is one of the key reasons why these services scale. Too strong a focus on locality, however, constrains the range of decisions that can be taken by individual nodes, and their ability to adapt to phenomena occurring at a global scale.

In an attempt to address this limitation, we focus, in this paper, on the particular problem of *global predictions* in large-scale decentralized systems, with an application to the placement of videos in a decentralized UGC video service. Being able to predict where a new video is likely to be consumed is a crucial ability for decentralized services that often lack the tightly integrated global infrastructure of large players. It can help inform storage and caching decisions in order to best exploit the resources these services can rely on [33,32].

More precisely, we consider the problem of a newly uploaded videos that must be stored and replicated within a peer-to-peer system in the countries where it is more likely to be viewed. We have shown in a previous work that the tags attached to videos are a good predictor of a video's view distribution [11]. Unfortunately, individual peers do not by default have access to the past videos and tags consumed within individual countries, and this information can be costly to aggregate explicitly. In this paper, we therefore propose *Mignon*, a novel *decentralized content consumption estimation mechanism* that is fast and scalable and eschews the need for any global aggregation. Mignon exploits the properties of self-organizing similarity overlays [5,21,36] and delivers estimations that are on average within 0.6% (respectively 13%) of an exhaustive view aggregation on a MovieLens (respectively YouTube) dataset.

2 Problem Statement and Related Work

We consider a global decentralized P2P UGC service, in which each user contributes her resources to the system. As we focus on video placement and view prediction, we assume our service can store and retrieve videos from users' machines [31,34,29]. As is now common in many on-line services, we also assume that the past activity of users can be used to predict their affinity with new content (Fig. 1). More precisely, the individual devices of users (Alice and Bob, label **1**) store the list of videos they have consumed (their *video profile*, label **2**). Each video is associated with a set of descriptive tags provided by its uploading user [17,15] (label **3**). Here for instance, Alice has viewed a BBC video with the tag '*news*' (🌐), and a video on environmental protection with the tags '*news*', and '*animals*' (🌐 🐻). The tags of the videos viewed by a user form her *tag profiles* (label **4**): [🌐:2,🐻:1] for Alice and [🐻:1] for Bob.

We rely on a tag-based *affinity function* f that measures a user's affinity with new videos (**5**) [5,11]. The only assumptions we make about f is that its result is correlated with the probability that this user will watch the video (**6**).

2.1 Placing new videos: the prediction problem

When uploading a new video, copies of this video should ideally be placed in storage locations close to where it might be most consumed. This is because the viewing patterns of many videos in UGC services present clear geographic trends [7], which are strongly correlated with a video's tags [11]. Table 1 shows

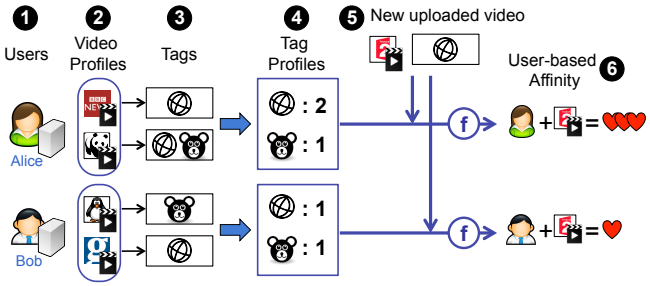
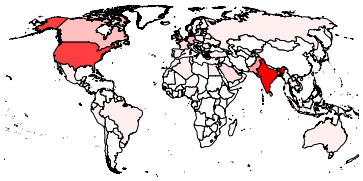


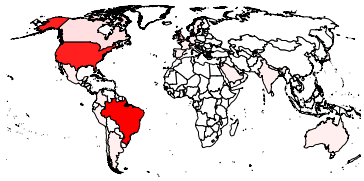
Fig. 1: Using tags to predict users' affinity with a new video

Table 1: Top 3 countries for *bollywood* (left) and *favela* (right)

country	#views	%age
India	200,956,055	39.8%
United-States	124,461,447	24.7%
United-Kingdom	29,506,586	5.8%

country	#views	%age
Brazil	19,834,633	47.9%
United-States	14,468,608	34.9%
United-Kingdom	1,701,496	4.1%





for instance how the tags "*bollywood*" and "*favela*" follow clearly distinctive geographic distributions in a Youtube dataset analyzed in an earlier work [11]. Correctly predicting the geographic distribution of a video's views is particularly important in decentralized systems that often lack the caching infrastructure of large integrated services. In Fig. 2 for instance, Dave must decide whether to store his new video in the USA or in France. This decision should be driven by the video's likely future popularity in both countries, which can be estimated as the sum of all user affinities in each country.

Obtaining this aggregated sum efficiently is unfortunately challenging in a large P2P system. Dave could trigger a P2P aggregation in the USA and France [27], but such an approach would require computing the similarity between the new video and every user in each country, a slow and costly operation.

In this paper, we therefore investigate how such a sum can be efficiently, rapidly, and accurately estimated in a fully decentralized system while involving only a small subset of the users in a given country.

2.2 Related work

A number of works have been proposed to perform aggregation operations in decentralized peer-to-peer systems [20,27]. These works typically use an epidemic procedure in which nodes repeatedly interact with other random peers in

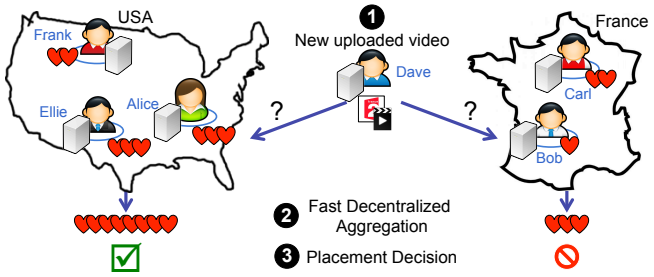


Fig. 2: Placing new videos based on aggregated affinity

a pair-wise fashion. They often further rely on a peer-sampling protocol [22,35] to maximize the diversity of interactions between peers. Following this strategy, averaging can for instance be implemented in the following manner: all peers p_i start with an initial value v_i^0. A given peer p_i then periodically selects another random peer p_j returned from the peer sampling service, and both peer update they respective value to $\frac{(v_i+v_j)}{2}$. This procedure guarantees that all nodes progressively converge to a value that is increasingly close to the average of all initial values $\frac{1}{N}\sum_{i=1}^{N} v_i$. The number of rounds required to attain a given aggregate accuracy primarily depends on the distribution of the original data [20].

This aggregation procedure can be used to estimate the size of a network, with all nodes but one starting with a value of 0, and one node (the initiator) a value of 1: all nodes will converge to a value of $\frac{1}{N}$ [27]. Combined with the above averaging protocol, such a size estimation can provide an estimate of the sum of the original peer values $\sum_{i=1}^{N} v_i$. Unfortunately, this approach is ill-suited to our case, as it would require the tags of every new video to be propagated to the entire network before any estimation may take place, incurring both additional latency and high network costs for every new load.

3 Fast Decentralized Sum Estimation

Instead of launching an expensive aggregation every time a new video is uploaded, we propose a cheaper mechanism to estimate the aggregated affinity of a video. Our approach exploits a similarity-driven overlay [5] that interconnects all the users in a country. In the following we first briefly describe similarity-driven overlays, and then present the details of our approach.

3.1 Self-organizing overlays

Similarity-driven overlay networks organize peers according to their similarity [21], with a wide range of applications [5,3,6,13,12]. In this work, we consider gossip-based similarity driven overlays, whose working is depicted in Figures 3-5. The machine of each user holds the user's profile: in our case the list of viewed videos and their attached tags (Fig. 3). Starting from random neighborhoods

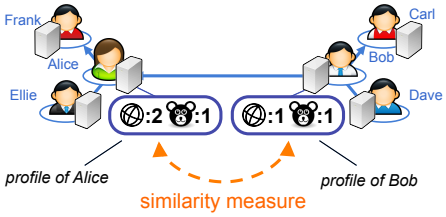


Fig. 3: A self-organizing overlay

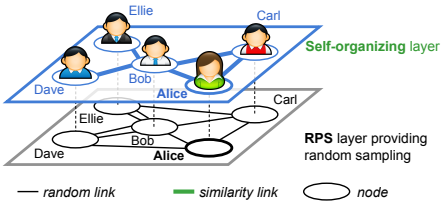


Fig. 4: Overlay Architecture

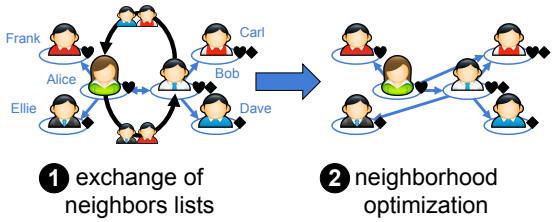


Fig. 5: Peer-to-peer neighborhood optimization

the overlay eventually connects each peer to its k most similar other peers in the network, according to some similarity metric (e.g. Jaccard's coefficient, or Cosine Similarity).

This construction uses two greedy mechanisms (Figures 4 and 5). With the first mechanism, a peer (e.g. Alice) regularly polls an underlying and constantly evolving Random Peer Sampling (RPS) overlay [22] to obtain a set of random peers from the rest of the system. In Fig. 4 for instance, Alice might discover Dave through the RPS layer. If Dave turns out to be a better neighbor for Alice than Bob (upper self-organizing layer), Alice will replace Bob by Dave in her neighborhood. This stochastic process ensures that the system eventually converges to an optimal state. The convergence might however be very slow.

To speed up convergence, peers use a second '*neighbor-of-neighbor*' mechanism (Fig. 5). The intuition is that if Alice is similar to Bob, and Bob to Carl, then Carl might be similar to Alice. Peers therefore periodically exchange their current neighbors lists (Step **1** in Fig. 5), and use the new peers they discover to optimize their neighborhoods (Step **2**). This mechanism greatly accelerates convergence (usually in $log(N)$ rounds [21]), but might get stuck in a local minimum, and is therefore complementary to the stochastic mechanism of Fig. 4.

3.2 Mignon: Fast Decentralized Estimation

In this paper we propose *Mignon*, a protocol that employs the similarity-driven overlay we have just described to estimate the aggregated affinity of a new video with all the users in a country. To this end, all the users in a country participate in a similarity-driven overlay whose similarity function is the affinity function

f of Fig. 1. When one of these users uploads a new video, v, she additionally creates a new virtual peer P_v, whose profile contains the tags associated with v.

Our estimation problem simply consists in computing the sum of the similarities between P_v and every other user in the country. To compute this sum exhaustively, either at peer P_v or using a standard aggregation protocol, we would either have to collect the profiles of all other nodes at P_v, or disseminate the profile of P_v to every other node. In both cases, the delay and the resulting network cost would be prohibitive for very large networks.

Instead, in Mignon, the uploading user simply impersonates the virtual peer by having it join the similarity-based overlay. In a very short time (generally logarithmic in the size of the network [21]), P_v obtains its k-nearest neighbors. Once this happens, the uploading user exploits the content of the KNN and RPS neighborhoods of P_v to estimate the video's aggregated affinity without any further network exchanges.

The key to the approach consists in considering the affinity values of users found in the KNN and RPS views of P_v as samples taken from a monotonically decreasing function. Fig. 6 shows this pictorially in two examples. The black vertical lines represent the affinity values of the users found in the KNN and RPS views of P_v. Mignon uses these values to interpolate the function's shape, from which we derive an aggregated affinity by integration. The values obtained from the KNN neighbors constitute the first k consecutive samples, while those in the RPS represent randomly chosen samples distributed along the rest of the x-axis. To associate each of them with an x-coordinate (which the RPS does not indicate), we rely on a network-size estimation protocol [25] that provides us with the length of the x-axis, and assume that the RPS samples are equally spaced along this axis.

It should be noted that the inherent cost of size-estimation does not offset the benefits provided by our approach in terms of delay and network cost. First, the size estimation protocol does not need to be run for every video upload. Rather, in a setup consisting of set-top boxes that are almost always on, the protocol can run every few days. Second, protocols like Sample & Collide [25] can estimate the size of the network within a reasonable error margin at a minimal cost. We evaluate the impact of protocols like Sample & Collide in Section 4.3. In the following we describe the two interpolation techniques we use in Mignon.

Trapezoidal rule. The first technique we consider is the trapezoidal rule, a well-known method for approximating the integral of a function. The rule replaces the function to be integrated with a sequence of linear segments and computes the integral as the sum of the areas of the corresponding trapezoids.

Polynomial Interpolation. As a second estimation mechanism, we consider a polynomial interpolation. Specifically, we compute the polynomial of degree $n-1$ that goes through all of the n samples in the KNN and RPS. We then use this polynomial to compute the values associated with the users that are not among the samples.

4 Evaluation

We evaluate Mignon on two distinct datasets. The first consists of an adaptation of the YouTube dataset we introduced in our previous work [10,18]. It contains 590,897 videos, each associated with a set of tags —11.18 per video on average, with a total of 705,415 distinct tags— and with a popularity vector that provides an estimated number of views per country. We extracted videos and tags directly from YouTube, while we computed the number of views for videos and tags by crossing YouTube data with information from Alexa Internet Inc.[3] as described in [10], with the following equation.

$$\mathbf{views}(v)[c] \simeq \frac{\widehat{\mathbf{p}}_{yt}[c] \times \mathbf{pop}(v)[c]}{\sum_{\gamma \in World} \left(\widehat{\mathbf{p}}_{yt}[\gamma] \times \mathbf{pop}(v)[\gamma]\right)} \times tot\_views(v) \tag{1}$$

Where $\mathbf{views}(v)[c]$ is the number of views of video v in country c, $\mathbf{p}_{yt}[c]$ is the proportion of Youtube views in country c at the time our data set was collected, and $\mathbf{pop}(v)[c]$ is a popularity vector issued from our ground hypothesis in [10], i.e. a number proportional to the share of video v's views in country c. To evaluate Mignon, we "reinterpreted" this dataset by considering each country as if it was a single user. Our modified dataset therefore consists of 257 users in a single country.

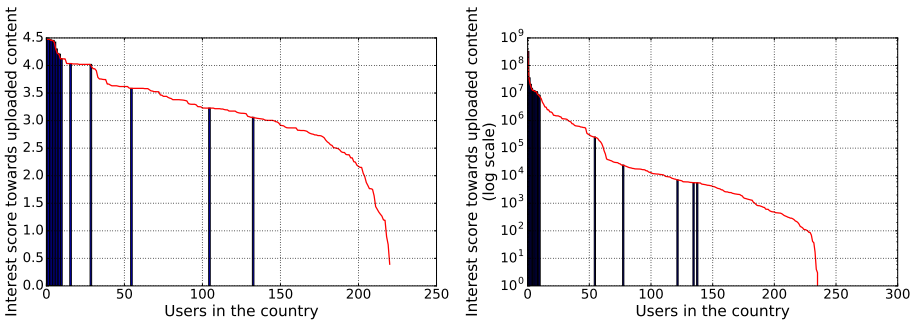
Our second dataset, MovieLens, consists of a trace from a movie recommendation system[4]. It contains a set of movies, each associated with a vector of ratings (1 to 5 integers) by a subset of the users, and a set of n pairs, each consisting of a tag and a real-valued relevance score. The rating $R_u(m)$ expresses the interest of a user u in movie m, while the relevance $r_m(t)$ score expresses the importance of a tag t for a given movie m. Based on this information, we compute the interest score u_t of a user u for a tag t as follows.

$$u_t = \frac{1}{n} \sum_{m=1}^{n} (r_m(t) * R_u(m)) \tag{2}$$

Since we want to evaluate Mignon's ability to estimate the aggregation of a score value, we consider a synthetic set of new "videos", whose profile only comprises a single tag taken from the dataset. For each such video v, we first select the set of users in its KNN and RPS views, and then compute its affinity with these users. We use this sample of affinity values to produce an estimate (noted $\hat{a}_v$) of the video's aggregated affinity with all the users in the system (which we note a_v). To assess the performance of different estimation techniques, we define an estimation ratio: $\text{ER}_v = \frac{\hat{a}_v}{a_v}$. We evaluate ER_v in a variety of configurations on each of our datasets. Let n be the number of tags in a dataset (and hence of synthetic videos), we present the distribution of ER_v, its mean $\overline{\text{ER}} = \frac{1}{n}\sum_{i=1}^{n} \text{ER}_{v_i}$, as well as its standard deviation $\sqrt{\overline{\text{ER}^2} - \overline{\text{ER}}^2}$.

[3] http://www.alexa.com/siteinfo/youtube.com

[4] www.movielens.org



(a) MovieLens (b) Youtube

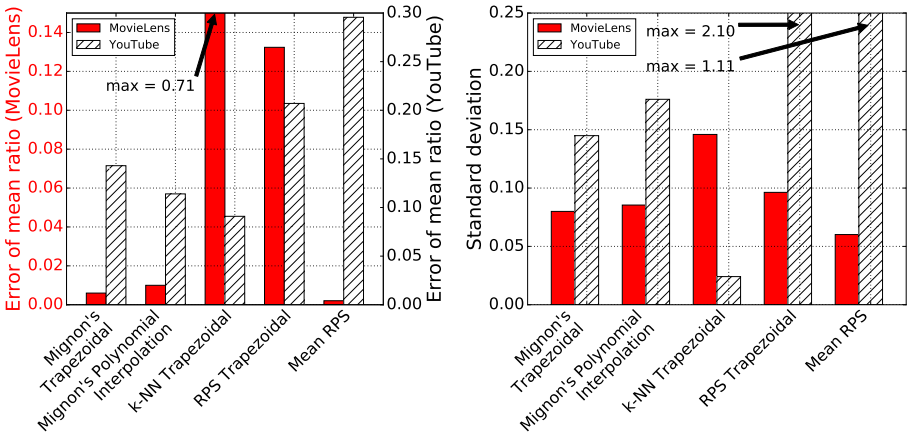
Fig. 6: Interest curve for MovieLens(a) and YouTube(b) datasets. Black vertical lines represent KNN and RPS samples.

Figure 6 exemplifies the affinity score distribution of particular tags (interpreted as videos) in each of the two dataset. The curve depicts the affinity score of each user for the tag in decreasing order, while the vertical bars represent the data available in the KNN and RPS views.

4.1 Accuracy Comparison

We start our evaluation by comparing the results obtained by Mignon with those obtained by three baseline approaches that exploit either the KNN or the RPS views but not both. For Mignon, we consider the two estimation techniques presented in Section 3.2 (the *Trapezoidal* and *Polynomial* interpolations). For the baselines, we tested both these techniques as well as linear and quadratic regression and selected the three that obtained the best performance. Specifically, **KNN-Trapezoid** applies the trapezoid rule on a KNN view without using the RPS, **RPS-Trapezoid** also applies the trapezoid rule but on an RPS view with no KNN, while **RPS-Mean** simply computes the average similarity of the nodes in the RPS view and multiplies it by the size of the network. We configured our techniques to use a KNN view size of 15 and an RPS size of 10, while all the baselines use a single view (RPS or KNN) of size 25.

Figure 7 shows the results on both of our datasets. Figure 7a depicts the the error on the mean estimation ratio, that is $|\overline{\mathrm{ER}} - 1|$, and shows that combining the KNN and the RPS views allows Mignon to adapt to multiple data sets. Specifically, both the Trapezoidal rule and Polynomial interpolation obtain very good estimates on both datasets with an error on the mean ratio respectively of 0.06 (6%) and 0.01 (1%) on MovieLens and of 0.143 (14.3%) and 0.114 (11.4%) on YouTube. The baselines, on the other hand, can achieve good performance on one of the datasets but not on both. KNN-Trapezoid achieves a very low error of 0.09 (9%) on YouTube, but a very high error of 0.7 (70%) on MovieLens. RPS-Mean achieves a very low error of 0.02 (2%) on MovieLens but a high error



(a) Error of the mean ratio

(b) Standard deviation

Fig. 7: Evaluation of the error and the standard deviation for both datasets MovieLens and YouTube

of 0.30 (30%) on YouTube, while RPS-Trapezoid achieves errors of 0.13 (13%) on MovieLens and of 0.21 (21%) on YouTube, worse than both of Mignon's approaches on both datasets.

Figure 7b completes the picture by showing the standard deviation of the estimation ratio. Again, Mignon obtains low standard deviations on both data sets, contrary to RPS-Trapezoid and RPS-Mean. KNN-Trapezoid also achieves good standard deviations on both dataset, but with a very high mean error on MovieLens (Figure 7a).

4.2 Sensitivity Analysis

Now that we have shown the effectiveness of Mignon's estimation approach on multiple datasets, we analyze how the KNN and RPS views impact its performance. We present our results in the form of whisker plots in Figures 8 and 9. Each box in the plot covers the values between the lower and the upper quartiles; the point in the box represents the mean, while the line the median. The endpoints of the whiskers represent the lowest datum still within 1.5∗InterQuartile Range (IQR) of the lower quartile, and the highest datum still within 1.5∗IQR of the upper quartile, while the points outside the whiskers represent outliers.

Trapezoidal rule. Figure 8 shows how the effectiveness of the trapezoid rule varies when we vary the sizes of the KNN and RPS views. For fairness we maintain a total view size of 25 and vary the proportion of nodes in the two views from |KNN|=2 |RPS|=23 to |KNN|=23 |RPS|=2. Figure 8a shows that larger KNN

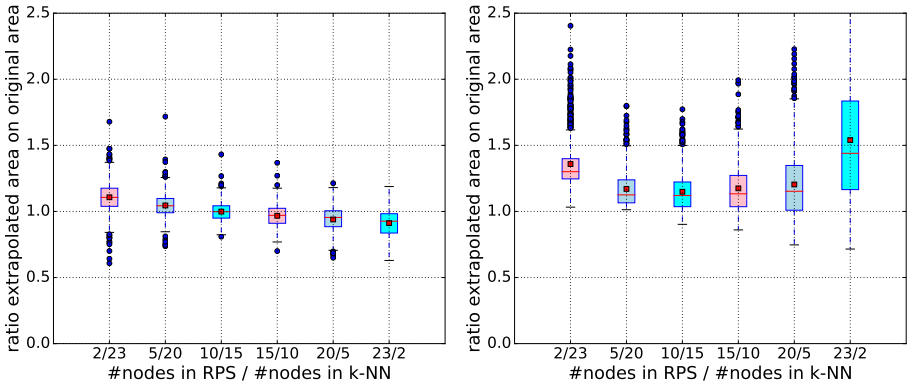


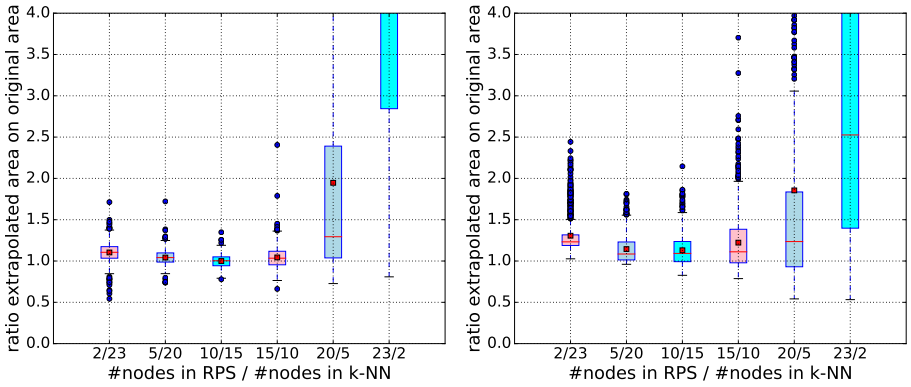
Fig. 8: Fast Decentralized Area Estimation using the trapezoid rule in the Movielens dataset(a) and YouTube dataset(b).

views slightly tend to overestimate the total affinity, while large RPS views slightly tend to underestimate it, with the best performance being achieved with a KNN view of 15 and an RPS view of 10. Additional tests (results not shown for space reason) showed that this results primarily from the size of the RPS view. Varying the KNN size with a constant RPS size has almost no impact, while varying the RPS size with a constant KNN size results in overestimation with few RPS nodes and in underestimation with too many RPS nodes.

Figure 8b complements the above results with the performance of the Trapezoid rule on the YouTube dataset. Again, we obtain the best performance with a KNN-to-RPS ratio of 3/2. With a KNN view of 15 and an RPS view of 10, the mean estimation ratio settles at 1.14. Moreover, slightly smaller or slightly larger KNN-to-RPS ratios impact this result only to a limited extent. In our tests, we observed that this results from the fact that when one view remains constant, performance consistently improves when increasing the size of the other.

Polynomial interpolation. Next, we evaluate the effectiveness of Mignon using polynomial interpolation. To this end, we used the Gregory-Newton interpolation algorithm as implemented in SciPy. Figure 9 shows the results. Both datasets exhibit similar behaviors. For low RPS sizes, results resemble those obtained with the trapezoid rule, with the best performance being achieved with an RPS of 10 and a KNN of 15. However, results start diverging as soon as the RPS size goes beyond 15. We experimentally verified that this also occurs when increasing the RPS size with a constant KNN size, but not when increasing the KNN size with a constant RPS size.

To understand the high variability associated with high RPS sizes, we examine two runs of the Gregory-Newton interpolation algorithm in Figure 10.



(a) (b)

Fig. 9: Fast Decentralized Area Estimation using Polynomial interpolation in the Movielens dataset(a) and YouTube dataset(b).

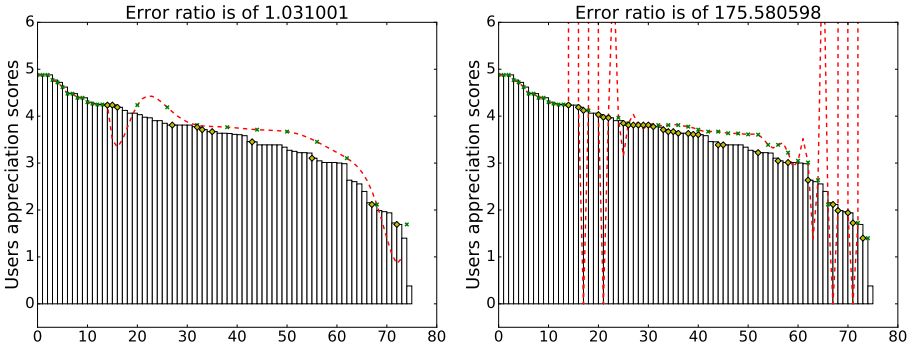
Figure 10a shows a run with 10 RPS nodes, while Figure 10b shows one with 30. In both figures, the diamonds represent the real abscissas of the samples on the curve, while the crosses represent those taken into account by our protocol (see Section 3.2). For KNN samples, the two coincide (points at the extreme left of the curve), but for the RPS the difference can be very large. This, together with the numerical instability of the Gregory-Newton's method causes oscillations at the right end of the curve. Some oscillations are visible even with an RPS of 10. But with an RPS of 30, they completely disrupt the estimation.

4.3 Influence of Sample & Collide

We now assess the impact of errors on the network-size estimation. As previously stated, nodes do not need to recompute the size of the network for every new upload as we assume the network to be relatively stable. Nonetheless, it is possible to limit the cost of size estimation by means of protocols like Sample & Collide [26]. Such a protocol yields an estimate with a 10% error at a very limited network cost. We estimate the impact of this error in Table 2 where we shows the absolute value of the error on the mean estimation ratio for both Mignon's approaches in the presence of a positive or negative error on the estimation size. The data shows that the error on the network size has almost no impact on YouTube, and a relatively low one on MovieLens.

4.4 Convergence speed

We conclude by evaluating the time required to compute the estimate using Mignon. First, let us consider a baseline system that would simply compute the



(a) RPS 10 (b) RPS 30

Fig. 10: Details of the Gregory-Newton interpolation with different RPS sizes in the Movielens dataset.

Error	0%	+10%	-10%
MovieLens	-0.8%	+8.8%	-11%
YouTube	+12.4%	+14.9%	+8.7%

(a)

Error	0%	+10%	-10%
MovieLens	-0.6%	+8.9%	-11.1%
YouTube	+14.3%	+10.4%	+17%

(b)

Table 2: Mean error percentage for various size-estimation errors, for Polynomial interpolation(a) and Trapezoidal rule(b).

sum of the affinities of the uploaded video with all the other nodes in the country. Such a system would either require the uploading node to contact each other node in the country to compute its affinity, or it would have to disseminate the video's profile so that other nodes could evaluate the video's affinity with them. Both of these approaches would clearly be difficult to scale to large numbers of nodes and their convergence time would be comparable, if not worse, than that required by a KNN protocol to converge from a completely random configuration.

Mignon, on the other hand, takes advantage of the presence of an already converged KNN protocol. This overlay allows the uploading node to quickly reach its closest neighbors. To evaluate this difference, we counted the number of gossip cycles required by a KNN protocol to reach convergence from scratch with 6000 nodes. In each cycle, a node contacts one other node, and is, on average, contacted by another one. We then added one random node, and counted the cycles it took to reach convergence again. Convergence from scratch took between 150 and 190 gossip cycles, while convergence after adding a node to an already converged network took an order of magnitude less $(10 - 20)$.

5 Conclusion

In this paper, we have proposed *Mignon*, a new protocol to rapidly estimate the aggregate affinity of a newly uploaded video in a community of users in a fully decentralized manner. Our proposal avoids an explicit and costly aggregation by relying on the properties of similarity-based self-organizing overlay networks, and can be used to decide where to place videos in a decentralized UGC system. By eschewing the need for a central support infrastructure, our approach hints at the possibility of fast reactive aggregate analytics in decentralized systems. This may be useful both to promote alternatives to the cloud-centered model of current UGC video services, but also to improve hybrid P2P/cloud architectures [23,38] by offloading complex adaptive tasks to the P2P part of a hybrid system.

References

1. Global internet phenomena report: 2h 2013. Technical report, Sandvine Incorporated, 2013.
2. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, Dec. 2004.
3. X. Bai, M. Bertier, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. Gossiping personalized queries. In *EDBT*, 2010.
4. R. Baraglia, P. Dazzi, M. Mordacchini, and L. Ricci. A peer-to-peer recommender system for self-emerging user communities based on gossip overlays. *J. of Comp. and Sys. Sciences*, 2013.
5. M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Middleware*, 2010.
6. A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. WhatsUp Decentralized Instant News Recommender. In *IPDPS*, 2013.
7. A. Brodersen, S. Scellato, and M. Wattenhofer. YouTube around the world: Geographic popularity of videos. In *WWW*, 2012.
8. M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *IMC*, 2007.
9. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS*, 2002.
10. S. Delbruel, D. Frey, and F. Taïani. Exploring the Geography of Tags in Youtube Views. Research Report RT-0461, IRISA, Inria Rennes, Apr. 2015.
11. S. Delbruel, D. Frey, and F. Taïani. Exploring the use of tags for georeplicated content placement. In *IC2E*, 2016.
12. M. El Dick, E. Pacitti, and B. Kemme. Flower-cdn: a hybrid p2p overlay for efficient query processing in cdn. In *EDBT '09*, pages 427–438. ACM, 2009.
13. D. Frey, M. Goessens, and A.-M. Kermarrec. Behave: Behavioral Cache for Web Content. In *DAIS*, 2014.
14. D. Frey, R. Guerraoui, A.-M. Kermarrec, B. Koldehofe, M. Mogensen, M. Monod, and V. Quéma. Heterogeneous gossip. In *Middleware*. 2009.
15. G. Geisler and S. Burns. Tagging video: conventions and strategies of the youtube community. In *ACM/IEEE-CS Joint Conf. on Digital Libraries*, 2007.
16. Google Inc. Google peering and content delivery . https://peering.google.com/about/ggc.html. (accessed 2/5/2015).

17. S. Greenaway, M. Thelwall, and Y. Ding. Tagging youtube - a classification of tagging practice on youtube. In *Int. Conf. on Sciento- & Informetrics*, 2009.
18. K. Huguenin, A.-M. Kermarrec, K. Kloudas, and F. Taïani. Content and geographical locality in user-generated content sharing systems. In *NOSSDAV*, 2012.
19. S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2P-diet: An extensible P2P service that unifies ad-hoc and continuous querying in super-peer networks. In *SIGMOD*, pages 933–934. ACM, 2004.
20. M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. 2004.
21. M. Jelasity, A. Montresor, and O. Babaoglu. T-man: Gossip-based fast overlay topology construction. *Comput. Netw.*, 53(13):2321–2339, Aug. 2009.
22. M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM TOCS*, 25, 2007.
23. G. Kreitz and F. Niemelä. Spotify – large scale, low latency, P2P music-on-demand streaming. In *P2P*, 2010.
24. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM Sigplan Notices*, 35(11):190–201, 2000.
25. E. Le Merrer, A.-M. Kermarrec, and L. Massoulie. Peer to peer size estimation in large and dynamic networks: A comparative study. In *HPDC*, 2006.
26. L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: Random walk methods. In *PODC*, 2006.
27. A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *DSN*, 2004.
28. J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. In *SIGCOMM*, 2010.
29. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, 2001.
30. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.
31. S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia systems*, 9(2):170–184, 2003.
32. N. Sastry, E. Yoneki, and J. Crowcroft. Buzztraq: Predicting geographical access patterns of social cascades using social networks. In *SNS*, 2009.
33. S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: Improving content delivery networks by tracking geographic social cascades. In *WWW*, 2011.
34. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160, 2001.
35. S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.
36. S. Voulgaris and M. v. Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par'05*.
37. Youtube, LCC. Statistics, viewership . http://www.youtube.com/yt/press/statistics.html. (accessed 2/5/2015).
38. M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec. Peer-assisted content distribution in Akamai NetSession. In *IMC*, 2013.