

# Web Page Rank Prediction with PCA and EM Clustering

Polyxeni Zacharouli<sup>1</sup>, Michalis Titsias<sup>2</sup>, and Michalis Vazirgiannis<sup>1</sup>

<sup>1</sup> Univ. of Economics and Business, Athens, Greece,  
{zaharouli06,mvazirg}@aueb.gr,

<sup>2</sup> School of Computer Science, University of Manchester, UK  
mtitsias@cs.man.ac.uk

**Abstract.** In this paper we describe learning algorithms for Web page rank prediction. We consider linear regression models and combinations of regression with probabilistic clustering and Principal Components Analysis (PCA). These models are learned from time-series data sets and can predict the ranking of a set of Web pages in some future time. The first algorithm uses separate linear regression models. This is further extended by applying probabilistic clustering based on the EM algorithm. Clustering allows for the Web pages to be grouped together by fitting a mixture of regression models. A different method combines linear regression with PCA so as dependencies between different web pages can be exploited. All the methods are evaluated using real data sets obtained from Internet Archive, Wikipedia and Yahoo! ranking lists. We also study the temporal robustness of the prediction framework. Overall the system constitutes a set of tools for high accuracy pagerank prediction which can be used for efficient resource management by search engines.

## 1 Introduction

The ranking of query results in a Web search-engine is an important problem and has attracted significant attention in the research community. In order to maintain timely and accurate web pages ranking a huge crawling and indexing infrastructure is needed. On the other hand continuous crawling of the Web is almost impossible due to its dynamic nature, the large number of Web pages and bandwidth constraints. The same holds for the indexing and page ranking process, since the computations needed are very expensive. Thus the need for *rank prediction* arises.

In this paper we describe learning algorithms for Web page rank prediction. We consider linear regression models and combinations of regression with probabilistic clustering and Principal Components Analysis. These models are learned from time-series data sets and can predict the ranking of a set of Web pages in some future time. Each training data set corresponds to preprocessed rank values of the Web pages observed in previous time points. A first algorithm uses separate linear regression models so as the ranking evolution process of each Web page is explained independently from the corresponding processes of other

Web pages. This simple technique is then combined with probabilistic clustering based on the EM algorithm. Clustering allows for the Web pages to be grouped together by fitting a mixture of regression models. A third method combines linear regression with dimensionality reduction by applying *PCA*. This *PCA*-based method can exploit dependencies between different web pages so as the rank predicted values of all Web pages are correlated with each other. All of the above mentioned methods are evaluated using real data sets obtained from Wikipedia, Internet Archive and Yahoo!.

The remainder of the paper is as follows. Section 2 presents related work, while section 3 discusses the data preprocessing of the rank values by using normalized ranking. Section 4 describes learning algorithms for predicting the Web page ranking, while sections 5 and 6 present the evaluation measures and the experimental study. The paper concludes with a discussion in section 7.

## 2 Related work

The problem of predicting PageRank is partly addressed in [9]. This work focuses on Web page classification based on URL features. Based on the proposed framework, the authors perform experiments trying to make PageRank predictions using the extracted features. For this purpose, they use linear regression; however, the complexity of this approach grows linearly in proportion to the number of features used. The experimental results show that PageRank prediction based on URL features does not perform very well, probably because even though these features correlate very well with the subject of pages, they do not influence the authority of the page in the same way. A recent approach towards page ranking prediction is presented in [1], generating Markov Models from historical ranked lists and using them for predictions.

An approach that aims at approximating PageRank values without the need of performing the computations over the entire graph is that of Chien et al. [10]. The authors propose an algorithm to incrementally compute approximations to PageRank, based on the evolution of the link structure of the Web graph. Given a set of link changes, they identify a small portion of the Web graph in the vicinity of these changes, and model the rest of the Web as a single node in this small graph. Then they compute a version of PageRank on the reduced graph and transfer these results to the original graph. Their experiments demonstrate that the algorithm performs well both in speed and quality and is robust to various types of link modifications. This approach, however, requires the continuous monitoring of the Web graph in order to track any link modifications. There has also been work in adaptive computation of PageRank [11, 14] or even estimation of PageRank scores [12].

In [16], Yang et al. propose a method called *predictive ranking*, aiming at estimating the Web structure based on the intuition that the crawling and consequently the ranking results are inaccurate (due to inadequate data and dangling pages). In this work, the authors do not make future rank predictions. Instead, they estimate the missing data in order to achieve more accurate rankings.

### 3 Normalized ranking

In order to predict future rankings of Web pages, we need to define a measure that effectively expresses the trends of Web pages among different snapshots of the Web graph. We have adopted a measure (*nrank*), introduced in [2] suitable for measuring page rank dynamics. We briefly present its design. Let  $G_{t_i}$  be the snapshot of the Web graph created by a crawl at time  $t_i$  and let  $n_{t_i} = |G_{t_i}|$  the number of Web pages at time  $t_i$ . We define  $rank(p, t_i)$  as a function providing the ranking at the time  $t_i$  of a Web page  $p \in G_{t_i}$  according to some criterion, for example PageRank values. Intuitively, an appropriate measure for Web pages trends is the rank change rate between two snapshots, but as the size of the Web graph constantly increases the trend measure should be comparable across different graph sizes. A way to deal with this problem is to consider the normalized rank (*nrank*) of a Web page. We impose that the *nrank* of all pages in a ranked list sum up to 1. If  $N$  is the normalizing factor and  $n$  are all the time stamps for which we have crawled the Web graph, then:

$$\sum_{i=1}^n nrank(p, t_i) = 1 \Rightarrow \frac{\sum_{i=1}^n i}{N} = 1 \Rightarrow \frac{n_{t_i}(n_{t_i} + 1)}{2N} = 1.$$

Thus, the *nrank* of a page  $p$  at  $rank(p, t_i)$  is:

$$nrank(p, t_i) = \frac{2 \cdot rank(p, t_i)}{n_{t_i}^2}. \quad (1)$$

(We assume here  $n_{t_i} \gg 1$  and thus do not distinguish between  $n_{t_i}$  and  $n_{t_i} + 1$ .) The *nrank* ranges between  $2n_{t_i}^{-2}$  and  $2n_{t_i}^{-1}$ .

## 4 Methods

### 4.1 Separate linear regression models

Assume a set of  $n$  Web pages for which we observe the *nrank* values at  $m$  times steps. Let  $x_i = (x_{i1}, \dots, x_{im})$  be the *nrank* values for the  $i$ th Web page at the time points  $t = (t_1, \dots, t_m)$ . Further, we assume that the  $n \times m$  design matrix  $X$  stores all the observed *nrank* values so as each row corresponds to a Web page and each column to a time point. Given these observations we wish to predict the *nrank* value  $x_{i*}$  for each Web page  $i$  at some time  $t_*$ .  $t_*$  will typically correspond to a future time point, i.e.  $t^* > t_i$ , with  $i = 1, \dots, m$ . Next we discuss a simple prediction method based on linear regression where the input variable correspond to time and the response variable is the *nrank* value.

For a certain Web page  $i$  we assume a linear regression model having the following form

$$x_{ik} = a_i t_k + b_i + \epsilon, \quad k = 1, \dots, m, \quad (2)$$

where  $\epsilon$  denotes a zero-mean Gaussian noise. Note that the parameters  $(a_i, b_i)$  are Webpage-specific. In other words, the above formulation defines a separate

linear regression model for each Web page and thus Web pages are treated independently. The specification of the values  $(a_i, b_i)$  is achieved by solving a linear system using least squares. Prediction of the nrank value  $x_{i*}$  at some future time  $t_*$  is carried by evaluating Equation (2) and ignoring the  $\epsilon$  noise term.

The linear model can be easily extended to account for simple non-linearities. For instance, Equation (2) can be replaced by a quadratic function over  $t_k$  or a higher order polynomial function. In the experiments we investigate a linear and a quadratic regression model.

The above framework treats each Web page independently from the remaining Web pages by learning a separate linear model. This can be restrictive since any global similarities and dependencies that might exist between different Web pages are not taken into account. In the next two sections we generalize the above baseline method in two ways. In section 4.2, we cluster the linear regression models using the EM algorithm in order to capture similarities between Web pages, while in section 4.3 we apply PCA in order to model inter-dependencies among the different Web pages.

## 4.2 Clustering using EM

We assume that the nrank values of each Web page follow one of  $J$  different types or clusters. The number of clusters  $J$  can be much smaller than the number  $n$  of Web pages. Clustering can be viewed as training a mixture probability model. To generate the nrank values  $\mathbf{x}_i$  for the  $i$ th Web page, we first select the cluster type  $j$  with probability  $\pi_j$  (where  $\pi_j \geq 0$  and  $\sum_{j=1}^J \pi_j = 1$ ) and then produce the values  $\mathbf{x}_i$  according to a linear regression model:

$$x_{ik} = a_j t_k + b_j + \epsilon_k, \quad k = 1, \dots, m \quad (3)$$

where  $\epsilon_k$  is independent Gaussian noise with zero mean and variance  $\sigma_j^2$ . The above formulation implies that given the cluster type  $j$  the nrank values are drawn from the following product of Gaussians:  $p(x_i|j) = \prod_{k=1}^m N(x_{ik}|a_j t_k + b_j, \sigma_j^2)$ . The cluster type that generated the nrank values of a certain Web page is an unobserved variable and thus after marginalization we obtain a mixture unconditional density for the observation vector  $x_i$ :  $p(x_i) = \sum_{j=1}^J \pi_j p(x_i|j)$ . To train the mixture model and estimate the parameters  $theta = (\pi_j, \sigma_j^2, a_j, b_j)_{j=1}^J$ , we can maximize the log likelihood of the data  $L(theta) = \log \prod_{n=1}^N p(x_i)$  by using the EM algorithm [4]. Given an initial state for the parameters, EM optimizes over  $theta$  by iterating between E and M steps:

Once we have obtained suitable values for the parameters, we can use the mixture model for prediction. Particularly, to predict the nrank value  $x_{i*}$  of the  $i$ th Web page at time  $t_*$  given the observed values  $x_i = (x_{i1}, \dots, x_{im})$  at previous times we express the posterior distribution  $p(x_{n*}|x_i)$  using the Bayes rule:

$$p(x_{i*}|x_i) = \sum_{j=1}^J R_j^i N(x_{i*}|a_j t_* + b_j, \sigma_j^2), \quad (4)$$

where  $R_j^i$  is computed according to  $R_j^i = \frac{\pi_j p(x_i|j)}{\sum_{\rho=1}^J \pi_\rho p(x_i|\rho)}$  for  $j = 1, \dots, J$  and  $i = 1, \dots, N$ . To obtain a specific predictive value for  $x_{i^*}$  we can use the mean value of the above posterior distribution  $x_{i^*} = \sum_{j=1}^J R_j^i (a_j t_* + b_j)$  or the median estimate  $x_{i^*} = a_j t_* + b_j$  where  $j = \arg \max_\rho R_\rho^j$  that considers a hard assignment of the Web page into one of the  $J$  clusters.

### 4.3 PCA and regression

Although the EM approach models global similarities between different Web pages by grouping them into clusters, the dependencies among different Web pages are not well modeled. To see this, note that the likelihood that generates the data factorizes across all Web pages. Dependencies between Web pages can naturally exist in the time-evolution ranking process. This is because the popularity of a certain Web page can affect the popularity of other Web pages. In this section we discuss a prediction method that takes into account such dependencies by applying dimensionality reduction.

The  $n \times m$  design matrix  $X$  of the observed nrank values can be viewed as a times-series instantiation of the large column vector of  $n$  measurements corresponding to all Web pages. Based on this, we can consider the  $m$  columns of  $X = [X_1, \dots, X_m]$ , where each  $X_k$  is a  $n$ -dimensional vector, as the observed data vectors and apply a dimensionality reduction technique. The promise of this formulation is that it allows us to model the correlation structure in the elements of  $X_k$  that reflects the possible inter-dependencies among Web pages. Note that this modeling view is rather different to that used in section 4.2 (and 4.1) where we considered a probability model to generate each row of  $X$ . Next we use PCA as the dimensionality reduction method and combine it with linear regression.

PCA is a well-established technique for data processing [5]. Although there are many ways to motivate the PCA framework, here we focus on the probabilistic view that is widely used in statistical machine learning [3, 6]. Each  $n$ -dimensional vector  $X_k$  of nrank values at time  $t_k$  is generated according to a latent variable model so as:

$$X_k = Az_k + \mu + \epsilon, \quad (5)$$

where the  $n \times q$  matrix  $A$  relates the  $q$ -dimensional latent vector  $z_k$  with the observed high dimensional vector  $X_k$  and  $\mu$  is the mean of  $X_k$ s. The noise vector  $\epsilon$  is drawn from a zero-mean isotropic Gaussian, i.e.  $\epsilon \sim N(0, \sigma^2 I_n)$ , while the latent vector  $z_k$  is drawn from  $N(0, I_q)$ . From Equation (5), we can easily marginalize out the latent variable  $z_k$  and obtain the unconditional probability distribution of  $X_k$ :  $p(X_k) = N(\mu, AA^T + \sigma^2 I)$ . The maximization of the likelihood with respect to the transformation matrix  $A$  gives rise to PCA [3], so as the columns of  $A$  will be equal to the  $q$  largest eigenvectors of the empirical covariance matrix.

We now discuss how we adapt the PCA framework to Web page rank prediction. The probability distribution in the previous equation is a Gaussian with

full covariance matrix  $AA^T + \sigma^2 I$  which implies that the correlation of different Web pages in the nrank vector  $X_k$  is modeled. Our objective is to learn how to predict  $X_*$  at future time  $t_*$ . We follow a two-stage learning process. First, we use the matrix  $X$  to find the  $q$  principal components  $A$  and we form the low dimensional projection of the data according to  $z_k = A^T(X_k - \mu)$ , for  $k = 1, \dots, m$ . Then, we train  $q$  independent linear regression models in the low dimensional space similarly to section 4.1. More specifically, assuming that  $Z$  is a  $q \times m$  matrix that collects together all the latent vectors, each linear regression model is constructed for any row of  $Z$ :

$$z_{jk} = a_j t_k + b_j + \epsilon, \quad k = 1, \dots, m \quad (6)$$

and  $j = 1, \dots, q$ .

The parameters  $(a_j, b_j)$  for each  $j$  are obtained by solving a linear system using least squares. To predict the nrank values  $X_*$ , we firstly estimate the latent vector  $z_*$  at time  $t_*$  by using Equation (6) and then we predict  $X_*$  according to the expression  $X_* = Az_* + \mu$ .

## 5 Top-k lists similarity measures

In order to evaluate the quality of predictions we need to measure the similarity of the predicted to the actual top- $k$  rankings. For this purpose, we employ measures commonly used for comparing rankings.

The first one, denoted as  $OSim(A, B)$  [13] indicates the degree of overlap between the top- $k$  elements of two sets  $A$  and  $B$  (each one of size  $k$ ):

$$OSim(A, B) = \frac{|A \cap B|}{k} \quad (7)$$

The second,  $KSim(A, B)$  [13] is based on Kendall's distance measure [15] and indicates the degree that the relative orderings of two top- $k$  lists are in agreement:

$$KSim(A, B) = \frac{|\{(u, v) : A', B' \text{ agree on order}\}|}{|A \cup B|(|A \cup B| - 1)} \quad (8)$$

where  $A'$  is an extension of  $A$  resulting from appending at its tail the elements  $x \in A \cap (B - A)$ . The added elements are shuffled such that they do not preserve their original relative ordering in list  $B$  ( $B'$  is defined analogously). In other words,  $KSim(A, B)$  is the probability that  $A'$  and  $B'$  agree on the relative ordering of a randomly selected pair of distinct nodes  $(u, v) \in |A \cup B| \times |A \cup B|$ .  $OSim$  indicates the concurrence of predicted pages with the actual visited ones, but does not take into consideration the ranks of the pages in the final rankings.  $KSim$  takes into consideration the common items of the two lists, and is proportional to the number of pairs that have the same relative ordering in both lists.

## 6 Experiments

In order to evaluate the effectiveness of our approach, we performed experiments on three different real-world datasets. The first is a subset of the European Archive<sup>3</sup>, consisting of 22 Web graph snapshots of UK government Websites. The second one is the Web graph of the English version of the Wikipedia encyclopedia<sup>4</sup>. The third one is a collection of top- $k$  ranked lists for 22 queries over a period of 37 days, as they result from Yahoo! search engine<sup>5</sup>. In our experiments, we evaluate the prediction quality in terms of similarities between the predicted top- $k$  ranked lists and the actual ones using the *OSim* and *KSim* similarity measures. All the experiments were run on low-end commodity hardware (3 GHz Pentium PCs with 2 GB of memory and local IDE disk, running either Windows XP or Linux).

### 6.1 Datasets Description and Preprocessing

For each dataset (Internet Archive, Wikipedia and Yahoo!), a wealth of snapshots were available, ensuring that we have enough evolution to test our approach. A concise description of each dataset follows. We also present in detail our query-based approach on Wikipedia and Yahoo!.

The Internet Archive dataset comprises of approximately 500,000 pages, referring to weekly collections of eleven UK government Websites. We obtained 22 graph snapshots evenly distributed in time between March 2004 and January 2006.

The Wikipedia dataset consists of 55 consecutive monthly snapshots of the English version of Wikipedia, ranging from January 2002 to July 2006. It represents a dynamic large-scale collection (grew from 4,593 to 1,467,982 articles between the first and last snapshots), frequently used as a benchmark dataset<sup>6</sup>. The snapshots were extracted from a database dump containing the entire history of the encyclopedia, which can be found at <http://download.wikipedia.org/>. We use the Wikipedia dataset and its evolution features to verify the validity of our prediction framework on query-based top- $k$  results. This is an intuitive choice as this is a natural way of producing top- $k$  results on the Web. Thus we maintain for each selected query (selection criteria follow) the top- $k$  results, for each of the 55 monthly snapshots. For some of the queries we have less top- $k$  lists, some query terms do not yield any results for some of the initial snapshots.

The Yahoo! dataset consists of 37 consecutive daily top-1000 ranked lists as they are computed using the Yahoo! Search Web Services<sup>7</sup>. We have collected such top- $k$  rankings for 22 queries. There is an overlap between this query set and the Wikipedia one.

<sup>3</sup> <http://www.europarchive.org/ukgov.php>

<sup>4</sup> <http://en.wikipedia.org/>

<sup>5</sup> <http://search.yahoo.com/>

<sup>6</sup> [http://en.wikipedia.org/wiki/Wikipedia:Wikipedia\\_in\\_academic\\_studies](http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_in_academic_studies) is a regularly updated list of relevant works

<sup>7</sup> <http://developer.yahoo.com/search/>

**Ranking Function for Querying Wikipedia** Regarding the ranking function in the Wikipedia top- $k$  lists, we take into account two features: *a*) the PageRank value of the page containing the term and *b*) the tf-idf value of the term for the specific page. Thus, given a term  $t$ , the score of document  $d$  is computed as follows:

$$score_t(d) = (tf-idf(t, d) \cdot title(t, d))^{1.5} \cdot pr(d) \quad (9)$$

with  $title(t, d) = 2$  if  $t$  appears in the title of  $d$ , 1 otherwise, and  $pr(d)$  the graph based PageRank score for the page  $d$ .

Both *tf-idf* and *title* are lifted to the power 1.5 to increase the importance of high tf-idf values, and decrease the importance of low tf-idf values, applying thus implicitly a smooth thresholding strategy. The choice of the aforementioned score function and parameter values was based on both an anecdotal evaluation of the top- $k$  resulting lists and a comparison to the respective top- $k$  lists produced by querying Wikipedia with Google—thus implicitly assuming Google’s ranking as a reference ones. Query results were reasonably comparable to Google’s ranking, and it is not the intent of this work to devise a very elaborate scoring function.

**Query selection for Wikipedia and Yahoo!** In the case of Wikipedia, we classify query terms (and therefore the respective top- $k$  lists) based on their *frequency* but also on their *dynamism*. Frequency is measured by the proportion of the documents containing a given term. Another property we are interested in is the *dynamism* of the query in terms of the rate of increase (or decrease) in the number of documents that contain the query time as time progresses. We define the previous features as follows: Let  $W = \{w_i\}$  be the set of Wikipedia snapshots, ordered in time (i.e.,  $w_{i+1}$  is the immediate successor of  $w_i$ ). Let  $TF_{w_i}(t_j)$  be the term frequency of term  $t_j$  in snapshot  $w_i$  (i.e., the number of documents  $t_j$  appears in, in snapshot  $w_i$ ). For each  $w_i$  in  $W$  and each  $t_j$  in  $w_i$  we compute  $TF_{w_i}(t_j)$ . Then we define dynamism rate of the term for the snapshot  $w_i$  as:

$$SD_{w_i}(t_j) = \frac{(TF_{w_i}(t_j) - TF_{w_{i-1}}(t_j))}{TF_{w_i}(t_j)} \quad (10)$$

and its aggregate value over the snapshots series:  $TSD(t_j) = TSD(t_j) + SD_{w_i}(t_j)$ . Then we compute for each term its *average dynamism*:  $D(t_j) = TSD(t_j) / (|W| - 1)$ , where  $|W|$  represents the number of snapshots the term is present in.

In the case of the Yahoo! dataset, since we have no access to information as for Wikipedia, we picked *a*) random queries from the Wikipedia dataset, *b*) popular queries that appeared in Google Trends (<http://www.google.com/trends>) and *c*) popular current queries (such as “euro 2008” or “olympic games 2008”). All queries used for Wikipedia and Yahoo! can be found at: [www.db-net.aueb.gr/michalis/WAW2009/queries.rar](http://www.db-net.aueb.gr/michalis/WAW2009/queries.rar)

## 6.2 Experimental methodology

We compare all predictions to a baseline scheme, called *Static*, which just returns the top- $k$  list of the previous snapshot (i.e., we consider that the pages remained

at the same rank). The steps we followed for all three datasets, follow. We first computed PageRank scores for each snapshot of our datasets. In order to obtain the global top- $k$  rankings in the case of Internet Archive, we ordered the pages using PageRank scores for each snapshot in a descending order. In a similar way, we obtained the Wikipedia top- $k$  rankings using the scoring function mentioned before. After computing the PageRank scores, we calculated the  $nrank$  values for each pair of consecutive graph snapshots and we stored them in a  $nrank \times time$  matrix (where rows represent the different Web pages and columns all available snapshots). Assuming an  $m$ -path of consecutive snapshots we predict the  $m + 1$  state. Then for each page  $p$  we predict a ranking which is compared to its actual ranking after using a 10-fold cross validation process: we use 90% of the dataset as training and the learned models are tested on the remaining 10%. This process is repeated 10 times each time with a different 10% test fold.

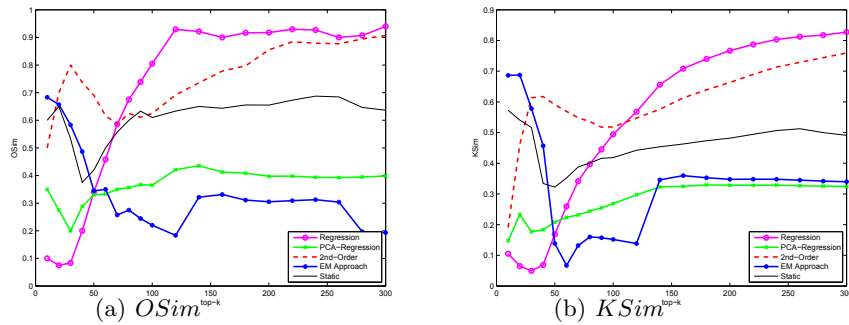
In the case of the EM approach, we tested the quality of clustering results for clusters cardinality between 2 and 10, for each query and chose the one that maximized the overall quality of the clustering. The overall quality of clustering (or score function) was defined as a monotone combination of the within-cluster variation and the between-cluster variation. A simple measure of within-cluster variation is the sum of squares of distances from each point to the center of the cluster it belongs to:  $w_c = \sum_{k=1}^J \sum_{x \in cluster_k} d(x, r_k)$  where  $r_k$  is the center of cluster  $k$ ,  $J$  is the total number of clusters (where  $2 \leq J \leq 10$ ) and  $d(x, r_k)$  is the simple Euclidean distance. Between-cluster variation can be measured by the distance between cluster centers:  $b_c = \sum_{1 \leq j < k \leq J} d(r_j, r_k)$ . The score function of clustering was then defined as the ratio  $b_c/w_c$ .

In the case of the PCA method combined with regression, we projected the data into the space formed by the  $k$  principal eigenvectors in such a way as to capture the 85% of the variance in the original data. The variance of the projected data can be expressed as  $\sum_{j=1}^k \lambda_j$ , where  $\lambda_j$  is the  $j$ -th eigenvalue. Equivalently, the squared error in terms of approximating the true data matrix using only the first  $k$  eigenvectors (where the total number of eigenvectors is  $p$ ) can be expressed: as:  $\sum_{j=k+1}^p \lambda_j / \sum_{l=1}^p \lambda_l$ .

### 6.3 Experimental Results

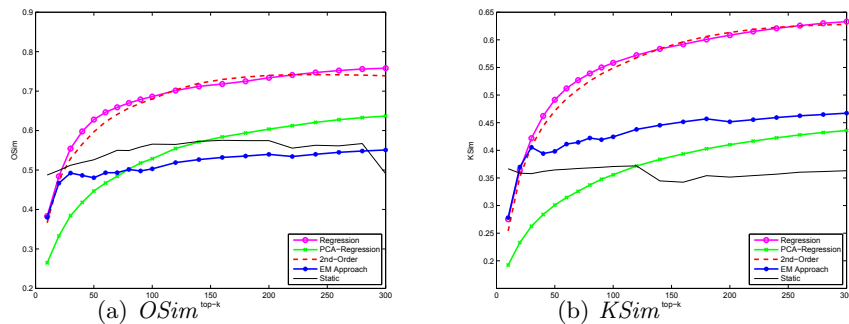
In this section we evaluate the prediction performance of our framework involving all three datasets: Internet Archive, Wikipedia and Yahoo!. Assuming that we know the web pages'  $nrank$  values for  $n$  time steps, we use the first  $n - 3$  time steps for training the prediction models and the static baseline, while the last three actual  $nrank$  values are used for comparing them to the ones we predicted. The results of our experiments are described in more detail in the following subsections.

**Internet Archive** In Fig. 1 we report results for the Internet Archive data set. The EM approach achieves the best performance for small values of  $k$  ( $k < 30$ ) achieving a prediction accuracy of 0.65 – 0.70. Then its performance degrades



**Fig. 1.** Prediction accuracy vs top-k list length: Internet Archive dataset

dramatically as the top- $k$  list's size grows. The static approach performs very well for small  $k$  values, then it degrades and increases again to a 0.6 accuracy figure. The static baseline is a good choice for this data set due its relative lack of dynamism (governmental sites change less frequently than the commercial ones). The best performing method for small  $k$  values (i.e. for the most meaningful rankings) is the 2nd order regression which achieves a 0.7 top 0.8 prediction accuracy for  $OSim$  and for  $k$  in  $[20, 50]$ . Then it slightly degrades and increases again to reach an outstanding 0.86 accuracy for larger values of  $k$ . The plain regression technique performs really bad for small  $k$  values while its performance increases exponentially with  $k$  and surpasses that of all other approaches for  $k \geq 80$ . On the other hand the PCA combined with regression methods perform consistently worse than all the others for most  $k$  values. This performance pattern remains for the  $KSim$  measure although the accuracy figures are somewhat smaller.



**Fig. 2.** Prediction accuracy vs top-k list length: Wikipedia dataset

**Wikipedia** We conducted a large series of experiments for the Wikipedia English dataset, using an one-month time interval between the 55 successive snapshots. We show in Fig. 2 the performance of the predicting schemes for different values of  $k$ , for all employed similarity measures ( $OSim, KSim$ ). The query load we used consists of 112 queries. The accuracy appearing in the figure is the

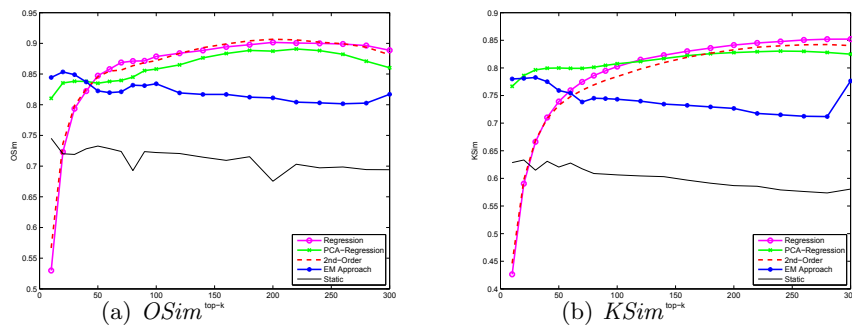
average value for all 112 queries, using 10– fold cross validation ensuring thus robustness and reliability of the results.

All approaches perform badly for small values of  $k$  ( $k < 50$ ). Interestingly the static approach has the best (although objectively bad) performance for  $k \leq 20$ . For larger  $k$ 's all approaches ameliorate drastically and outperform the static one with linear and 2nd order regression being the best approaches for this data set reaching a prediction accuracy 0.75. The *PCA* and *EM* approaches perform clearly worse than plain regression ones.

**Yahoo!** In Fig. 3 we report the predictions' results for the Yahoo! data set. We consider these results as most valuable since the data represent real worlds rankings to popular queries. Even though we do not have access to the ranking algorithm our task is to learn predictors for top- $k$  lists. Also we have to stress the top- $k$  lists time interval which is very short for Yahoo! as data are collected daily.

The results are very encouraging with regards to the predictions. The EM approach excels for small values of  $k$  (i.e. the most interesting ones among real users) achieving a prediction accuracy of 0.8 - 0.85 depending on the similarity measure. The performance slightly degrades as the top- $k$  list's size grows.

On the other hand the PCA combined with regression approach performs very good for small  $k$  values (although a bit worse than EM) while its performance increases slightly and outperforms the EM approach for larger  $k$  values. The plain regression techniques perform really bad for small  $k$  values while their performance increases exponentially with  $k$  and surpasses that of all other approaches for  $k \geq 50$  (100) for the *OSim* (and *KSim*) similarity measure reaching an outstanding 90% prediction accuracy.



**Fig. 3.** Prediction accuracy vs top- $k$  list length: Yahoo! dataset

## 7 Conclusions

The dynamics of the Web provide a fascinating domain of study for researchers from both academic and commercial fields. Searching the Web inherently involves

the ranking issue. In this paper, we proposed learning algorithms for Web page rank prediction based on combinations of linear regression, clustering and PCA. We capitalize on previous Web page rankings and built a family of predictors. We conducted extensive experiments on large scale real Web data (Internet Archive), as well as on query-based data from the English Wikipedia and Yahoo! query results. We studied the prediction performance for various query types and query loads. statistically robust as they result from a 10– fold cross-validation process. The results are overall encouraging, for all similarity measures across all datasets used achieving high prediction accuracy.

*Acknowledgements.* We warmly thank Klaus Berberich, Pierre Sennelart and Thimios Kostakis for their suggestions and help at previous stages of this work.

## References

1. M. Vazirgiannis, D. Drosos, P. Sennelart, and A. Vlachou, Web Page Rank Prediction with Markov Models, WWW poster, Beijing, China, 2008.
2. A. Vlachou, K. Berberich, and M. Vazirgiannis, Representing and quantifying rank-change for the Web graph, In *Proc. WAW*, Banff, Canada, Nov. 2006.
3. M.E. Tipping and C.M. Bishop, Mixtures of probabilistic principal component analyzers, *Neural Computation*, 11: 443–482, 1999.
4. A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
5. I. T. Jolliffe, *Principal Component Analysis*, New York: Springer-Verlag, 1986.
6. C.M. Bishop, *Machine learning and pattern recognition Information Science and Statistics*, Springer, 2006.
7. K. Järvelin and J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, *TOIS*, 20(4):422–446, Oct. 2002.
8. C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
9. M.-Y. Kan and H. O. N. Thi, Fast webpage classification using URL features, In *Proc. CIKM*, Bremen, Germany, Oct. 2005.
10. S. Chien, C. Dwork, R. Kumar, D. R. Simon, and D. Sivakumar, Link evolution: Analysis and algorithms, *Internet Mathematics*, 1(3):277–304, 2003.
11. A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen, Efficient PageRank approximation via graph aggregation, *Information Retrieval*, 9(2): 123–138, Mar. 2006.
12. Y.-Y. Chen, Q. Gan, and T. Suel, Local methods for estimating PageRank values, In *Proc. CIKM*, Washington, USA, Nov. 2004.
13. T. H. Haveliwala, Topic-sensitive PageRank, In *Proc. WWW*, Honolulu, USA, May 2002.
14. A. N. Langville and C. D. Meyer, Updating PageRank with iterative aggregation, In *Proc. WWW*, New York, USA, May 2004.
15. M. G. Kendall and J. D. Gibbons, *Rank Correlation Methods*, Charles Griffin, London, UK, 1990.
16. H. Yang, I. King, and M. R. Lyu, Predictive ranking: a novel page ranking approach by estimating the Web structure, In *Proc. WWW*, May 2005.