Appeared in 2007 SIGIR conference on research and development in information retrieval

Random Walks on the Click Graph

Nick Craswell and Martin Szummer
Microsoft Research Cambridge
7 JJ Thomson Ave
Cambridge, UK
{nickcr,szummer}@microsoft.com

ABSTRACT

Search engines can record which documents were clicked for which query, and use these query-document pairs as 'soft' relevance judgments. However, compared to the true judgments, click logs give noisy and sparse relevance information. We apply a Markov random walk model to a large click log, producing a probabilistic ranking of documents for a given query. A key advantage of the model is its ability to retrieve relevant documents that have not yet been clicked for that query and rank those effectively. We conduct experiments on click logs from image search, comparing our ('backward') random walk model to a different ('forward') random walk, varying parameters such as walk length and self-transition probability. The most effective combination is a long backward walk with high self-transition probability.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation, Theory

Keywords

Web search, models, click data, user behavior, image search

1. INTRODUCTION

A search engine can track which of its search results were clicked for which query. For a popular system, these click records can amount to millions of query-document pairs per day. Each pair can be viewed as a weak indication of relevance: that the user decided to at least view the document, based on its description in the search results. Although clicks are not real judgments, there is evidence that they are useful, for example as training data [6, 2], as annotations [14, 15], for query suggestion [13, 4, 3] or directly as evidence for ranking [1].

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

We can use the clicks of past users to improve the current search results. However, the clicked set of documents is likely to differ from the current user's relevant set. Some differences arise because we are aggregating clicks across users, who may simply disagree about which documents are relevant. Other differences are due to presentation issues; for example, the user must decide whether to click based on a short summary and is influenced by the ordering of results [7]. For any given search, a large number of documents are never seen by the user, therefore not clicked.

From the perspective of a user conducting a search, documents that are clicked but not relevant constitute noise in the click data. Documents that are relevant but not clicked constitute sparsity in the click data.

One class of approaches attempts to reduce noise in click data, by building a click model that may use additional information about the user's behaviour [5, 1]. For example, taking into account the user's browsing patterns after clicking a document. These approaches can significantly reduce noise, by identifying some clicked documents as irrelevant.

This paper focuses on the sparsity problem, although our model also has noise reduction properties. The model gives a probabilistic ranking of documents, which includes relevant documents that have not yet been clicked for the current query. The sparsity problem is evidenced by power law distributions observed in click logs [15]. Most queries in the click log have a small number of clicked documents. In such cases, it is useful to identify additional relevant documents.

We first describe the click information as a graph, and survey a range of click graph applications. Then we detail our Markov random walk model for finding relevant documents. The subsequent sections describe a real click dataset, and empirical evaluation of the new methods.

2. ALGORITHMS ON THE CLICK GRAPH

Our current model uses click data alone, without considering document content or query content. To describe the data, and related non-content-based approaches, it is useful to think of the click information as a graph. The graph is bipartite, with two types of nodes: queries and documents. An edge connects a query and a document if we have observed a click for that query-document pair by any user. The edge may be weighted according to the total number of clicks from all users. An example click graph, taken from image search, is depicted in Figure 1.

Algorithms employing the graph can find associations between nodes that are not directly linked (adjacent), perhaps

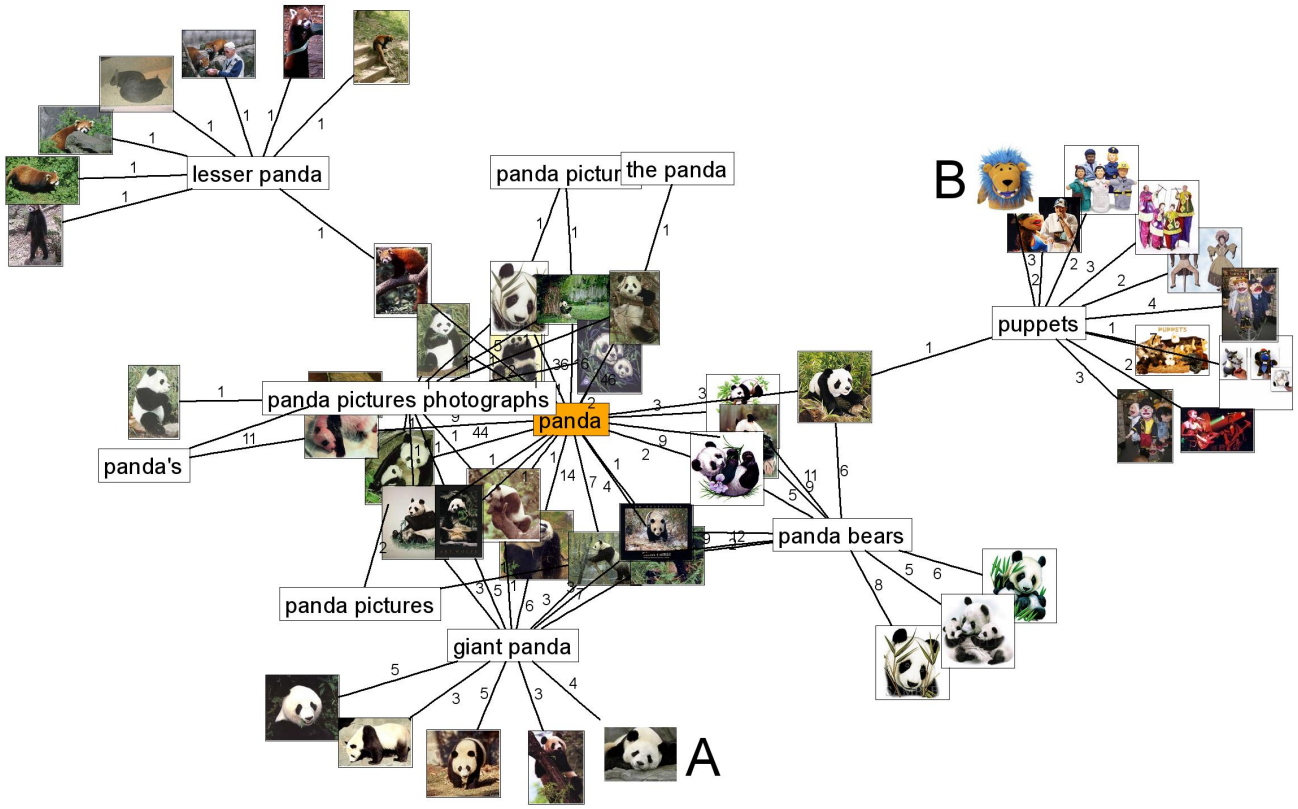

Figure 1: Click graph. Nodes are queries or images, edges indicate clicks. Images A and B are equidistant from the query 'panda' (distance=3), so retrieval based on a naïve shortest-path algorithm could not distinguish them. Our Markov random walk approach sums over paths, so image A benefits from having 7 distinct paths of length 3. In other words, nodes A and 'panda' are connected by a large "volume" of paths.

by following paths in the graph. Application areas for such algorithms include:

- **Query-to-document 'search'.** Given a query, find relevant documents, as in adhoc search. Relevant documents should be ranked highly regardless of whether they are adjacent to the query. Search is the focus of this paper.
- **Query-to-query 'suggestion'.** Given a query, find other queries that the user might like to run. This is more difficult to evaluate, but approaches already exist for finding related queries using the click graph [4, 13].
- **Document-to-query 'annotation'.** Given a document, attach related queries to it. This method for creating document surrogates, based entirely on the click graph, was studied in [15].
- **Document-to-document 'relevance feedback'.** Given an example document that is relevant to the user, find additional relevant documents.

Two of the above approaches [4, 13] use the graph to identify query-to-query similarity followed by the application of clustering algorithms. Xue *et al* [15] use the graph to find document-to-document similarities. Annotations (query associations) are then spread amongst similar documents. All three papers make use of co-click/co-visitation information, for example, two queries are similar if they have overlapping sets of clicked documents. Xue *et al* also try an iterative algorithm, based on query-to-query and document-to-document similarity.

The focus of this paper is the search problem. However, the Markov random walk methods studied here handles all these cases. We discuss other cases briefly in Section 5.

3. RANDOM WALK MODEL

To derive our probabilistic retrieval model, we first propose a basic query formulation model. The model captures a process that starts from an information need and ends with a query.

We assume that query formulation begins with the user imagining a single document, representing their information need. They then think of a query that is associated with the document. The process might stop at that query, at which point they issue the query. Alternatively, the query makes them imagine another document, and that document makes them imagine another query. This thought process of query-document and document-query transition can repeat, or it can stop at a query which is then issued.

Our model, detailed in Section 3.1, makes a number of simplifying assumptions. The user has limited memory, so forgets their previous location after each transition. Although they do not remember their starting point, our model limits the number of transitions to keep them in the vicinity

of their information need. We do not base our model on a real study of query formulation behavior, but instead estimate our transition probabilities from clicks of many users. It is also a simplifying assumption to use a single document to represent the information need.

This model can also be thought of as a noise process. It starts with a desired document, then adds noise via taking some number of steps. Given a document, this noise process describes a probability distribution over queries. Mathematically, it corresponds to a Markov random walk. The benefits of such a walk are robustness to spurious clicks (edges) and, by considering multiple weighted paths in the graph, implicitly finding a cluster of nodes.

Our retrieval model is obtained by inverting the query formulation model. It starts from an observed query, and attempts to undo the noise, inferring the underlying information need. This inference is done by conditioning the query formulation model on the observed query. The result is a distribution over documents, describing how likely they are to be the users original document. This corresponds to a Markov random walk in the opposite ('backwards') direction, compared to the query formulation walk, with similar noise reduction and clustering benefits.

The probability distribution over documents allows us to rank according to the probability ranking principle. Documents are ranked in descending order of the probability that they were the information need.

3.1 Random Walk Computation

Our probabilistic model and notation are heavily influenced by Szummer and Jaakkola [10]. Let $\mathcal{D}$ be the set of documents, and $\mathcal{Q}$ be the set of queries. We construct a graph whose nodes $\mathcal{V}$ are the union of these, $\mathcal{V} = \mathcal{D} \cup \mathcal{Q}$. The edges $\mathcal{E}$ correspond to user clicks, with weights given by click counts C_{jk}, associating node j to k.

We define transition probabilities $P_{t+1|t}(k \mid j)$ from j to k by normalizing the click counts out of node j, so $P_{t+1|t}(k \mid j) = C_{jk} / \sum_i C_{ji}$, where i ranges over all nodes. The notation $P_{t_2|t_1}(k \mid j)$ will denote the transition probability from node j at step t_1 to node k at time step t_2. While the counts C_{jk} are symmetric, the transition probabilities $P_{t+1|t}(k \mid j)$ generally are not, because the normalisation varies across nodes.

The original click edges form a bipartite graph, but we extend it by adding self-transitions to the nodes. Let the self-transition probability be s. Then the transition probabilities become

$$P_{t+1|t}(k \mid j) = \begin{cases} (1-s)C_{jk} / \sum_i C_{ji} & \forall k \neq j \\ s & \text{when } k = j. \end{cases} \quad (1)$$

Self-transitions allow the random walk to stay in place, and reinforce the importance of the starting point by slowing diffusion to other nodes. In query formulation terms, this corresponds to the user favouring the current query or document, and not changing it for a step.

We can organise the one-step transition probabilities as a matrix $\boldsymbol{A}$ whose j,k-th entry is $P_{t+1|t}(k \mid j)$. The matrix $\boldsymbol{A}$ is row stochastic so that rows sum to 1.

Now we perform the random walk: we calculate the probability of transitioning from node j to node k in t steps, denoted $P_{t|0}(k \mid j)$, and equal to $P_{t|0}(k \mid j) = [\boldsymbol{A}^t]_{jk}$.

The random walk sums the probabilities of all paths of length t between the two nodes. It gives a measure of the *volume* of paths [10] between these two nodes; if there are many paths the transition probability will be higher. See Figure 1 for an example.

For retrieval, we calculate the backward random walk: given that we *ended* a t-step walk at node j, we find the probability of *starting* at node k, denoted $P_{0|t}(k \mid j)$. This is calculated by Bayes rule $P_{0|t}(k \mid j) \propto P_{t|0}(j \mid k)P_0(k)$, assuming that the starting points are chosen uniformly at random, $P_0(k) = 1/N$, where N is the number of nodes. The normalisation required by Bayes rule can be written as a matrix multiplication, so that $P_{0|t}(k \mid j) = [\boldsymbol{A}^t \boldsymbol{Z}^{-1}]_{kj}$, where $\boldsymbol{Z}$ is diagonal and $\boldsymbol{Z}_{jj} = \sum_i [\boldsymbol{A}^t]_{ij}$. Finally, we rank the documents according to $P_{0|t}(k \mid j)$.

Since our click datasets are large, we compute the random walks in an efficient way as follows. We represent the transitions as a sparse matrix $\boldsymbol{A}$. For a backward walk, we encode the distribution at step t as a vector $\boldsymbol{q}_j$ with a single unit entry corresponding to the query node j. Then we calculate $P_{0|t}(k \mid j) = [\frac{1}{Z_j}\boldsymbol{A}(\ldots(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{q}_j)))]_k$, in order of the parentheses, and where Z_j normalises the result to sum to one over k. This is efficient because these matrix operations are sparse. Similarly, to calculate a forward random walk we encode the start distribution as a row vector $\boldsymbol{v}_j$ with a unit entry at query node j, and obtain $P_{t|0}(k \mid j) = [(((\boldsymbol{v}_j\boldsymbol{A})\boldsymbol{A})\ldots)\boldsymbol{A}]_k$.

3.2 Forward vs. Backward Walks

We have proposed to rank retrieved documents by $P_{0|t}(k \mid j)$. Previously, forward random walks that correspond to $P_{t|0}(k \mid j)$ have been assumed in spectral clustering, PageRank, and various contexts (Section 3.6). Note, PageRank is a query-independent forward random walk on the link graph, which proceeds to its stationary distribution. Ours is a query-dependent backward random walk on the click graph, where walk length is moderated by t and s.

In statistics, the backward walk model is referred to as diagnostic: to find the "cause" of the query j, we infer what documents k the walk may have arisen from. In contrast, the forward walk model is predictive: we start from a query j, and calculate probabilities of ending documents $P_{t|0}(k \mid j)$.

To illustrate the difference, consider the limit of infinite walks, $t \to \infty$. The forward random walk approaches the stationary distribution (assuming ergodic transitions), which gives high probability to nodes with large numbers of clicks. The backward random walk approaches the prior starting distribution, which we have taken to be uniform. We believe that the backward walk is a more natural formulation for our retrieval task, and results (section 4.2) show that it performs better.

3.3 Clustering Effect

The random walk implicitly performs a soft clustering of nodes [11]. As the walk gets longer, the identity of nodes in clusters blurs together. Given an end node that is part of a cluster, we have similar probabilities of having started the walk from any node in the cluster, but lower probability for nodes outside the cluster. The clusters do not have hard boundaries, and a ranking of nodes is still retained for finite walks, as the start point probabilities do not become exactly equal.

To illustrate this, consider a synthetic document-query graph containing two clusters (Figure 2). We apply backward walks with self-transition probability of 0.9. Query

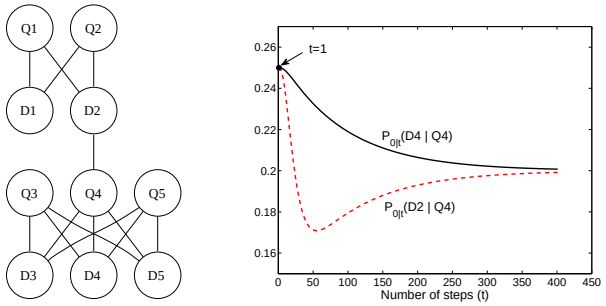

Figure 2: Two clusters (left figure). A one step walk ending at Q4 is just as likely to have come from D2 as D4 (right figure, point marked t=1), even though D2 is in a different cluster. A walk of medium length finds D4 a more likely starting point, exhibiting a clustering effect. (The shown probabilities are conditioned on starting at a document).

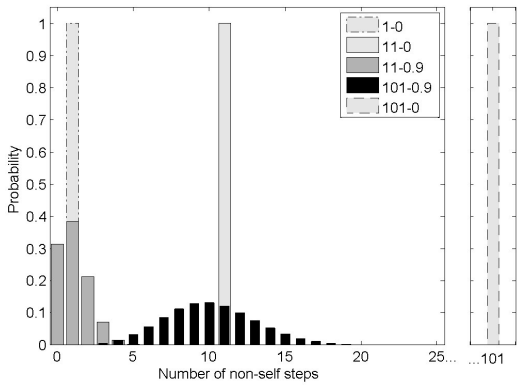

Figure 3: A walk with zero self-transition probability will make exactly t non-self transitions ($t = 1, 11, 101$ shown here). In general, the number of non-self transitions has a binomial distribution, shown here for 5 different walks.

node Q4 is adjacent to several documents and a 1-step walk finds them equally probable. A longer walk exhibits a clustering effect, finding the adjacent node from the same cluster (D4) more likely. The clustering also suppresses the effect of spurious edges that may be due to noise (such as the edge spanning the two clusters in the example). A very long walk finds all nodes to be equally likely starting points.

3.4 Walk Parameters

The behaviour of the Markov random walk is affected by the transition matrix and the number of steps in the walk. The number of steps determines the resolution of the walk. A short walk preserves information about the starting node at a fine scale; start nodes close to the end node have much higher probability than the others, and nodes further away cannot even be reached and have zero probability. A long walk preserves only coarse information about what cluster of nodes the walk was started from.

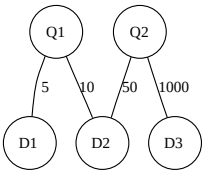

Figure 4: Document-query transition. In our model, the most likely transition from D2 is to Q2, because the edge has 50 clicks (as opposed to 10).

We can choose the number of steps t based on a few heuristics. If $t+1$ equals the diameter of a singly connected graph, then we can ensure that $P_{0|t}(k \mid j) > 0$ so each node influences every other node. However, this scheme ignores transition probabilities. Instead, the *mixing time* of a graph measures the time it takes to approach the stationary distribution. The graph mixes faster the smaller the second largest eigenvalue λ_2 of the transition matrix $\boldsymbol{A}$ (the largest eigenvalue is always 1). We wish to choose t so that we are relatively far from the stationary distribution, where all information about the query has been lost.

The self-transition parameter determines how quickly the walk diffuses. It is related to the number of steps in the walk, but slightly different. A self-transition probability close to 1 corresponds to a slow walk, which mostly stays at the same node. Such a walk requires a large number of steps before other nodes accumulate any significant probability. A self-transition probability of 0 corresponds to a fast walk, which forces a change of node at each step. Figure 3 shows how many non-self transitions are made for the parameter settings used in our experiments. Alternatively, one can also use a decaying exponential distribution over walk lengths, which puts more emphasis on shorter walks, but still includes arbitrarily long walks (capped for computational reasons). Initial experiments show that this performs at least as well as the others.

3.5 Alternative Transition Model

For query-document transitions, it seems natural to prefer the most-clicked document for the query. For document-query transitions it is not so clear. Our main model treats documents and queries symmetrically (eq. 1) so prefers the query with the most clicks. The advantage of this is that the walk will prefer to follow edges where we have the most evidence of relevance. A potential disadvantage is that the walk will prefer popular queries. Figure 4 shows a case where D2 has 5% of the Q2 clicks and 67% of the Q1 clicks, yet under our transition model Q2 is the more likely transition.

We tried a model that normalizes document-query transitions differently, namely by probability rather than by click count,

$$P_{t+1|t}(k \mid j) = \begin{cases} C_{jk} / \sum_i C_{ji} & \forall j \in \mathcal{Q} \\ P_{t+1|t}(j \mid k) / \sum_k P_{t+1|t}(j \mid k) & \forall j \in \mathcal{D}. \end{cases} \quad (2)$$

Here transitions from documents are invariant to the popularity (absolute click-counts) of queries. We do not present evaluation results for this alternative model, but we note that when tested it performed slightly worse than our main model. Another model that merits further study is to combine both absolute and relative click-counts.

3.6 Related Models

Lafferty and Zhai [8] perform a random walk on a different graph, the document-term graph. However, their walk is a forward walk and experiments only show very short 2-step walks, which do not offer much robustness against spurious links or clustering behaviour. Toutanova and Manning et al. [12] also use short 2- and 3-step walks to induce word dependencies for resolving prepositional phrase attachments. They have several different types of node transitions, based on observed word cooccurrences, morphology, WordNet synonyms, and others. They combine the transitions via learned mixture weights. In the machine learning literature, Markov random walks have been employed for clustering of data vectors [11] and for semi-supervised classification [10].

Spectral clustering techniques [9] are closely related to forward Markov random walks. They begin from a similar transition matrix $\boldsymbol{A}$, but clustering is based on the eigenvectors of this matrix. The eigenvectors of the forward random walk $P_{t|0}(k \mid j) = \boldsymbol{A}^t$ are the same, as they are invariant to the choice of t. The random walk weights the eigenvectors by powers of their corresponding eigenvalues λ^t. For high t, only the top eigenvectors will remain significant, just as used in spectral techniques.

The backward random walk differs. The representation $P_{0|t}(k \mid j) = [\boldsymbol{A}^t \boldsymbol{Z}^{-1}]_{kj}$ is normalised by a diagonal matrix $\boldsymbol{Z}^{-1}$, that ensures that the columns of $\boldsymbol{A}^t$ sum to 1. This normalisation does not preserve the right-side eigenvalues or eigenvectors.

4. EXPERIMENTS

4.1 Dataset and Evaluation Method

Our click logs were extracted from a 14-day log of web usage, which includes 12.5 million clicks from a number of commercial image search engines. The set contains 5 million image URLs. There are 1.6 million queries after converting to lower case and normalizing white space. There are 5.9 million unique query-URL click pairs (edges). We observe a number of power-law distributions, including URLs-per-query, queries-per-URL and repetitions of query-URL pairs.

We chose image search because image search logs have the characteristics typical of other click logs, such as power law degree distribution and many 'missing edges' [15]. Yet image click logs have a relatively low level of noise. Users can quickly glance at a number of thumbnails before they click, and the thumbnails are very accurate summaries of image content, so there are fewer speculative clicks. The set of judged images with distance 1 from the query had precision of 75%. This shows that we have sufficient high-quality connections without applying additional noise reduction techniques [5, 1] and can concentrate on our random walk model.

To make the dataset size more manageable we perform a two-stage pruning of the graph. We first remove URLs that are only connected to one query, then remove queries that are only connected to one URL. Our pruned graph has 1.1 million edges, 505,000 URLs and 202,000 queries. We note that pruning may also have decreased noise in the dataset, although we did not attempt to quantify this effect.

The query set for evaluation is sampled, with uniform probability, from the set of query nodes. Thus a rare query is just as likely as a popular query. This study is concerned with sparsity in click data, so we believe a sampling strategy that yields less popular and therefore low-degree queries is reasonable. We rejected pornographic queries and queries that were too difficult to judge (such as URL fragments) giving a set of 45 queries.[1]

Our relevance judgments are to pool depth 20. Evaluating the 45 queries for all the approaches described in Section 4.2 required 2278 relevance judgments in total. Before judging each query, the assessor was allowed to research the query using web resources such as Wikipedia. Judges were then shown full-sized images, with no other information such as URL or position in the graph. Each image was judged to be relevant or irrelevant, identifying 818 relevant images in total. Because the pool size was small, and because in many cases the relevance of an image is evident quickly, the present authors were able to carry out all judging.

Considering the aforementioned 75% relevance rate within a query's judged clicked images, we do not have a breakdown of why the 25% irrelevant were clicked. In some cases it is difficult to judge relevance based on viewing the thumbnail alone, for example in group photos it is sometimes difficult for the user to see who is present based on the thumbnail. In other cases, the user's underlying information need might differ from that assumed by the assessor (and from those of other users who typed the same query). We have also been quite strict in our judging, perhaps assuming a knowledgeable and focused user. For example, for the query 'vassily kandinsky' we would not judge an image of a Miró painting as relevant, but a user who is not knowledgeable or who is not focused on finding Kandinsky might click on a Miró. We take the stricter interpretation because we are interested in building a precise retrieval system.

Given a fully-judged top 20, we can accurately measure precision at 20 (P@20) and also mean average precision at cutoff 20 (MAP@20). However, since we are exploring methods that increase the number of retrievable images, it is also interesting to consider recall, despite the shallow pools. So for analysis we also consider precision-recall graphs.

4.2 Results

Each random walk configuration is denoted by a string such as '1-0-forward'. This means a 1-step forward walk with zero self-transition probability. We consider 1-0-forward to be our baseline, since it is equivalent to ranking a query's URLs in descending order of click count. We also include a control system 'dist' which will be used to help us analyze ranking effects: it ranks images in ascending order of their graph distance from the query, with random ordering within each equidistant image set. The main variables of study are walk length (1, 11 or 101 steps), self-transition probability (0 or 0.9) and direction forward vs. backward.

[1]Judged queries: god of war; ninja metal gear solid; wallpaper bmw; sahin nuri; seth cohen; catalonia map; apollo creed; the fastest cars in the world; lindsay lohan parent trap; pics of space; christina aguilera fighter; fiat bravo 2007; tattoos angelina jolie; zatchbell; matthew macfadyen; psp background; denver broncos cheerleaders; robbie williams feel; the perfect man hilary duff; wrestling cards; kenny miller; vassily kandinsky; black metal; unicorn and fairy; muppet show; candace cameron bure; orc hunter; hilary duff as lizzie mcguire; funny osama bin laden pictures; adu; my fair lady; tiger paintings; ryan and marissa the oc; bunkers; heidi muller; world strongest boy; stars wallpaper; pyramid picture; humming birds; the titanic leonardo dicaprio; slam dunk contest; billabong pro; the pirates of the caribbean; jenny garth; dunkirk.

System	P@20	MAP@20
101-0.9-backward	0.487*	0.567*
11-0-backward	0.482*	0.546*
11-0.9-backward	0.473*	0.564*
11-0.9-forward	0.446*	0.529*
101-0.9-forward	0.442*	0.498*
11-0-forward	0.438*	0.485*
dist	0.339*	0.412
101-0-backward	0.331*	0.319
101-0-forward	0.263	0.252
1-0.9-backward	0.218	0.350
1-0-forward	0.218	0.335
1-0-backward	0.218	0.350
1-0.9-forward	0.218	0.335

Table 1: Overall comparison of systems on 45 queries. The best system is a 101-step backward walk with 0.9 self-transition. Results that differ significantly (paired t-test $p < 0.01$) from the baseline system 1-0-forward are marked with '*'.

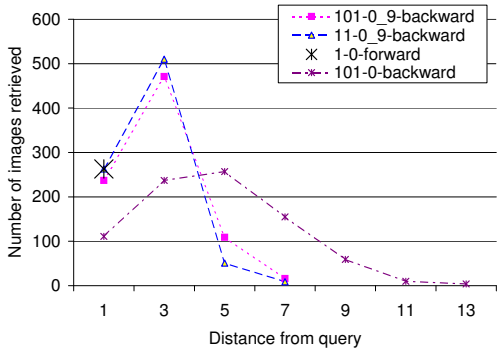

Figure 5: Total number of images retrieved across 45 queries, broken down by graph distance. The one step walk can only find adjacent images, so retrieves in total 263 images, marked as a black cross.

The furthest node from any of our test queries is at distance 41, so the 101-step walk gives non-zero probability to all reachable images. The walks are computed off-line and the most expensive 101-step computation takes less than 20 seconds per query.

The overall results in Table 1 indicate that several of the longer walks are significantly better than the baseline 1-step forward walk. The best method on both metrics is the 101-step backward walk with 0.9 self-transition probability. It is significantly better than the forward walk with the same parameters (101-0.9-forward) with $p = 0.020$ for P@20 and $p = 0.032$ for MAP@20. It is also significantly better than the same walk with zero self transition (101-0-backward) with $p < 10^{-4}$ for P@20 and $p < 10^{-6}$ for MAP@20. However, it is statistically indistinguishable from the backward 11-step walks.

A key advantage of the 11-step and 101-step walks is that they can find relevant documents other than those directly clicked for the query. Figure 5 shows longer walks that return roughly the same number of distance-1 images, but also significant numbers from further afield. The 101-step walk

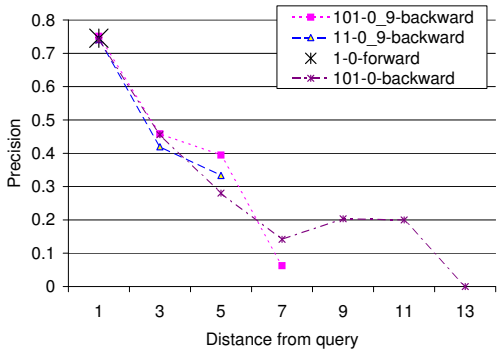

Figure 6: The proportion, at each distance level, of total retrieved (Figure 5) that were relevant.

with zero self-transition possibly goes too far, returning too few distance-1 images.

Figure 6 shows the proportion of results within each distance group that were relevant. The retrieved distance 1 images have 0.75 precision, while those at distances 7 and above are mostly irrelevant. The 101-0-backward walk finds 12/59 relevant at distance 9, 2/10 relevant at distance 11 and 0/4 relevant at distance 13. It finds relevant images that no other system finds, but retrieves too few items at distances 1 and 3 (Figure 5), where high precision may be achieved.

The 1-step walks seem to be failing because they retrieve too few images. Another way of seeing this is in a precision-recall curve (which we calculate to rank 1000 although our pool depth is only 20, assuming unjudged are irrelevant). At the leftmost points of Figure 7 the 1-step walks have higher precision than other approaches. However, they have lower precision at all other points in this interpolated precision-recall graph, being unable to recall our set of known-relevant images. The figure also compares forward walks to backward walks, with backward walks superior in every case. Figure 8 shows the effect of switching self-transition probability from 0 to 0.9. The 101-step walk moves from being a poor performer to one of the best three.

For comparison, we also show the curve for the 'dist' baseline, which lists images in order of graph distance from the query. Although it starts well, its performance drops at higher recall points. There is a clear gap between this simple method and the best random walk approaches, which are able to rank images based on the structure of the graph, rather than simply distance.

5. DISCUSSION AND CONCLUSION

We have applied a Markov random walk model to the click graph, giving us a high-quality ranking of documents for a given query, including those that are as-yet unclicked for that query. In practice, the approach enables us to show users more images that are relevant than previously possible. In some cases, users will click on the result, creating a new edge in the graph. In this way, applying diverse ranking methods can address the sparsity of the click graph.

A backward walk was more effective than a forward walk in every case (Figure 7). This supports the notion underlying our backward walk, modeling the inverse of the query formulation process. The user moves from an imagined doc-

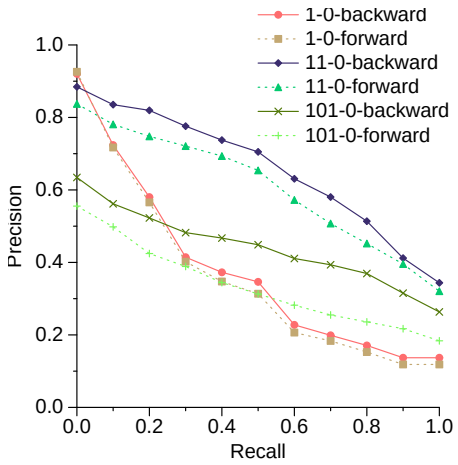

Figure 7: Precision-recall curves of forward and backward walks, with a self-transition probability of zero.

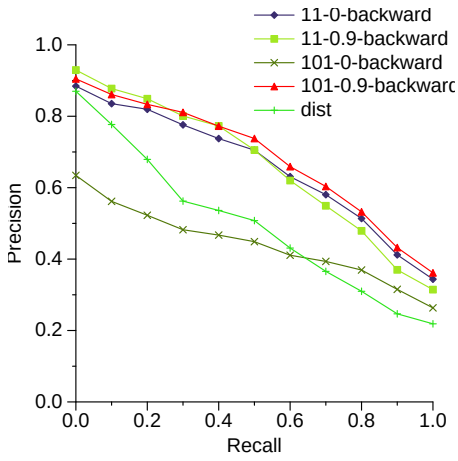

Figure 8: Precision-recall curves of backward walks with and without self-transitions. The control run 'dist' is also included.

ument to a query via some noisy process. Given the user's query, we infer their likely starting point.

As for the length of the walk, we got the best results from a walk of 11 steps, or a walk of 101 steps with high self-transition probability. In general, a longer walk can find query-document pairs connected by many different paths. We believe that this makes it more robust to noise than a purely distance-based ranking ('dist').

Tuning the parameters is beyond the scope of the current study, and we also have an insufficient number of queries to perform a training-test split. However, after finalizing the results of our main experiments (i.e. this is not tuning) we analyzed the parameter sensitivity of MAP@20 for a backward walk. Figure 9 indicates that the model performs well for a wide range of parameters, for our test set of 45 queries. The best results (the white region) are bounded to the right by a diagonal line representing a fairly constant mean length of no more than 3–10 non-self transitions. The left vertical edge shows that good results require walks of at least 5 steps, in order to reach enough nodes. 5-step walks appear relatively insensitive to self-transition probability: the 5-step walk with 0.98 self-transition probability has a mean length of only 0.1 non-self transitions, but the walk still ranks all nodes within 5 steps away.

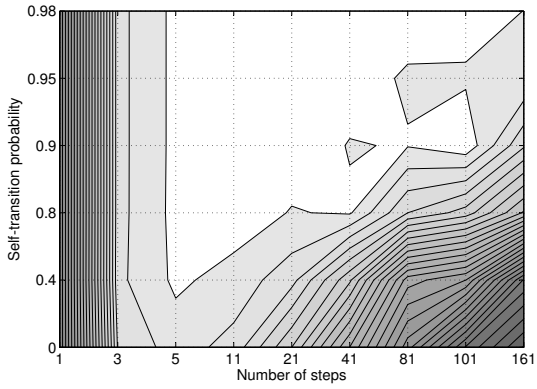

Figure 9: Parameter sensitivity for a backwards walk. Each contour shows a 0.01 variation in MAP@20. Grid intersections indicate the parameter combinations tried. The large plateau has the highest MAP@20 (0.56–0.57).

Another post-finalization experiment was to modify the transition probabilities. As noted in Section 3.5 we tried a model that normalizes document-query transitions differently, giving less preference to popular queries. However, this led to a slight drop in performance. We also tried assigning uniform probability to all non-self transitions, which gave us a 5–10% drop in P@20. This indicates that our use of click counts is helpful but not essential. In future work we could try other transition models, and perhaps even develop separate query-document and document-query transition models.

We have studied adhoc retrieval in this paper. However, we believe this approach — particularly a long backward random walk with high self-transition probability — could be effective in many of the applications listed in Section 2. We give an example of annotation in Figure 10. Given an image k, queries are ranked according to $P_{0|t}(q \mid k)$, the probability that we started at query q, given that we ended a t-step walk at k. The image is of a boxer puppy. We see that the top 10 queries given by the backward random walk all contain both the concept 'puppy' and 'boxer', while adding new vocabulary to the image such as 'pups' and 'baby'. One of the associated queries is at distance 5 from the given image. By contrast, the ranking based purely on distance tends to find queries that are 'boxer' or 'puppies' but seldom both.

We could imagine applications beyond those listed in Section 2. In the case of relevance feedback, the set of given nodes might be mixed, including both documents and the user's original query. In the query-to-document case, we might start from multiple query nodes. For example, taking a large set of 'adult' queries as input, and producing a labeling of documents, for use in adult filtering.

We also have a prototype based on clicks from Web document engines, rather than image engines. Early indications

$k=$ 

```
Annotation using a random walk:
 P      Query                  Distance
0.075   boxer dog puppies       3
0.066   boxer puppy pics        3
0.060   boxer puppies           1
0.056   puppy boxer             3
0.056   boxer puppy pictures    3
0.049   boxer pups              3
0.049   boxer puppy             3
0.038   puppy boxers            5
0.034   boxer pup               3
0.030   baby boxer              3

Annotation using distance alone:
        Query                  Distance
        boxer puppies           1
        boxer dog               3
        boxer dog puppies       3
        boxer                   3
        pictures of boxer dogs  3
        akc puppies pics        3
        boxer dogs              3
        boxer pups              3
        boxers dog              3
        puppies for sale        3
```

Figure 10: Annotation. The given image k has only been clicked for one query 'boxer puppies', which can be interpreted as annotation [14]. We employ a 101-0.9-backward walk to find ten annotations, as well as showing ten annotations based on graph distance alone.

are promising that the system can be effective, even without any special noise reduction. This is perhaps not surprising, given positive results of previous studies using Web document co-click information, such as Xue *et al* [15].

Given our probabilistic model, another possible step would be to incorporate document content and query content, by incorporating a language model. This could further extend the reach of our click-based models, in particular when language models can be applied to find relevant documents that are not yet part of the click graph.

6. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM Press.

[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pages 3–10, New York, NY, USA, 2006. ACM Press.

[3] R. Baeza-Yates, C. Hurtado, M. Mendoza, and G. Dupret. Modeling user search behavior. In *LA-WEB '05: Proceedings of the Third Latin American Web Congress*, page 242, Washington, DC, USA, 2005. IEEE Computer Society.

[4] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM Press.

[5] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, 2005.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[7] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM Press.

[8] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR 01*, pages 111–119, 2001.

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Mach. Intell. (PAMI)*, 22(8):888–905, Aug. 2000.

[10] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 945–952. MIT Press, Jan. 2002.

[11] N. Tishby and N. Slonim. Data clustering by Markovian relaxation and the information bottleneck method. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 640–646, 2001.

[12] K. Toutanova, C. D. Manning, and A. Y. Ng. Learning random walk models for inducing word dependency distributions. In *Intl. Conf. Machine Learning (ICML)*, 2004.

[13] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 162–168, New York, NY, USA, 2001. ACM Press.

[14] L. Wenyin, S. Dumais, Y. Sun, H. Zhang, M. Czerwinski, and B. Field. Semi-automatic image annotation. *INTERACT2001, 8th IFIP TC. 13 Conference on Human-Computer Interaction*, 2001.

[15] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126, New York, NY, USA, 2004. ACM Press.