A Simple and Efficient Method to Handle Sparse Preference Data Using Domination Graphs: An Application to YouTube

Shumeet Baluja

Google, Inc.
shumeet@google.com

Abstract
The phenomenal growth of the number of videos on YouTube provides enormous potential for users to find content of interest to them. Unfortunately, as the size of the repository grows, the task of discovering high-quality content becomes more daunting. To address this, YouTube occasionally asks users for feedback on videos. In one such event (the YouTube Comedy Slam), users were asked to rate which of two videos was funnier. This yielded sparse pairwise data indicating a participant's relative assessment of two videos. Given this data, several questions immediately arise: how do we make inferences for uncompared pairs, overcome noisy, and usually contradictory, data, and how do we handle severely skewed, real-world, sampling? To address these questions, we introduce the concept of a domination-graph, and demonstrate a simple and scalable method, based on the *Adsorption* algorithm, to efficiently propagate preferences through the graph. Before tackling the publicly available YouTube data, we extensively test our approach on synthetic data by attempting to recover an underlying, known, rank-order of videos using similarly created sparse preference data.

Keywords: Preference Propagation, Social Networks, Label Assignment

1 Introduction

Since the launch of YouTube in 2005, YouTube has become an immensely popular destination for users to find content and share their own videos. It is estimated that YouTube has over 1 billion users who watch hundreds of millions of hours of videos each day. The magnitude of the growth in number of videos and popularity is perhaps best conveyed in the fact that over 300 hours of videos uploaded every *minute* [15].

The task of providing suggestions of non-text content has been explored in a variety of contexts. One of the best known is recommending DVDs to users of Netflix based on ratings data (The Netflix Challenge)[8]. Image search, such as found on Google.com, also incorporates visual and non-visual signals. Closely related, graph-based, approaches have been published


```
axZ2xvYhOKw,TgrUs_ZwwDc,left
Lr4SS1zxRYg,QwpWaLrDHnU,left
wvybYNEDQWE,fY_FQMQpjok,right
Ai6DSPPoOVI,axZ2xvYhOKw,right
Lr4SS1zxRYg,fY_FQMQpjok,left
Lr4SS1zxRYg,fY_FQMQpjok,left
Lr4SS1zxRYg,axZ2xvYhOKw,right
```

Figure 1: Left: YouTube's "Cute-Slam". User-facing video comparison tests conducted (Image from [14]). Right: Sample of the publicly available YTCS data. Each triplet contains an identifier for the 2 videos compared and either "left" or "right" indicating which video the user found funnier.

throughout the last decade: particularly close are Baluja et. al. [2], Chandar et. al [5] and Wu et. al [13], as well as studies in music recommendation [4] and web ranking [10, 9].

In contrast to previous studies with YouTube that use the massive amount of implicit preferences derived from co-views (which videos are often viewed together), we have *explicitly* obtained user preferences. This yields a stronger signal, but is orders of magnitude less plentiful. Through the YouTube Comedy Slam event [14], users were asked to compare two videos and mark the one that they thought funnier (Figure 1). Because of the data sparsity, many issues with noise and contradictory information become pronounced. For example, most pairs of videos are never compared. Of those that are compared, most are compared by a small set of users (< 7), many who have differing opinions of which of the two videos is funnier. Unlike the co-view statistics derived from the massive YouTube view logs, in which the sheer volume disambiguates many contradictions, we do not have that luxury. In this paper, we extend the Adsorption algorithm [2], to work with *domination graphs* to handle these limitations.

2 YouTube Comedy-Slam Data

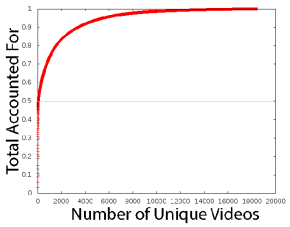
YouTube Comedy Slam was a ranking experiment running on YouTube for a few months in 2011 and 2012 [14] [11]. In this experiment, a pair of videos was shown to the user, and the user was asked to vote for the video that they found funnier. Left/right positions of the videos were randomly selected before being presented to the user to eliminate position bias. Videos were selected from a large pool of weekly updated set of videos. A number of user-facing public test comparisons, or "Slams", were conducted, see Figure 1.

One of the outcomes of the "Comedy-Slam" experiment was a dataset for automatically predicting which video would be deemed funnier by users. Each entry corresponds to one vote over a pair of YouTube videos. The votes were recorded chronologically. For privacy, other information about the voter (e.g. ID of the user or other videos the user may have seen) is not provided. Additionally, we did not use any information (textual comments, extracted labels, visual features) about the video other than the votes represented in this dataset.

In the YouTube Comedy Slam (YTCS) dataset there are total of 912,969 entries. Within these entries, 18,474 unique videos are present with a total of 327,091 unique comparison sets between two videos. Because this was a live test with real-users, for user-experience considerations, the sampling of the videos presented to users was far from uniform; see Figure 2. Of the 912,969*2 total videos in the data set, just 59 videos account for 50% of total entries. The remaining 18,415 videos account for the rest of the 50% of samples.

This biased sampling of videos also yields an extremely non-uniform distribution of comparisons. From the 327,091 total video-pair comparisons, there are 358,978 directed pairs (for

Cumulative Distribution of Videos in the Dataset



Histogram of entropy for pairs of nodes that have contradictory wins over each other.

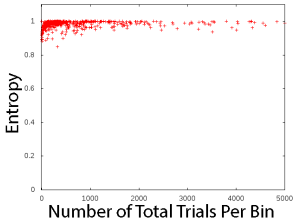


Figure 2: The cumulative distribution of videos in the dataset. X-axis: number of unique videos. Y-Axis: percentage of entries in the data accounted for. Very few videos account for over 50% of the data seen. **Right**: Entropy histogram, grouped by number of total number of trials between node a & b (X-Axis). Note entropy remains near 1.0 (Y-Axis), indicating that the edge from a to b has close to the same weight as the edge from b to a, which means there is no clear winner.

this count, if A wins it is counted differently than if B wins). When we examine the distribution among the 358,978 directed pairs, over 50% of these come from approximately 1600 unique pairs. With 18,474 videos, it is unlikely to find all pairings with enough samples to be certain of their ordering. However, the sparsity problem is far worse, not only do the majority of pairing never appear, but of the pairings that do appear, only a few users compared them.

Given the set of triplets described above, the most straightforward interpretation in terms of a graph is representing each video as a node in a graph. The comparisons can be represented as 358,978 directed weighted edges, with the edge weight denoting the number of wins for each pair of videos. One difficulty in using this simplistic representation is that in addition to the sparsity problems present in the data, the amount of inconsistent / contradictory information contained in the logs is large. For comparisons in which a wins over b there are commonly short paths through which b beats a (for example, b beats c and c beats a, etc.) In fact, of the 327,091 pairs of nodes (a, b) that are in the logfile's triplets, 10% have entries directly indicating that a is funnier than b *and also* that b is funnier than a. Unfortunately, for the nodes that have connections both emanating and ending at each other, the distribution of weights on the nodes is often not discriminatory. Figure 2(Right) shows the average binary entropy when there are both in and out edges from the pair of nodes. Note that the entropy stays large — indicating that the edges have only small differences in weight.

In a directed graph, we can represent edges with (perhaps normalized) weights reflecting the number of votes in each direction. When the directed graph has edges in both direction between two nodes, it represents different users' assessments of the relative "funniness" of the two videos. An alternative is to create a *domination* graph, which indicates for each set of nodes (a, b), which node dominates the other one, in terms of votes. In this graph, rather than having two edges, only the edge with the larger number of votes is kept. Various methods of edge weight normalization were explored; the simplest approach worked the best — simply keep the edge with the larger number of votes and set the weight to the difference in the number of votes between the edges. In this graph, if a beat b N times, and b beat a N times, then there is no information as to which video is actually funnier, only information to be used for transitivity relations. Additionally, the information that the video was seen N*2 times is not important by itself since the number of times a video was presented for comparison was based on external factors. More sophisticated methods of normalization were also used, though of the ones that were explored, little difference in performance was noted.

Next, we describe the Adsorption algorithm, which is used to propagate dominating video-labels through the graph. Intuitively, in the domination graph, each video label should only be propagated, in the first step, to other videos which users have deemed to be *less* funny; hence the label from the dominating node will become present in the nodes that it dominates.

3 Label Propagation

Our primary goal is to rank a node's "goodness/funniness" score relative to the other nodes in the graph. If we know that a video b has been dominated by a, and that b dominates c, how should a's effect on c be measured? Many graph-distance measures have been proposed, e.g. shortest distance, commute time, electrical resistance, etc., but most are expensive to compute for large graphs. Furthermore, conceptually simple ones like shortest distance have undesirable properties; for example, they do not take into account the number of paths between the two nodes. Additionally, many simple methods can be susceptible to noise in the underlying data.

We cast the pairwise ranking problem into a more familiar label propagation framework. Imagine that each node in the graph (which initially represents a unique video) propagates its own label across all of its out-edges. In the next step, each node propagates it's own label *and* the labels it received in the previous step across each of its out-edges. This process is iterated. After a sufficient amount of steps, if we wish to ascertain the relative ranks of a and b, we can compare how much of a's label appears on node b with how much of b's label appears on a.

If we adopt this label-propagation methodology, Adsorption, first presented in [2], provides an intuitive manner in which to propagate labels in a graph that is robust to the types of noise encountered with real-world web data. Adsorption has two intuitive properties that are useful in this domain. First, node v should have a high weight on label l only when there are short paths, with high weight, to other nodes labeled l. Second, the more short paths with high weight that exist, the more evidence there is for l.

In Adsorption, given a graph where nodes have labels, the nodes that have labels will forward them to their neighbors, who, in turn, will forward them to their neighbors, and so on. Each node will collect and forward all the labels that they receive. Thus each node has two roles, forwarding labels and collecting labels. The crucial detail is the choice of how to retain a synopsis that will both preserve the essential parts of this information as well as guarantee a convergent set of label assignments. Formally, we are given a graph $G = (V, E, w)$ where V denotes the set of vertices (nodes), E denotes the set of edges, and $w : E \rightarrow \mathbf{R}$ denotes a non-negative weight function on the edges. Let L denote a set of labels, and assume that each node v in a $V_L \subseteq V$ subset carries a probability distribution L_v on the label set L. We will refer to V_L as the set of labeled nodes. For the sake of exposition, we introduce a pre-processing step, where for each vertex $v \in V_L$, we create a "shadow" vertex $\tilde{v}$ with exactly one out-neighbor, v, connected via an edge with weight 1.0, $(\tilde{v}, v)$; furthermore, for each $v \in V_L$, we will re-locate the label distribution L_v from v to $\tilde{v}$, leaving v with no label distribution. Let $\tilde{V}$ denote the set of shadow vertices, $\tilde{V} = \{\tilde{v} | v \in V_L\}$. From now on, we will assume that at the beginning of the algorithm, only vertices in $\tilde{V}$ have non-vacuous label distributions. (This assumption will not hold when the propagation begins).

Once the above terms are specified, Adsorption is straightforward to implement and can be conceptually explained in only a few lines of pseudo-code; see Figure 3. It is important to note that we do not *update* the label distribution in each round; rather, we *fully recompute* it based on the distributions delivered by the incoming edges from the neighbors.

The goal of Adsorption is to maintain a synopsis of all the labels that are reachable from a vertex. The normalization step that follows the summation step is what distinguished Adsorption from many of the other label propagation algorithms. Labels that are received from multiple (or highly-weighted) neighbors will tend to have higher mass after this step, so this normalization step renders the Adsorption algorithm as a *classifier* from a machine learning perspective. The algorithm is close to the one presented in Zhu et.al. [17] [16], where they considered the problem of semi-supervised classifier design using graphical models (also see [12] [1] [7]).

ADSORPTION

- Input: $G = (V, E, w), L, V_L$
- repeat:

  for each $v \in V \cup \tilde{V}$ do:

  Let $L_v = \Sigma_u w(u, v) L_u$

  end-for

  Normalize L_v to have unit L_1 norm
- repeat until converged
- Output: Distributions $\{L_v | v \in V\}$

Figure 3: Basic Adsorption Algorithm. Adsorption has an efficient iterative computation (similar to PageRank [3]), where, in each iteration, a label distribution is passed along every edge.

As described in [2], Adsorption has a formal equivalence in terms of taking random-walks on the connectivity graph. This is similar to PageRank (PR) in which a fixed Markov random walk is considered. In PR, the stationary probability distribution gives, for each node of the graph, the probability that the walk visits that node. Assuming the walk is ergodic (and no absorbing states), the starting point of the random walk is irrelevant in determining the probability of reaching any particular node. Consequently, PR does not allow us to measure the influence of nodes on each other. With Adsorption, the labeled nodes are *absorbing states of the random walk*; therefore, the starting point of the walk determines the probability of stopping at any of the absorbing states (and outputting the labels found). Thus, the probabilities revealed *do* measure the influence of nodes on each other. This is exactly what we need in this domain, where we wish to determine the relative influence of videos on each other.

From the random-walk interpretation of the algorithm, an important extension is derived. The random walk is augmented with an abandonment probability. At each step, with a small probability, the algorithm abandons the random walk *without producing any output label*. By having a positive abandonment probability at each step, it effectively restricts the length of the random walk – thereby ensuring that the influence of a label l on a node u falls off exponentially in the number of nodes along the paths from nodes that carry l to u. Thus, this strengthens the effect of a node on *nearby* nodes; a crucial property. Though beyond the scope of this paper, if we replicate this abandonment probability in the averaging version of the Adsorption algorithm, the abandonment probability is translated into a second, shadow, injection-node into each node. The label that is injected is a "Dummy" label – which serves to dampen the effect of the other labels that are both injected and passed through the node.

In our implementation, we replace [2]'s "Dummy" label with a small uniform distribution over all of the labels that are not already directly injected into the node via a shadow vertex. This is equivalent to creating edges (in the reverse graph) that lead to all the other shadow vertices with a small probability. We set the total probability of these edges to $d\%$ of the total injection value coming into the node from the shadow vertex. In one interpretation, this is similar to the random jump probability in PageRank; so we set $d = 0.15$ as is standard with PR [3]. This also has the benefit of placing each label (with a small probability) on each node; this will simplify rank comparisons, as described in the next section.

4 Controlled Experiments

Before working with the YTCS data, we present extensive experiments with synthetic data. Working with the synthesized data helped set the algorithm's hyper-parameters so that little tuning is required with the YTCS data and the results can be considered clean. In the synthetic data set, 1,000 videos are simulated. They are explicitly ordered according to a single metric (to make it analogous to the YTCS data, assume this is an underlying "funniness" score). We

then sample pairs of videos from this dataset and return a single (left,right) result, indicating which of the two videos is deemed funnier. The underlying triplet (video1, video2, left/right) is similar to the YTCS data. A total of (4-trial-settings $\times$ 5-noise-settings $\times$ 2-distributions =) 40 variants are tested for every approach tried.

- *Trials*: how many triplets are collected. Four settings are tried: (t =) 5000, 10,000, 100,000 and 250,000. Note that with 1000 nodes, collecting only 5,000 samples yields a severely sparse graph.
- *Noise*: the probability of the winner not being selected according to the actual ordering. Five settings were tried: (n =) 0.00, 0.05, 0.10, 0.25, 0.40. When $n = 0.40$, in 40% of the triplets, the winner does not correspond to the true underlying rank.
- *Sampling Distribution*: the videos selected for inclusion in the triplets are either selected uniformly or through a biased sampling. The biased sampling is more representative of the YTCS data - where an item's popularity is not directly a function of its quality. In our tests, the biased trials are sampled from an extremely steep Zipf's law distribution.

Because we are creating an underlying graph with 1,000 nodes, note that in the cases with a small number of trials (particularly $t < 10,000$), the graph will be sparse and the majority of pairs of videos will not be represented in the triplet-data collected. The sparsity of the graph necessitates the propagation of information through many hops. This contrasts previous studies [6] in which modeling the transitions of the existing links was the primary objective.

For the experiments, we need to infer the ranks between pairs of videos which are not directly compared by users. From the 499,500 total possible pairs, 1,000 pairs are randomly selected and inserted into test set S. No restriction is placed on the distance between pairs (in terms of edge hops). When the t trials are simulated, none are simulated between any pair in S. The goal of this task is to recover the correct pairwise ranking of nodes not directly connected in the graph. In this experiment, because we are working with synthesized data, the underlying ranking of all the videos is known and the accuracy of the results is straightforward to measure.

First, we employ Adsorption. Each of the 1,000 nodes injects its own label, v in each step. The injection amount is weighted by the total number of *wins* for the video (the number of times is appeared in the logs and won). After mixing, up to 1,000 labels are present on each of the 1,000 nodes in varying amounts; a unit-L_1 norm label distribution L_u is present on node u. When two nodes are compared, and the single funnier one must be chosen, the rank of the two videos a & b is computed as follows (after the label propagation iterations are completed): we compare how much *Label-b* is present on node a (i.e. $L_a(b)$ - the magnitude of the label b in the normalized label distribution) and how much *Label-a* is present on node b (i.e. $L_b(a)$). If $L_a(b) < L_b(a)$ then a ranks higher than b (in terms of YTCS, a is funnier than b).

In addition to Adsorption, PR is also tested. Recall that when the domination graph was created, the goal was to retain the edges on which to propagate labels. If node b had an incoming edge from node a, semantically, it meant that a won more times when compared to b than vice-versa. Therefore, b directly received the label a from the edge that connected it to node a (as well as the other labels that a accumulated in previous steps). Although this is a suitable graph for Adsorption, for PR, each node should, instead, *point to other nodes that it considers good*. (When PR was used for web-page ranking and search, the assumption was that a web page links to other good web-pages). We therefore reverse the domination graph. Each video now *points to* the videos that it is dominated by. Once PR is run on this reversed graph, video a and b are compared simply by their PR score.

Table 1 shows the results on the task of correctly determining the relative rank of the 1,000 test pairs of videos while the number of total log-entries (trials), t, and the amount of noise n is varied. For this round of testing, the videos are chosen uniformly randomly. The table is

Table 1: Results with synthetic data on 1,000 videos uniformly sampled. Accuracy is shown on the held-out test set, S. Right Table: % Relative reduction in error (in parentheses: absolute reduction in error). Bolding indicates Adsorption outperformed PR. Omitted results are not statistically significant.

AD-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.997	0.935	0.951	0.946	0.879
t =100000	0.988	0.921	0.934	0.918	0.818
t =10000	0.875	0.854	0.838	0.778	0.638
t =5000	0.798	0.806	0.782	0.713	0.599

PR-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.982	0.866	0.894	0.914	0.852
t =100000	0.972	0.844	0.869	0.879	0.780
t =10000	0.906	0.807	0.765	0.708	0.614
t =5000	0.853	0.778	0.730	0.652	0.570

Signif-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	**81% (0.01)**	**51% (0.07)**	**54% (0.06)**	**37% (0.03)**	**18% (0.03)**
t =100000	**58% (0.02)**	**49% (0.08)**	**50% (0.07)**	**32% (0.04)**	**17% (0.04)**
t =10000	-33% (-0.03)	**24% (0.05)**	**31% (0.07)**	**24% (0.07)**	**6% (0.02)**
t =5000	-37% (-0.05)	**13% (0.03)**	**19% (0.05)**	**18% (0.06)**	**7% (0.03)**

Table 2: Same as above, when 1,000 videos are sampled with heavy bias.

AD-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.974	0.897	0.849	0.773	0.709
t =100000	0.956	0.895	0.872	0.776	0.680
t =10000	0.845	0.832	0.809	0.726	0.603
t =5000	0.779	0.778	0.754	0.677	0.577

PR-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.917	0.815	0.777	0.757	0.720
t =100000	0.908	0.819	0.784	0.748	0.685
t =10000	0.819	0.779	0.747	0.663	0.583
t =5000	0.750	0.718	0.693	0.625	0.556

Signif-Test	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	**69% (0.06)**	**45% (0.08)**	**32% (0.07)**	-	-
t =100000	**53% (0.05)**	**42% (0.08)**	**41% (0.09)**	**11% (0.03)**	-
t =10000	**14% (0.03)**	**24% (0.05)**	**25% (0.06)**	**19% (0.06)**	**5% (0.02)**
t =5000	**12% (0.03)**	**21% (0.06)**	**20% (0.06)**	**14% (0.05)**	**5% (0.02)**

divided into three sub-tables. The two sub-tables on the left show the accuracy of determining the correct ordering for Adsorption (above) and PageRank (below). Each result cell is an average of at least 10 trials in which the entire test scenario (the test-nodes, the trial-pairs and the trial outcomes) was created randomly. In the right sub-table, if an entry is present, the difference between Adsorption and PageRank was statistically significant ($p < 0.05$). Each entry is the percent reduction in error between Adsorption and PageRank; in parentheses is the raw difference in performance.

As the number of trials, t, increases, we notice a steady increase in accuracy. With more trials, more of the graph is connected, and therefore more confidence can be placed in the estimates. Importantly, there is usable signal even with noise at even 40%. Adsorption outperforms PageRank for this task in the majority of cases. The settings in which PageRank outperforms Adsorption are at the bottom-left of the tables – where there are few samples, but also no noise. Even with small amounts of noise, PR does worse. Because of the noisy nature of the real YTCS data, we expect that we will be closer to the middle columns of the table.

In addition to testing how well the algorithms performed in the pairwise ranking of videos not directly in the training data, a second experiment was conducted. In the second task, *all* pairs of nodes were tested to examine how well the complete set of pairwise ranks is encoded in the graph. For these experiments, the full 499,500 pairwise combinations are tested. The accuracy on this full-set is almost identical to Table 1 (new results not shown due to space).

One of the fundamental differences between the tests presented above and the YTCS data is the procedure by which videos are selected for inclusion in the trials. As described earlier, there is a severe bias in the sampling of videos that were selected for inclusion in the user-ranking. To simulate this, instead of uniform sampling, the videos are sampled with the severely biased distribution and all experiments are repeated. See Table 2.

Similar to drawing with uniform distributions, Adsorption outperforms PR in the majority of settings. Only when the noise reaches $> 25\%$ do the differences in PR and Adsorption disappear. Additionally, recall that this is a pessimistic scenario in which the trial sampling

Table 3: Results with synthetic data and biased sampling. Shortest-Path (left) and best-of-5 heuristics (right). Fraction correct when inferring all pairwise ordering of videos.

Shortest Path

Shortest-Path	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.947	0.922	0.912	0.860	0.792
t =100000	0.904	0.881	0.868	0.831	0.750
t =10000	0.628	0.780	0.774	0.715	0.586
t =5000	0.490	0.697	0.694	0.630	0.535

5 Heuristics

Best-of-5	n =0.0	n =0.05	n =0.10	n =0.25	n =0.40
t =250000	0.884	0.871	0.858	0.814	0.735
t =100000	0.867	0.852	0.839	0.790	0.695
t =10000	0.758	0.744	0.726	0.658	0.545
t =5000	0.689	0.676	0.660	0.587	0.497

has no correlation with the rank/quality of the video. In summary, in comparison to the uniform selection tests, the performance decreased across all settings. However, given the severity of the lopsidedness in samples, the reduction in performance remained surprisingly small.

4.1 Six Simpler Approaches: Shortest Paths, Popularity and More

Rather than using PageRank or Adsorption, a simpler alternative is to use the shortest paths through the graph. In the directed domination-graph, when comparing video a to b, if a has a shorter path to b than b does to a, we believe that there is stronger indication that a dominates b. Unlike Adsorption and PR, however, shortest path-based metrics do not account for multiple paths between nodes. For completeness, 5 additional heuristics were also tested:

- *Popularity*: Video with the most overall views wins.
- *Smallest In-Degree (Least Dominated)*: Video that lost to the fewest other unique videos wins.
- *Smallest Total Losses*: Video that lost the fewest times in the logs wins.
- *Largest Out-Degree (Most Dominating)*: Video that won over the most other unique videos wins.
- *Most Total Wins*: The video that won the most trials in the logs wins.

The entire cross section of experiments was repeated for all six heuristics, again with uniform and biased sampling. In the interest of space, the complete set of results is omitted. Instead, we report results on shortest-paths and also combine the remaining five heuristics as a single meta-heuristic and use the best-score for each of the five and compare that single best score with the Adsorption scores. Note that this makes it maximally hard for Adsorption to beat these heuristics. The results are presented in Table 3 (Left: shortest paths) & (Right: Best-of-5). With shortest paths and uniform sampling, the results were as expected. Under biased sampling, with enough data ($t = 100,000$) shortest path is able to outperform both PageRank and Adsorption. This leaves shortest-path as a candidate for the YTCS data.

For the combination of five heuristics, when there are fewer trials, the best-of-five heuristics do not perform as well as Adsorption. As the number of trials increases, the performance differential diminishes and even turns in favor of the heuristics. In the best case, these heuristics are most effective when there is a abundance of data relative to the size of the graph. However, as the YTCS data is extremely sparse, these heuristics are unlikely to be suited for this task.

5 YouTube Comedy Slam

In the previous section, extensive experiments were conducted with carefully controlled data. In this section, we return to the task at hand: the YouTube Comedy Slam Data. Unfortunately, we do not know, even for individual pairs, which video should *actually* be ranked higher. Therefore, to generate the test set we must use and trust the data in the logs. For the test set we select a pair of nodes (a, b) to be included if: (1) The pair of nodes (a, b) was compared by users more than 5 times; this helps overcome spurious noise. (2) The ratio of a's wins over b is either greater

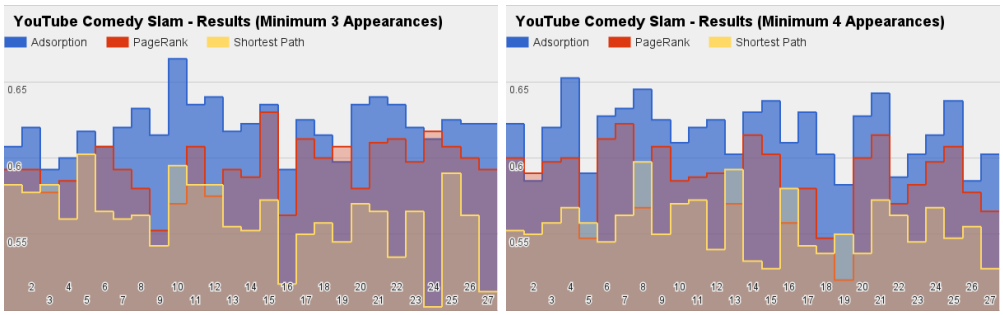


Figure 4: YouTube Comedy Slam Results - Adsorption (Blue), PageRank (Red) and Shortest-Path (Yellow) compared. Left: Videos must appear 3+ times in the log file to be included in the test. Right: 4+ times. Each of the 28 shown experiments was initialized with a random test set, S. Similar average performance was obtained for 3+ and 4+ constraints.

than 2.0 or less than 0.5. This ensures that there is a real difference in perceived funniness. (3) Neither a nor b were previously selected to be in the test set. This ensures that a diverse set of pairs that are well distributed throughout the graph are tested.

Given these constraints, 400 pairs of nodes were randomly chosen to be included in the test set, S. The edges between each of the 400 pairs of nodes were removed from the graph. Four algorithms were compared: Popularity, Shortest Path, PageRank and Adsorption. Each was applied in exactly the same manner as described in the previous two sections. For each algorithm, the experiments were repeated at least 28 times with a different set of randomly selected pairs of nodes in S for each trial. Figure 4(left) shows the individual results of the trials. As can be seen, in the vast majority of cases, Adsorption surpassed Shortest Path and Page Rank. Popularity is not shown due to its poor performance.

In the above test, in order to be included in the graph, a video must have appeared in the log-files at least three times ($\sim 14,500$ videos matched that criteria). We also checked how the results would differ if we increased the minimum requirement to four appearances ($\sim 12,700$ videos matched that criteria). By increasing the minimum requirement, we hoped to eliminate some of the potentially spurious connections in the graph and see if that changed the relative ordering of the algorithms. The results are shown in Figure 4(right) and summarized in Table 4.

For the experiments with 3+ and 4+ minimum appearances of a video in the logs, the results were similar. All the differences in the algorithms tested were statistically significant (Table 4). As is clearly evident in the graphs and table, the overall ordering of the algorithms from worst to best is: Popularity, Shortest Path, PageRank and Adsorption.

Table 4: YTCS-results summary. Each cell represents the average of at least 28 experiments. Two experimental setups were tried (Left:) Each video had to appear 3+ times in the logs. (Right:) 4+ times. Results are measured on the randomly selected test set, S.

	Popularity	Shortest-Path	PageRank	Adsorption
Correct Score	0.46	0.56	0.59	0.62
Reduction in Error from Popularity		18.0% (0.10) ($p < 0.001$)	23.6% (0.13) ($p < 0.001$)	29.27% (0.16) ($p < 0.001$)
Reduction in Error from Shortest Path			6.8% (0.03) ($p < 0.001$)	13.7% (0.06) ($p < 0.001$)
Reduction in Error from PageRank				7.4% (0.03) ($p < 0.001$)

	Popularity	Shortest-Path	PageRank	Adsorption
Correct Score	0.46	0.55	0.58	0.62
Reduction in Error from Popularity		17.0% (0.091) ($p < 0.001$)	22.8% (0.122) ($p < 0.001$)	28.60% (0.153) ($p < 0.001$)
Reduction in Error from Shortest Path			7.0% (0.031) ($p < 0.001$)	14.0% (0.062) ($p < 0.001$)
Reduction in Error from PageRank				7.5% (0.031) ($p < 0.001$)

6 Conclusions

This paper has explored methods to use sparse, noisy, pairwise relative rank data to infer the order of (1) nodes that were not included in the pairwise data and (2) the entire set of nodes to obtain a a complete pairwise ranking of nodes in the graph. In the YouTube Comedy Slam tests, Adsorption and PageRank outperformed common heuristics (Adsorption outperformed popularity by 29% and shortest-paths by 14%), with Adsorption further outperforming PageRank by 7.5%. These performance gains are particularly notable given the complexity of the underlying real-world data. Not only was the data skewed because of the sampling methodology, but also contained severely contradictory and incomplete information.

In addition to testing on the YTCS data, an extensive set of experiments were conducted with synthetic data. In these experiments, the number of trials and the amount of noise was explicitly controlled to understand when each algorithm will outperform the others.

This work opens up a number of interesting future directions for study. First, unlike PageRank, where a single value is stored on a vertex, in Adsorption, a distribution over all labels is required. Rather than keeping all labels on every node, maintaining only a small top-subset of the labels on each node may suffice. This has the potential to improve training times. Second, questions arose from the experiments that remain intriguing, for example with biased selection, shortest-paths had very different/better performance characteristics than other algorithms. Third, given YouTube's upload rate, augmenting the graph with new videos as they become available is an important avenue to consider.

References

[1] Arik Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning*, pages 49–56. ACM, 2007.

[2] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW-2017*, 2008.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[4] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music rec. by unified hypergraph: combining social media inf. & music content. In *Proc. Int. conf. on Multimedia*, 2010.

[5] P. Chandar and B. Carterette. Using pagerank to infer user preferences. In *35th SIGIR*, 2012.

[6] Ravi Kumar, A. Tomkins, S. Vassilvitskii, and Erik Vee. Inverting a steady-state. In *WSDM*, 2015.

[7] Yuzong Liu and Katrin Kirchhoff. A comparison of graph construction and learning algorithms for graph-based phonetic classification. Technical report, UWEE Technical Report, UWEETR-2012-0005, 2012.

[8] Netflix. The netflix prize, 2007. `http://www.netflixprize.com/index`.

[9] Shuang Qiu, Jian Cheng, Ting Yuan, Cong Leng, and Hanqing Lu. Item group based pairwise preference learning for personalized ranking. In *Proceedings of the 37th ACM SIGIR*, pages 1219–1222. ACM, 2014.

[10] Amit Sharma and Baoshi Yan. Pairwise learning in recommendation: experiments with community recommendation on linkedin. In *Proc. of the 7th ACM Conf. on Rec. Sys.*, pages 193–200. ACM, 2013.

[11] Sanketh Shetty. Quantifying comedy on youtube: why the number of o's in your lol matter, 2012. `http://googleresearch.blogspot.com/2012/02/quantifying-comedy-on-youtube-why.html`.

[12] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *NIPS-14*, 2002.

[13] Xiao-Ming Wu, Zhenguo Li, Anthony M So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. In *Advances in Neural Information Processing Systems*, pages 3077–3085, 2012.

[14] YouTube. Slam, 2011. `http://youtube-global.blogspot.com/2011/12/introducing-youtube-slam.html`.

[15] YouTube. Youtube press statistics, 2015. `https://www.youtube.com/yt/press/statistics.html`.

[16] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.

[17] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.