# The YouTube Video Recommendation System

James Davidson
Google Inc
davidson@google.com

Benjamin Liebald
Google Inc
liebald@google.com

Junning Liu
Google Inc
ljn@google.com

Palash Nandy
Google Inc
palash@google.com

Taylor Van Vleet
Google Inc
tvv@google.com

## ABSTRACT

We discuss the video recommendation system in use at YouTube, the world's most popular online video community. The system recommends personalized sets of videos to users based on their activity on the site. We discuss some of the unique challenges that the system faces and how we address them. In addition, we provide details on the experimentation and evaluation framework used to test and tune new algorithms. We also present some of the findings from these experiments.

## Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; H.4 [**Information Systems**]: Information Systems Applications

## General Terms

Algorithms, Measurement

## 1. INTRODUCTION

Personalized recommendations are a key method for information retrieval and content discovery in today's information-rich environment. Combined with pure search (querying) and browsing (directed or non-directed), they allow users facing a huge amount of information to navigate that information in an efficient and satisfying way. As the largest and most-popular online video community with vast amounts of user-generated content, YouTube presents some unique opportunities and challenges for content discovery and recommendations.

Founded in February 2005, YouTube has quickly grown to be the world's most popular video site. Users come to YouTube to discover, watch and share originally-created videos. YouTube provides a forum for people to engage with video content across the globe and acts as a distribution platform for content creators. Every day, over a billion video plays are done across millions of videos by millions of users, and every minute, users upload more than 24 hours of video to YouTube.

In this paper, we present our video recommendation system, which delivers personalized sets of videos to signed in users based on their previous activity on the YouTube site (while recommendations are also available in a limited form to signed out users, we focus on signed in users for the remainder of this paper). Recommendations are featured in two primary locations: The YouTube home page (`http://www.youtube.com`) and the "Browse" page at `http://www.youtube.com/videos`. An example of how recommendations are presented on the homepage can be found in Figure 1.

### 1.1 Goals

Users come to YouTube for a wide variety of reasons which span a spectrum from more to less specific: To watch a single video that they found elsewhere (*direct navigation*), to find specific videos around a topic (*search* and *goal-oriented browse*), or to just be entertained by content that they find interesting. Personalized Video Recommendations are one way to address this last use case, which we dub *unarticulated want*.

As such, the goal of the system is to provide personalized recommendations that help users find high quality videos relevant to their interests. In order to keep users entertained and engaged, it is imperative that these recommendations are updated regularly and reflect a user's recent activity on the site. They are also meant to highlight the broad spectrum of content that is available on the site.

In its present form, our recommendation system is a top-N recommender rather than a predictor [4]. We review how we evaluate the success of the recommendation system in section 3 of this paper. An additional primary goal for YouTube recommendations is to maintain user privacy and provide explicit control over personalized user data that our back-end systems expose. We review how we address this goal in section 2.5.

### 1.2 Challenges

There are many aspects of the YouTube site that make recommending interesting and personally relevant videos to users a unique challenge: Videos as they are uploaded by users often have no or very poor metadata. The video corpus size is roughly on the same order of magnitude as the number of active users. Furthermore, videos on YouTube are mostly short form (under 10 minutes in length). User interactions are thus relatively short and noisy. Compare this to user interactions with movie rental or purchase sites


*RecSys2010,* September 26–30, 2010, Barcelona, Spain.


**Figure 1: A screenshot of the recommendations module on the YouTube home page**

such as Netflix or Amazon where renting a movie or purchasing an item are very clear declarations of intent. In addition, many of the interesting videos on YouTube have a short life cycle going from upload to viral in the order of days requiring constant freshness of recommendation.

## 2. SYSTEM DESIGN

The overall design of the recommendation system is guided by the goals and challenges outlined above: We want recommendations to be reasonably recent and fresh, as well as diverse and relevant to the user's recent actions. In addition, it's important that users understand why a video was recommended to them.

The set of recommended videos videos is generated by using a user's personal activity (watched, favorited, liked videos) as seeds and expanding the set of videos by traversing a co-visitation based graph of videos. The set of videos is then ranked using a variety of signals for relevance and diversity.

From an engineering perspective, we want individual components of the system to be decoupled from each other, allowing them to be understood and debugged in isolation. Given that our system is part of the larger YouTube ecosystem, recommendations also needs to be resilient to failure and degrade gracefully in case of partial failures. As a consequence, we strive to minimize complexity in the overall system.

### 2.1 Input data

During the generation of personalized video recommendations we consider a number of data sources. In general, there are two broad classes of data to consider: 1) content data, such as the raw video streams and video metadata such as title, description, etc, and 2) user activity data, which can further be divided into explicit and implicit categories. Explicit activities include rating a video, favoriting/liking a video, or subscribing to an uploader. Implicit activities are datum generated as a result of users watching and interacting with videos, e.g., user started to watch a video and user watched a large portion of the video (long watch).

In all cases, the data that we have at our disposal is quite noisy: Video metadata can be non-existent, incomplete, outdated, or simply incorrect; user data only captures a fraction of a user's activity on the site and only indirectly measures a user's engagement and happiness, e.g., the fact that a user watched a video in its entirety is not enough to conclude that she actually liked it. The length of the video and user engagement level all influence the signal quality. Moreover, implicit activity data is generated asynchronously and can be incomplete, e.g., the user closes the browser before we receive a long-watch notification.

### 2.2 Related Videos

One of the building blocks of the recommendation system is the construction of a mapping from a video $v_i$ to a set of similar or *related* videos $R_i$. In this context, we define similar videos as those that a user is likely to watch after having watched the given *seed video* $v$. In order to compute the mapping we make use of a well-known technique known as association rule mining [1] or co-visitation counts. Consider sessions of user watch activities on the site. For a given time period (usually 24 hours), we count for each pair of videos $(v_i, v_j)$ how often they were co-watched within sessions. Denoting this co-visitation count by $c_{ij}$, we define the *relatedness score* of video $v_j$ to base video $v_i$ as:

$$r(v_i, v_j) = \frac{c_{ij}}{f(v_i, v_j)} \tag{1}$$

where $c_i$ and $c_j$ are the total occurrence counts across all sessions for videos $v_i$ and $v_j$, respectively. $f(v_i, v_j)$ is a normalization function that takes the "global popularity" of both the seed video and the candidate video into account. One of the simplest normalization functions is to simply divide by the product of the videos' global popularity: $f(v_i, v_j) = c_i \cdot c_j$. Other normalization functions are possible. See [6] for an overview of possible choices. When using the simple product of cardinalities for normalization, $c_i$ is the same for all candidate related videos and can be ignored in our setting, so we are normalizing only by the candidate's global popularity. This essentially favors less popular videos over popular ones.

We then pick the set of related videos $R_i$ for a given seed video $v_i$ as the top $N$ candidate videos ranked by their scores $r(v_i, v_j)$. Note that in addition to only picking the top N videos, we also impose a minimum score threshold. Hence, there are many videos for which we will not be able to compute a reliable set of related videos this way because their overall view count (and thereby co-visitation counts with other videos) is too low.

Note that this is a simplified description. In practice there are additional problems that need to be solved—presentation bias, noisy watch data, etc.—and additional data sources beyond co-visitation counts that can be used: sequence and time stamp of video watches, video metadata, etc.

The related videos can be seen as inducing a directed graph over the set of videos: For each pair of videos $(v_i, v_j)$, there is an edge $e_{ij}$ from $v_i$ to $v_j$ iff $v_j \in R_i$, with the weight of this edge given by (1).

## 2.3 Generating Recommendation Candidates

To compute personalized recommendations we combine the related videos association rules with a user's personal activity on the site: This can include both videos that were watched (potentially beyond a certain threshold), as well as videos that were explicitly favorited, "liked", rated, or added to playlists. We call the union of these videos the *seed set*.

In order to obtain candidate recommendations for a given seed set $S$, we expand it along the edges of the related videos graph: For each video $v_i$ in the seed set consider its related videos $R_i$. We denote the union of these related video sets as $C_1$:

$$C_1(S) = \bigcup_{v_i \in S} R_i \tag{2}$$

In many cases, computing $C_1$ is sufficient for generating a set of candidate recommendations that is large and diverse enough to yield interesting recommendations. However, in practice the related videos for any videos tend to be quite narrow, often highlighting other videos that are very similar to the seed video. This can lead to equally narrow recommendations, which do achieve the goal of recommending content close to the user's interest, but fail to recommend videos which are truly new to the user.

In order to broaden the span of recommendations, we expand the candidate set by taking a limited transitive closure over the related videos graph. Let $C_n$ be defined as the set of videos reachable within a distance of $n$ from any video in the seed set:

$$C_n(S) = \bigcup_{v_i \in C_{n-1}} R_i \tag{3}$$

where $C_0 = S$ is the base case for the recursive definition (note that this yields an identical definition for $C_1$ as equation (2)). The final candidate set $C_{final}$ of recommendations is then defined as:

$$C_{final} = (\bigcup_{i=0}^{N} C_i) \setminus S \tag{4}$$

Due to the high branching factor of the related videos graph we found that expanding over a small distance yielded a broad and diverse set of recommendations even for users with a small seed set. Note that each video in the candidate set is associated with one or more videos in the seed set. We keep track of these seed to candidate associations for ranking purposes and to provide explanations of the recommendations to the user.

## 2.4 Ranking

After the generation step has produced a set of candidate videos they are scored and ranked using a variety of signals. The signals can be broadly categorized into three groups corresponding to three different stages of ranking: 1) video quality, 2) user specificity and 3) diversification.

Video quality signals are those signals that we use to judge the likelihood that the video will be appreciated irrespective of the user. These signals include view count (the total number of times a video has been watched), the ratings of the video, commenting, favoriting and sharing activity around the video, and upload time.

User specificity signals are used to boost videos that are closely matched with a user's unique taste and preferences. To this end, we consider properties of the seed video in the user's watch history, such as view count and time of watch.

Using a linear combination of these signals we generate a ranked list of the candidate videos. Because we display only a small number of recommendations (between 4 and 60), we have to choose a subset of the list. Instead of choosing just the most relevant videos we optimize for a balance between relevancy and diversity across categories. Since a user generally has interest in multiple different topics at differing times, videos that are too similar to each other are removed at this stage to further increase diversity. One simple way to achieve this goal is to impose constraints on the number of recommendations that are associated with a single seed video, or by limiting the number of recommendations from the same channel (uploader). More sophisticated techniques based on topic clustering and content analysis can also be used.

## 2.5 User Interface

Presentation of recommendations is an important part of the overall user experience. Figure 1 shows how recommendations are currently presented on YouTube's home page. There are a few features worth noting: First, all recommended videos are displayed with a thumbnail and their (possibly truncated) title, as well as information about video age and popularity. This is similar to other sections on the homepage and helps users decide quickly whether they are interested in a video. Furthermore, we add an *explanation* with a link to the seed video which triggered the recommendation. Last, we give users control over where and how many recommendations they want to see on the homepage.

As mentioned in section 2.4, we compute a ranked list of recommendations but only display a subset at serving time. This enables us to provide new and previously unseen recommendations every time the user comes back to the site, even if the underlying recommendations have not been recomputed.

## 2.6 System Implementation

We choose a batch-oriented pre-computation approach rather than on-demand calculation of recommendations. This has the advantages of allowing the recommendation generation stage access to large amounts of data with ample amounts of CPU resources while at the same time allowing the serving of the pre-generated recommendations to be extremely low latency. The most significant downside of this approach is the delay between generating and serving a particular recommendation data set. We mitigate this by pipelining the recommendation generation, updating the data sets several times per day.

The actual implementation of YouTube's recommendation system can be divided into three main parts: 1) data collection, 2) recommendation generation and 3) recommendation serving.

The raw data signals previously mentioned in section 2.1 are initially deposited into YouTube's logs. These logs are processed, signals extracted, and then stored on a per user

basis in a Bigtable [2]. We currently handle millions of users and tens of billions of activity events with a total footprint of several terabytes of data.

Recommendations are generated through a series of MapReduces computations [3] that walk through the user/video graph to accumulate and score recommendations as described in section 2.

The generated data set sizes are relatively small (on the order of Gigabytes) and can be easily served by simplified read-only Bigtable servers to YouTube's webservers; the time to complete a recommendation request is mostly dominated by network transit time.

## 3. EVALUATION

In our production system we use live evaluation via A/B testing [5] as the main method for evaluating the performance of the recommendation system. In this method, live traffic is diverted into distinct groups where one group acts as the control or baseline and the other group is exposed to a new feature, data, or UI. The two groups are then compared against one another over a set of predefined metrics and possibly swapped for another period of time to eliminate other factors. The advantage of this approach is that evaluation takes place in the context of the actual website UI. It's also possible to run multiple experiments in parallel and get quick feedback on all of them. The downsides are that not all experiments have reasonable controls that can be used for comparison, the groups of users must have sufficient traffic to achieve statistically significant results in a timely manner and evaluation of subjective goals is limited to the interpretation of a relatively small set of pre-defined metrics.

To evaluate recommendation quality we use a combination of different metrics. The primary metrics we consider include click through rate (CTR), long CTR (only counting clicks that led to watches of a substantial fraction of the video), session length, time until first long watch, and recommendation coverage (the fraction of logged in users with recommendations). We use these metrics to both track performance of the system at an ongoing basis as well as for evaluating system changes on live traffic.

## 4. RESULTS

The recommendations feature has been part of the YouTube homepage for more than a year and has been very successful in context of our stated goals. For example, recommendations account for about 60% of all video clicks from the home page.

Comparing the performance of recommendations with other modules on the homepage suffers from presentation bias (recommendations are placed at the top by default). To adjust for this, we look at CTR metrics from the "browse" pages and compare recommendations to other algorithmically generated video sets: a) Most Viewed - Videos that have received the most number of views in a day, b) Top Favorited - Videos that the viewers have added to their collection of favorites and c) Top Rated - Videos receiving most like ratings in a day.

We measured CTR for these sections over a period of 21 days. Overall we find that co-visitation based recommendation performs at 207% of the baseline Most Viewed page when averaged over the entire period, while Top Favorited and Top Rated perform at similar levels or below the Most Viewed baseline. See figure 2 for an illustration of how the relative CTR varies over the period of 3 weeks.

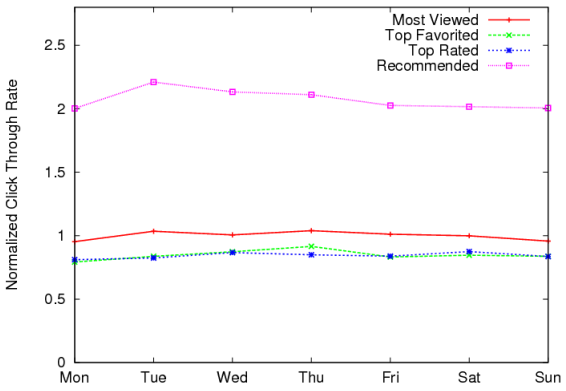


**Figure 2: Per-day average CTR for different browse page types over a period of 3 weeks.**

## 5. ACKNOWLEDGMENTS

We would like to thank John Harding, Louis Perrochon and Hunter Walk for support and comments.

## 6. ADDITIONAL AUTHORS

Additional authors: Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, Dasarathi Sampath (all Google Inc, emails {ullas, sujoy, yuhe, lambert, blivingston, dasarathi}@google.com).

## 7. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *USENIX '07*, pages 205–218, 2006.

[3] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI '04*, pages 137–150, 2004.

[4] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[5] S. Huffman. Search evaluation at Google. `http://googleblog.blogspot.com/2008/09/search-evaluation-at-google.html`, 2008.

[6] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *KDD '05*, pages 678–684, New York, NY, USA, 2005. ACM.