Mathematical Methods for Website Optimisation

Zhen Shao

29 September 2016
Abstract

Google search is so good that it has become ubiquitous. In contrast, the search engines used by internal websites are often not as reliable as Google. This problem is primarily due to the limited size of internal website compared to the whole Internet. Unable to find the relevant information by the search function in an internal website, frustrated users would have to follow the links and hope to find the information, and whether they could be successful depends on the link structure of the website.

Hyperlink-Induced Topic Search (HITS, also known as hubs and authorities) [7] is a link analysis algorithm that rates Web pages. It is an alternative search algorithm to PageRank, which is the main algorithm used by Google search engine. It quantifies how good a webpage is for • providing information (authorities) • providing links to information (hubs), and it achieves this by giving it two scores. The essential idea behind the algorithm is that pages having many out-links to good authorities are good hubs, while pages with many in-links from good hubs are good authorities. Together with Professor Raphael Hauser, we devised a way to adapt the HITS algorithm to identify a list of pages (about 25) that would satisfy nearly all users’ need for hubs of information, using the specific example of Pembroke College website as a data source. They include pages related to general information about Pembroke, conferences, contact, and information for students, as expected. But surprisingly, they also include pages related to jobs and accommodation. The full list of the hubs will be given in Chapter 4, and we recommend to put a clear and easy link to these pages to optimise the user experiences.

We also mention a potential reason for failure of HITS algorithm under certain web-structures, and we have used our proposed improvement in our
Acknowledgement

I want to thank for the excellent supervision provided by Professor Raphael Hauser, the ROKOS scholarship which provides funding for the project, and kindly provision of computational facilities by the mathematical institute.

Contents

1 Introduction 4

2 Mathematical Techniques 6
   2.1 Statistical Tools ................................. 6
   2.2 Low Rank Matrix Completion ..................... 7
   2.3 Singular Value Decomposition ..................... 9
   2.4 Hyper-link Induced Topic Search ................. 13

3 Detailed Approach 15
   3.1 Gathering the data ............................ 15
   3.2 Processing the data ............................ 15
   3.3 Imperfection of the data ....................... 16
   3.4 Constructing a useful data structure for analysis .... 17
   3.5 The problem of scarcity and low rank matrix completion 18
   3.6 Implementing HITS ............................ 19
   3.7 Effective Linkage structure and HITS ............ 24

4 Results and Analysis 25
   4.1 Eigen-Users and Analysis ...................... 25
   4.2 Results of HITS and Analysis ................. 28
5 Conclusion 31

6 Appendix 32
Chapter 1

Introduction

In this report, we present a way of using mathematical techniques to automatically detect usage patterns and optimise the linkage structure of an internal webpage. The specific data we worked with stems from traffic to the Pembroke College webpage, but the technique could be used to optimise any internal website.

Unlike the highly sophisticated techniques applied to a general topic search across the World Wide Web (WWW), techniques for topic searches in an internal website are still under development. Internet search engines such as Google have become so good that many people take them as the first port of call when they want to find some information on the WWW. However, when it comes to find information in a specific internal website, although such a website may provide a search function as well, our general experiences tell us that such a search may not yield accurate result, and we would click around to find our information instead.

This behaviour creates a huge challenge for website management teams. They must be able to figure out a way to present the most relevant information to users, without knowing what they are searching for. This challenge arises because users no longer enter any search term, but would only try to find the information they are seeking by the existing link structure. Under this circumstance, optimising link structure becomes very important to user experiences. We will propose a way to adapt HITS algorithm, which is an alternative to Google’s PageRank algorithm, for internal website link structure optimisation. To achieve this, we will also use techniques such as Principal Component Analysis (PCA) and Low Rank Matrix Completion.

The next chapter will focus on detailed description of mathematical techniques and algorithms we would be using to achieve our goals, interpretations of those techniques in the context of an internal website optimisation are also presented. Chapter three will be dedicated to the implementational details of our approaches, along with small samples of the data we are dealing with. Results and recommendations are presented in chapter four, and we
conclude this report in chapter five.
Chapter 2

Mathematical Techniques

2.1 Statistical Tools

In statistics, an outlier is an observation point that is distant from other observations. There is no rigid mathematical definition of what constitutes an outlier; determining whether or not an observation is an outlier is ultimately a subjective exercise. For example, in the data set \{0, 1, 2, -1, -2, 2, 1, 0, 2, 1, 1000\}, the observation 1000 is considered an outlier for most cases, but in the data set \{0, 1, 2, -1, -2, 2, 1, 0, 2, 1, 5\}, it is not entirely clear whether the observation 5 should be an outlier or not. An outlier may be due to variability in the measurement or it may indicate experimental error.

In our analysis, for each person visiting the website, we have an observation about how long they spent on each page. The median duration is 21 seconds, and the 75th percentile is 58 seconds, which means 75% of all page visits have a duration smaller than 58 seconds. However, there are big observations, on the order of several thousands, and the maximum is 5540 seconds. From a statistical point of view, they are outliers, and from a practical point of view, it is highly unlikely that someone would spend 5540 seconds, which is more than one and a half hour, actively reading a webpage. As we will see later, those outliers are produced by a technical difficulty in detecting the page visit duration for the last page visited in a session, i.e. experimental errors.

Including outlier in the data analysis can lead to misleading conclusions. For example, the mean of page visit durations is 95 seconds, which is greater than 75% of data. If we naively interpret the mean as the duration of a typical page visit, we would reach incorrect conclusions. To mitigate the effect of outliers, we introduce a technique called Winsorization.

Let D be a set of real numbers, considered as a data set. The a p\% winsorization is a transformation of the data set D, according to the following procedure:
• Calculate the \( p \)th percentile of the data set \( D \), call it \( t \).

• For each observation \( data \) in \( D \), if \( data > t \), replace \( data \) by \( t \).

2.2 Low Rank Matrix Completion

Assume we have a data matrix, \( M \), where the rows are sessions and columns are pages. For example, entry \((i, j)\) is the page visit duration, in seconds, for session number \( i \), page number \( j \). The data matrix is expected to be sparse, as a typical website would have thousands of pages, while a session would only visit a couple of them. So each row would only have a couple of non-zero entries. Pages may be unvisited for various reasons, in particular, it might simply be that the user has not found the page, instead of not being interested to it. One challenge is to predict how interested the user might be in unvisited pages, by comparing his behaviour to other users. In essence, we want to fill the matrix \( M \). That is our matrix completion problem.

One of the variants of the matrix completion problem is to find the lowest rank matrix \( X \) which matches the matrix \( M \) (of size \( n \times m \), which we wish to recover) for all entries in the set \( E \) of observed entries. The mathematical formulation of this problem is as follows:

\[
\min_{X \in \mathbb{R}^{n \times m}} \text{rank}(X)
\]

subject to

\[X_{ij} = M_{ij} \ \forall i, j \in E\]

The low rank assumption is essential for the successful recovery. We will focus on mathematical formulation here leaving the question of why we could assume a low rank matrix completion problem to the next chapter.

Unfortunately, not only is the above problem NP-hard, which means that there is no efficient algorithm to solve it, but all known algorithms for exactly solving it are doubly exponential in theory and in practice[1].

A popular alternative is the convex relaxation [3], using nuclear norm minimization, which is the following problem:

\[
\min_{X \in \mathbb{R}^{n \times m}} \|X\|_*
\]

subject to

\[X_{ij} = M_{ij} \ \forall i, j \in E\]

Where \( \|X\|_* \) is the nuclear norm of the matrix \( X \), defined by \( \|X\|_* = \sum_{i=1}^{\min\{m,n\}} \sigma_i \), where \( \sigma_i \) are singular values of \( X \) (see the next section).
It is proven that nuclear-norm minimization succeeds nearly as soon as recovery is possible by any method whatsoever. [4]

However, in any real world application, one will only observe entries corrupted at least by a small amount of noise. For example, the page visit durations we observed are at least affected by: (1) the precision of measurement (2) user’s current mode, how busy he is, etc. It has been shown under small noise corruptions, accurate recovery is still possible, and we only need to slightly modify our problem.[2] The noisy problem is:

\[
\min_{X \in \mathbb{R}^{n \times m}} ||X||_* \quad \text{subject to} \quad \sqrt{\sum_{i,j \in E} |X_{ij} - M_{ij}|^2} < \delta
\]

Where \( \delta = \sqrt{\sum_{i,j \in E} Z_{ij}^2} \). In practice, the noise level is of course unknown, but we could estimate the noise level by sample standard deviations. In particular, we set \( \delta = \text{std} \times \sqrt{k} \), where \( \text{std} \) is the standard deviation of the completed matrix \( X \), and \( k \) is the cardinality of the set \( E \). So the condition \( \sqrt{\sum_{i,j \in E} |X_{ij} - M_{ij}|^2} < \delta \) signifies that the norm of the errors are smaller than the inherited variance in the data itself, and we could view the completion as a good match. In practice, this criterion would actually ensure that \( |X_{ij} - M_{ij}| \) be very small for all \( i, j \in E \).

To solve the noisy problem, we use the FPCA (Fixed Point Continuation with Approximate SVD) algorithm [5]. We outline the steps of FPCA here and attach the code in the Appendix.(Code 6.8)

- Given \( M \), choose a decreasing sequence \( S \), of \( \mu \), real numbers, according to the largest singular value of \( M \). Let \( E \) be the index set of sampled entries. Starting from any matrix \( X \).

- Choose \( \tau \), which should be viewed as a damping parameter in the gradient descent step.

- For each value of \( \mu \) in \( S \), repeat the following steps, until the successive updates of \( X \) converges.

  - \( \forall i, j \in E, X_{ij} = X_{ij} - \tau \times (X_{ij} - M_{ij}) \)
  - Calculate SVD of \( X \), \( X = USV^T \), note \( S \) is a diagonal matrix.
  - Set a lower bounds on singular values of \( X \). For all \( \sigma_i \) in the diagonal of matrix \( S \), if \( \sigma_i < \tau \times \mu \), set \( \sigma_i = 0 \)
  - Update \( X, X = USV^T \). Note, \( S \) has been modified.
2.3 Singular Value Decomposition

We will use singular value decomposition (SVD) to identify the usage patterns of typical visitors to the website. This approach to pattern recognition is also called principal component analysis (PCA). Formally, the singular value decomposition of a \( m \times n \) real matrix \( M \) is a factorization of the form \( M = U S V^T \), where \( U \) is an \( m \times m \) real matrix, \( S \) is an \( m \times n \) rectangular diagonal matrix with non-negative, decreasing, numbers on the diagonal, and \( V \) is an \( n \times n \) real matrix. The diagonal entries \( s_i \) of \( S \) are known as the singular values of \( M \). The columns of \( U \) and the columns of \( V \) are called the left-singular vectors and right-singular vectors of \( M \), respectively.

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

PCA is often implemented by calculating the SVD of the data matrix. Let \( M \) be our (completed) data matrix, where the rows are sessions and columns are pages, and entry \((i, j) = k\) means the \( i \)th session visited the \( j \)th page, with a page visit duration \( k \). We could think about the visiting duration of page \( j \) as a real random variable, \( P_j \), whose empirical distribution is known by the values of column \( j \) of the data matrix \( M \). Note that the variables \( P_i \) and \( P_j \) are generally not independent, for example, a visitor who spent a long time on the page about undergraduate accommodation (a high value of \( P_i \)) might also be interested in pages about undergraduate courses (high values of \( P_j \)), while not very interested in graduate accommodation (low values of \( P_k \)). The (empirical) covariance matrix \( X = M^T M \) describes the relationships between different variables.

A basis of a vector space is a linearly independent set of vectors such that every element in the vector space is a linear combination of elements in the basis. For example, the set \{\((1, 0, 0), (0, 1, 0), (0, 0, 1)\)\} is a basis of the three-dimensional space we live in everyday life. We use the idea of vector space to model the modality of users. Each session (or a visit by some user), can be represented abstractly as a linear combination of basic users, which act as a basis of the vector space of all users, \( u_i = \sum_{k=1}^{m} c_k p_k \), where \( c_k \) are page visit durations to page \( k \), and \( p_k \) is a basic user, who will have the behaviour of spending one second visiting page \( k \) and nothing else. The set of basic users, forms a basis, because they are independent, i.e. we
cannot get the user behaviour represented by $p_i$ by superimposing the user behaviours of other basic users, as any basic user other than $p_i$ would have zero page visit duration to page $i$, and so does any linear combination of them. On the other hand, every possible user behaviour can be characterised by a linear combination of different basic users, for example, a person who spent 3 seconds on page 1, 2 seconds on page 2 and nothing else, could be characterised as $3p_1 + 2p_2$.

The choice of basic users is not unique, just as the choice of a basis is not unique. It would be easier to analyse the covariance matrix $X$ by diagonalizing it (if it were not full rank, this diagonalization will produce some zeros), this amounts to a change of basis, or in our context, a change of basic users. Note that the SVD of the data matrix $M$ readily produces a diagonalization of $X$, since $X = M^T M = V S^T U^T U S V T = V S^T S V T = V D V^T = V D V^{-1}$, as $V$ is orthogonal. And since $S$ is diagonal, so is $D = S^T S$. Recall that diagonal entries of $S$ are non-negative and decreasing, so diagonal entries of $D$ are non-negative and decreasing, and they are actually square of singular values $s_i$. This produces a diagonalization of $X$ because now $D = V^{-1} X V$ and more over, columns of $V$ represent the basis that we should use to diagonalize $X$. In another word, $V$ represent our new basic users that would diagonalize $X$. For example, if $V$ is a $3 \times 3$ real matrix, then our new basic users are $(p_{\text{new}})_i = \sum_{k=1}^{3} V_{ki} p_k$, where $p_k$ are the old basic users.

In this new basis, the diagonal matrix $D$ is also a covariance matrix, but for a set of different random variables. Recall that the old set of random variables are $P_i$, which represents the page visit duration to page $i$. Now, we have $D = V^{-1} X V = V^T M^T M V = (MV)^T M V$, so $D$ is the covariance matrix of transformed random variables, and the transformation is represented by $V$, in the same way as $V$ transforms the basic users. For example, in the three dimensional cases, our new random variables are $(P_{\text{new}})_i = \sum_{k=1}^{3} V_{ki} P_k$. The important thing is that since the covariance matrix is diagonal, the transformed random variables are independent, i.e. knowing the value of one gives no information of values of the others. Note that the sampling values of those random variables are actually the coefficients in the user characterisation. Recall we had $u_i = \sum_{k=1}^{m} c_k p_k$ to characterise the $i$th user’s behaviour, where $c_k$ are page visit durations, i.e. sampling values of the old random variables. After we transform everything accordingly, we still have $u_i = \sum_{k=1}^{m} (c_{\text{new}})_k (p_{\text{new}})_k$, which can be verified by direct substitution, and $(c_{\text{new}})_k$ are now sampling values of our transformed random variable $(P_{\text{new}})_k$.

In this way, we have produced the PrincipalComponents, which are first few of our transformed variables $(P_{\text{new}})_k$, of the data matrix $M$. They are uncorrelated (independent), orthogonal (as $V$ is orthogonal, hence the coordinate vector of them will be orthogonal), and they have the largest possible variances (since variance is the diagonal entries of the covariance
matrix $D$, which has decreasing, non-negative diagonal entries). Because they have the largest possible variances, they form the most important individual traits of different users, i.e. they explained the different behaviours of different users. Knowing their sampling values would be much more important than knowing the sample values of some other variables with much smaller variances. For example, suppose the value of variable $(P_{new})_{1}$ is actually an indicator of how much the user is like a students, then a high value of it would mean the user is more likely to be interested in pages about courses and academics. (It has high variance as there are lots of users who do not behave like a student at all, for example, administrative staff of the college, while we also have a quite large student population) In contrast, suppose the value of variable $(P_{new})_{100}$ represents how long a user spends on a really obscure page. Since almost everyone spends very little time on it, the variance is small, but knowing how much one spends on such page provides almost no information of how one would behave in other situations, as almost everyone did the same. In another word, without inspecting the value of variable $(P_{new})_{100}$, one would know to a high accuracy, that the value is close to zero.

In essence, the first few principal components are really what is needed to reconstruct the whole data set (this technique is used in image compression). But here, we have another assumption, that our data matrix is low rank or approximately low rank.

Let’s consider, for simplicity, the case that our data matrix is exactly low rank, say, rank 5. This means there can only be five non-zero diagonal entries in the diagonalization (or really, diagonalization with zero entries). This means the new covariance matrix $D$ is zero except for the first five columns, which means the remaining random variables are constants. Now, if we have done mean-centered before (i.e. we subtract the empirical mean for each random variable from the observation, so that each random variable have mean zero, and hence any linear combination of them would be zero mean), the remaining transformed random variables much be identically zero! (Having zero mean and zero variance) Recall that each user is characterised as $u_i = \sum_{k=1}^{m} (c_{new})_k (p_{new})_k$, where $(c_{new})_k$ is observation values for the transformed variables. Since every variable except the first five are identically zero, each user is a linear combination of five eigen-users only! (We now call the transformed basic users eigen-users) This is dramatic result, as there might be hundreds of thousands of users, but we have managed to explain their behaviour by using only five eigen-users. If we optimise the website linkage structure for these five eigen-users, then we would have optimised the linkage structure for all users!

The reality is, however, never perfect. Even if the true data matrix is low rank, it is certainly corrupted by noises, and the numerical rounding off error in modern computational software package essentially entails almost every matrix is full rank! Nevertheless, we have good reasons to believe that the
data matrix is \textit{ApproximatelyLowRank}, that is, it is not too much different from a truly low rank matrix, and the difference could be well caused by noise, or the difference could be real. In the situation where the differences are real, we would be discarding informations by writing each user as a linear combination of only a few \textit{eigen}–\textit{users}, but if the matrix is approximately low rank, we are preserving most informations, and we could approximately optimise the linkage structure for all users.

The condition under which we could preserve most informations by taking only a few \textit{eigen}–\textit{users} is that we must have a few large singular values and all the other singular values are small, in the SVD of the data matrix. This is related to the unstructured problem of low rank matrix approximation.

Let $D$ be a matrix. The unstructured problem with similarity measured by the Frobenius norm is

$$\text{minimize over } \hat{D} \quad \| D - \hat{D} \|_F \quad \text{subject to } \text{rank} (\hat{D}) \leq r$$

This problem has solution in terms of the singular value decomposition of the data matrix. The result is referred to as the Matrix Approximation Lemma or Eckart–Young–Mirsky Theorem.[6] Let $D = USV^T \in \mathbb{R}^{m \times n}$, $m \leq n$ be the singular value decomposition of $D$ and partition $U$, $S = \text{diag}(s_1, \ldots, s_m)$, and $V$ as follows:

$$U = [U_1 \ U_2], \quad S = \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix}, \quad \text{and} \quad V = [V_1 \ V_2],$$

where $S_1$ is $r \times r$, $U_1$ is $m \times r$, and $V_1$ is $n \times r$. Then the rank-$r$ matrix, obtained from the truncated singular value decomposition $:\hat{D}^* = U_1 S_1 V_1^T$, is such that $:\| D - \hat{D}^* \|_F = \min_{\text{rank}(\hat{D}) \leq r} \| D - \hat{D} \|_F = \sqrt{s_{r+1}^2 + \cdots + s_m^2}.$ The minimizer $\hat{D}^*$ is unique if and only if $s_{r+1} \neq s_r$. It can also be shown that the $D^*$ is also the best approximation in terms of spectral norm.

Note that the above approximation is equivalent to set $\hat{V} = \{ v_1, v_2, \ldots, v_r, 0, 0, \ldots, 0 \}$ in the singular value decomposition, where $v_i$ are columns of $V$, then replace the data matrix $X$ by $X = USV^T$. And the effect of this is equivalent to writing each user simply as a linear combination of the first $r$ \textit{eigen}–\textit{users}. We will see that in our case, $r = 5$.

Finally, we mention that for our analysis in reality, normalization and mean-centering should be done before doing PCA. One reason is to make the analysis simpler, as we have used the fact that each variables has zero mean in the above analysis. More importantly, these two techniques can actually reduce ranks of the data matrix, hence make the low rank assumption more valid. Normalization is the procedure of diving every entry $(i,j)$ of row $i$ by the $\sum_{j=1}^m X_{ij}$, so absolute visiting duration is replaced by page visit duration relative to the total visiting time. This allows us to put users
with the similar visiting pattern but different total visiting duration on the same footing. Mean centering is even more important, and without it, we can often fail to reduce dimensions even when it is possible. Consider the example where we have a set of data with two features, people’s heights and weights, and we could plot the data points on a 2-dimensional Cartesian coordinate system. PCA (with one component only) amounts to find a line going through the origin, such that when projecting each data points to the line, the total errors are small. We know that people’s heights and weights are often positively correlated, so such one-dimensional representation should be possible. However, it might be difficult to draw such a line through the origin, unless the data has been mean-centered first. For example, if we have two data points, weights = \{64,70\} and heights = \{170, 182\}, then the desired line would be \(y = 2x + 42\), which does not go through the origin, and if we are forced to choose a line going through origin, we will get really bad representation. So without mean centering, we have to use two dimensions, i.e. rank 2 approximation. After mean centering, the data would be \{-3,3\} and \{-6,6\}, and clearly, we could use \(y = 2x\) to get an excellent fit.

2.4 Hyper-link Induced Topic Search

We give a detailed description of HITS algorithm here. Interested readers are referred to the Matlab implementation of HITS for this particular analysis in the appendix (Code 6.9). First of all, we will formulate a mathematical description of the linkage structure of a website. Given any linkage structure (A table describing the source and the destination of all links), we could encode all information in a matrix \(L\), by the following procedure:

- Step 1. Let \(S\) be the number of pages in this websites, index the pages accordingly so that each page is represented by a numerical value \(p\), where \(p = 1, 2, \ldots, S - 1, S\). Let \(L\) be a \(S \times S\) real matrices, with zeros on every entry.

- Step 2. Let \(link\) be a link in the linkage structure, with source page \(source\) and destination page \(destination\).

- Step 3. Find the page index \(i\) of \(source\) and page index \(j\) of \(destination\).

- Step 4. \(L_{ij} = L_{ij} + 1\), where \(L_{ij}\) is the \((i, j)\)th entry of \(L\)

- Step 5. Repeat Step 2 to 4 until we have gone through all links in the linkage structure.

It is clear that after finishing this procedure, the \((i, j)\)th entry of the matrix \(L\) would represent number of links from page \(i\) to page \(j\), and this is non-zero if and only if there is a link from page \(i\) to page \(j\).
Given the matrix $L$, and a set of page indices $RootSet$ (usually produced by some other means, in our case, produced by PCA) the HITS algorithm could be described by the following:

- **Step 1:** Identify the Focus Subgraph. Let $s$ be the size of the square matrix $L$ (number of rows or number of columns). Let $baseSet$ be equal to $RootSet$, and do the following loop
  
  For $i \in RootSet$,
  
  For $j = 1, 2, 3, \ldots, s$,
  
  If $L_{ij} > 0$, add $j$ to $BaseSet$. The above produces $BaseSet$, which is an expansion of $RootSet$, including any page originally in $RootSet$ and any page that links to those original members.

- **Step 2:** Let $a.score$ be the vector of authority scores, $h.score$ be the vector of hub scores, so that $a.score(i)$ is the authority score for page $i$, and similarly for $h.score$. Assign each page in $BaseSet$ a hub score and authority score of 1.

- **Step 3:** Authority Update. Update each page’s authority score to be equal to the sum of the hub scores of each page that points to it. That is,
  
  For each page $i \in BaseSet$
  
  For each page $j \in BaseSet$
  
  If $L_{ji} > 0$, then $a.score(i) = a.score(i) + h.score(j)$

- **Step 4:** Hub Update. Update each page’s hub score to be equal to the sum of the authority scores of each page that it points to. That is,
  
  For each page $i \in BaseSet$
  
  For each page $j \in BaseSet$
  
  If $L_{ij} > 0$, then $h.score(i) = h.score(i) + a.score(j)$

- **Step 5.** Normalize $a.score$ and $h.score$. Let $sum.a = \sum_{i \in BaseSet} a.score(i)$.
  
  $sum.h = \sum_{i \in BaseSet} h.score(i)$.
  
  For each $i \in BaseSet$
  
  $a.score(i) = a.score(i)/sum.a$, and $h.score(i) = h.score(i)/sum.h$.

- **Step 6**, repeat Step 3 to 5 until the vectors $h.score$ and $p.score$ converge.
Chapter 3

Detailed Approach

3.1 Gathering the data

In order to do any analysis, data must be gathered in first place. To this end, we collaborated with a consultant, Simon Wallace, who kindly helped us extract the visiting records data with Google Analytics. For each “click” on the Pembroke College website, we extract: PageURL, UserId, Time when the click happened (TimeStamp) and the duration of the page visit (PageVisitDuration). There are about 40k records that we are going to work with. A sample of the original data is included below.

<table>
<thead>
<tr>
<th>PageURL</th>
<th>UserId</th>
<th>TimeStamp</th>
<th>PageVisitDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.104664e+09</td>
<td>unknown</td>
<td>2016-06-21T08:05:30.19-07:00</td>
<td>17</td>
</tr>
<tr>
<td>1.238444e+09</td>
<td>internal</td>
<td>2016-06-21T08:47:16.726-04:00</td>
<td>20</td>
</tr>
<tr>
<td>1.039427e+09</td>
<td>unknown</td>
<td>2016-06-21T08:47:52.129-04:00</td>
<td>12</td>
</tr>
<tr>
<td>1.238444e+09</td>
<td>internal</td>
<td>2016-06-21T08:49:16.719-04:00</td>
<td>288</td>
</tr>
<tr>
<td>2.089758e+09</td>
<td>unknown</td>
<td>2016-06-21T09:55:45.566-04:00</td>
<td>4</td>
</tr>
<tr>
<td>8.015173e+07</td>
<td>unknown</td>
<td>2016-06-21T10:08:56.677-05:00</td>
<td>59</td>
</tr>
<tr>
<td>2.846882e+07</td>
<td>undergraduate</td>
<td>2016-06-21T11:45:44.305+01:00</td>
<td>7</td>
</tr>
<tr>
<td>2.846882e+07</td>
<td>undergraduate</td>
<td>2016-06-21T11:46:07.73+01:00</td>
<td>47</td>
</tr>
<tr>
<td>8.693352e+08</td>
<td>undergraduate</td>
<td>2016-06-21T11:48:36.607+01:00</td>
<td>12</td>
</tr>
<tr>
<td>4.904227e+08</td>
<td>staff</td>
<td>2016-06-21T12:03:36.41+01:00</td>
<td>184</td>
</tr>
</tbody>
</table>

3.2 Processing the data

Google Analytics gathers and returns raw data, that has to be processed before the type of analytics we have in mind can be carried out. For the purpose of this project, we used Matlab as our main programming language.

First, we needed to convert the URLs, which was in the format of “string” to numerical values, we used a Matlab script to create a dictionary for the URLs. (Code 6.1 in appendix)
Secondly, the *TimeStamps* come in the string format, which could not be easily used in computation, so we parsed the string using Code 6.2 (in appendix).

After these two steps, we have converted the raw data into numerical forms. The following is a sample of the data in numerical form. Where we see the URLs has been replaced by numerical indices in the first column, and timestamps converted into separate numerical columns.

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>566</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>2.105</td>
<td>20</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>699</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>30.666</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>778</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>40.876</td>
<td>28</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>859</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>24.934</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1126</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>21.778</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1126</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>26.821</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1126</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>46</td>
<td>37.942</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1126</td>
<td>583997.1458</td>
<td>2016</td>
<td>6</td>
<td>30</td>
<td>2</td>
<td>47</td>
<td>9.254</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>763090.1469</td>
<td>2016</td>
<td>7</td>
<td>14</td>
<td>16</td>
<td>43</td>
<td>11.115</td>
<td>10</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1568</td>
<td>763090.1469</td>
<td>2016</td>
<td>7</td>
<td>14</td>
<td>16</td>
<td>40</td>
<td>6.96</td>
<td>167</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

### 3.3 Imperfection of the data

Before we proceed, we had to investigate to what extent the data can be trusted to be accurate. A few simple experiments showed that the Google Analytics API did not record a page visit if someone opened a page but forgot to close it. For example, I made a visit to the Pembroke College website, followed some specific links, then went to make a coffee without closing the browser. After I came back, I closed the page, then found my visiting records in the API recorded some 900 seconds page visit duration (apparently it had included the time of coffee break). This is due to a technical constraint by which it is impossible to tell whether a user is reading the current page or has taken a break to get coffee. This is a serious misleading, as one would greatly overestimate the duration of the visit to the page and thus its authority score for this user. Luckily, statistically speaking, we could expect most users to close their webpage after their visits. We thus take the view that the above described situations manifest as "outliers", and we will be able to mitigate their influences using outlier removal techniques mentioned in chapter two, section one. We give a histogram plot of all the *PageVisitDuration*, and we could easily see that the vast majority of visits have a *PageVisitDuration* of below several hundreds. In fact, 203 is the 90th percentile and we would apply a 90% winsorization.
3.4 Constructing a useful data structure for analysis

Our mathematical techniques, whether SVD or Low Rank Matrix Completion, or even HITS, requires that the data be put in a matrix format. It
is useful to summarise all the visiting records in a matrix, where the rows are sessions and columns are pages (see chapter two, section two). The Matlab script (Code 6.3) takes the preprocessed raw data as the input, and construct the required page-session matrix $M$.

And we give a sample of the data matrix we have now:

$$
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
58 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

Note that the matrix is extremely sparse, as there are a large number of pages in total (1650), but it is common for any single session to visit only a few pages. Another thing we may note is that a large percentage of visitors have spend a considerable amount of time on the entry page, which is given by the first column. All the other entries in the examples are zero, because for the first ten sessions, no one visited page 1 to 10.

### 3.5 The problem of scarcity and low rank matrix completion

As mentioned in the last section, the data matrix is extremely sparse. Running SVD directly on this matrix would give a misleading result. As an entry of zero does not signify that the user is not interested in this page. It may simply happen because that page is buried deep in the website, but is actually very useful. One of our tasks is to identify such pages, and make them more accessible to users.

As in chapter two, section two (Low Rank Matrix Completion), we think each page as a random variable, whose values are correlated. E.g. a person who spends a large time on the page about graduate accommodation will spend less time on undergraduate accommodation (he is probably a graduate). The scarcity of the matrix means we only have relatively few samples. We want to use these few samples, and the similarity between users’ behaviours, to predict how interested they would be on those unvisited pages. This problem is very similar to the Netflix problem, where there are a huge data base of movies, but each user has only rated a few movies, and we want to know their potential ratings for unwatched movies so that we could make relevant recommendations.
The technique that will be used here is the low rank matrix completion (see chapter two). We can assume the original matrix is approximately low rank, because we can assume the Pembroke College website, or any internal website in general, have only a limited number of intended usage pattern. For example, the Pembroke college website might be devised to provide services mainly for students, alumni, staff, which consists of the ‘basis’ of all the visiting records. Everyone is a combination of them, and something else (individual trait), but most visitors have behaviours that could be largely explained by viewing them as a linear combination of the three only, while discarding their subtle individual traits. If this is the case, than students, alumni, staff are our three ranks, and the original matrix should be approximately rank three. In reality, we might have a rank a bit larger than three, but we will expect it to be small, because people come to a specific website only for some specific purposes, especially for a college website.

We implemented FPCA (see chapter two) to complete the matrix, assuming the completed one is approximately low rank. The mathematical institute of Oxford has kindly provided us with the computing facilities as the computation is rather expensive for a large data matrix. The FPCA algorithm gives us the completed data matrix, $\hat{M}$, and after normalization (using relative times) and mean centering (subtracting each column by its means), we arrived at a processed data matrix, $\hat{\hat{M}}$. Those two matrices will already give us quite a lot of informations about user behaviours, but we will do something further. The first 20 singular values of $\hat{\hat{M}}$ are $133.6694, 51.9148, 45.9313, 10.7616, 8.9796, 7.6903, 5.2978, 4.9687, 4.3707, 3.7162, 3.4864, 3.2237, 2.7994, 2.4566, 2.0525, 1.6972, 1.6049, 1.4357$. We should think of the square of them as the relative importance of each singular vectors (mathematically, they are the variances of transformed random variables corresponding to eigen $-$ users (See the section about SVD in chapter two). From the data, we see that the first four singular vectors will do a good job at representing the overall data, i.e. they are the most common type of usage of the website. Quantitatively, they can represent more than 95% of user behaviours. And we will analyse what do they mean latter.

3.6 Implementing HITS

Out ultimate aim is to implement HITS (see chapter two, section four) to optimise the linkage structure of the website. To implement HITS, we need two inputs, in the desired mathematical formats. One of them is a set of pages which we use as our RootSet. While it is useful already to analysis eigen $-$ users from SVD, more importantly, they give us potential RootSet that we could work with. The reason being that if there are only
four eigen users that really matter, and we know individual preferences for those eigen users, we could optimise the website linkage structure for those eigen users by using the sites that they like as RootSet. We then run HITS four times, and will get four different rankings for pages as good hubs. If we then take the most highly ranked pages from all the four sets, and put them in easily accessible places, all the four eigen users would find it easy to get informations they want. And since any real user is approximately a linear combination of them, any real user would find it easy to get informations they want!

As explained above, we have our RootSet, and we still need the linkage structure of the website coming in matrix format described in chapter two. To extract the linkage structure, we use a website crawler called StreamingFrog. It gathers all links within the websites, and we give a sample of data it gathers below.
<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/academic-administration">http://www.pmb.ox.ac.uk/students/academic-administration</a></td>
<td><a href="http://www.pmb.ox.ac.uk/conferences/accommodation">http://www.pmb.ox.ac.uk/conferences/accommodation</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/international-symposium-growth-crisis-democracy-europe-and-japan">http://www.pmb.ox.ac.uk/content/international-symposium-growth-crisis-democracy-europe-and-japan</a></td>
<td><a href="http://www.pmb.ox.ac.uk/students/admissions/courses/theology-oriental-studies">http://www.pmb.ox.ac.uk/students/admissions/courses/theology-oriental-studies</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/robert-grosseteste-greatest-mind-youve-never-heard">http://www.pmb.ox.ac.uk/content/robert-grosseteste-greatest-mind-youve-never-heard</a></td>
<td><a href="http://www.pmb.ox.ac.uk/contact-us/jcr">http://www.pmb.ox.ac.uk/contact-us/jcr</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/hutcheson-andrew-guy">http://www.pmb.ox.ac.uk/content/hutcheson-andrew-guy</a></td>
<td><a href="http://www.pmb.ox.ac.uk/students/graduate-students">http://www.pmb.ox.ac.uk/students/graduate-students</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/particle-physics-strange-stories-neutrino-and-susy">http://www.pmb.ox.ac.uk/content/particle-physics-strange-stories-neutrino-and-susy</a></td>
<td><a href="http://www.pmb.ox.ac.uk/content/fullbright-lecture-2016-prof-michael-ignatieff">http://www.pmb.ox.ac.uk/content/fullbright-lecture-2016-prof-michael-ignatieff</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/node/3031">http://www.pmb.ox.ac.uk/node/3031</a></td>
<td><a href="http://www.pmb.ox.ac.uk/rowing/history">http://www.pmb.ox.ac.uk/rowing/history</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/discover-pembroke/news-events/college-events/past-events?page=4">http://www.pmb.ox.ac.uk/discover-pembroke/news-events/college-events/past-events?page=4</a></td>
<td><a href="http://www.pmb.ox.ac.uk/students/admissions/courses/law">http://www.pmb.ox.ac.uk/students/admissions/courses/law</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/news/famelab-runner-spot-becky-smethurst">http://www.pmb.ox.ac.uk/news/famelab-runner-spot-becky-smethurst</a></td>
<td><a href="http://www.pmb.ox.ac.uk/content/cian-wade-medicine-oxford">http://www.pmb.ox.ac.uk/content/cian-wade-medicine-oxford</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/fellows-staff/profiles/mr-ufuk-ozturk">http://www.pmb.ox.ac.uk/fellows-staff/profiles/mr-ufuk-ozturk</a></td>
<td><a href="http://www.pmb.ox.ac.uk/">http://www.pmb.ox.ac.uk/</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/fellows-staff/profiles/dr-kirsty-mchugh">http://www.pmb.ox.ac.uk/fellows-staff/profiles/dr-kirsty-mchugh</a></td>
<td><a href="http://www.pmb.ox.ac.uk/discover-pembroke/mcgowin-library/usefullinks-and-resources">http://www.pmb.ox.ac.uk/discover-pembroke/mcgowin-library/usefullinks-and-resources</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/finance-graduate-extra-curricular">http://www.pmb.ox.ac.uk/finance-graduate-extra-curricular</a></td>
<td><a href="http://www.pmb.ox.ac.uk/conferences/meetings">http://www.pmb.ox.ac.uk/conferences/meetings</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/sterling-robert-william">http://www.pmb.ox.ac.uk/content/sterling-robert-william</a></td>
<td><a href="http://www.pmb.ox.ac.uk/contact-us/communications">http://www.pmb.ox.ac.uk/contact-us/communications</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/node/2867">http://www.pmb.ox.ac.uk/node/2867</a></td>
<td><a href="http://www.pmb.ox.ac.uk/content/ep-third-year-advanced-options">http://www.pmb.ox.ac.uk/content/ep-third-year-advanced-options</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/about/history/alumni-and-pembrokians">http://www.pmb.ox.ac.uk/about/history/alumni-and-pembrokians</a></td>
<td><a href="http://www.pmb.ox.ac.uk/conferences/meetings">http://www.pmb.ox.ac.uk/conferences/meetings</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/content/north-west-science-centre-chemical-engineering-visit">http://www.pmb.ox.ac.uk/content/north-west-science-centre-chemical-engineering-visit</a></td>
<td><a href="http://www.pmb.ox.ac.uk/">http://www.pmb.ox.ac.uk/</a></td>
</tr>
</tbody>
</table>
These data are all information we want, but to do any real problems with mathematics, the first step is always to process the data and turn it into the form we want, in this case, a matrix representing the link structure as mentioned before. The implementation of this is relatively straightforward, we simply go through all the rows in the table, look up what is the page index for every URL we encounter, then add one to the relevant entries in the link matrix (signifies we have found one more link). To speed up the execution, as there are around a million links to look up, we created a dictionary in the Map format, where look up index from URL is a very quick operation, regardless of how many pages are there. We have also attached the code for creating the dictionary at the appendix (Code 6.4 and 6.5).

Theoretically, we have got everything we need and HITS should immediately give us the desired result. However, we find two problems.

One problem is that the linkage information gathered by ScreamingFrog doesn’t distinguish links with different visibility. Some links maybe hard to find, while others are more accessible. What’s more, there is a drop-down menu, including nearly all pages in the website, being put in the top right corner of nearly every page. And due to the coding convention of html, the source code of the drop-down menu includes the URLs of all pages it contains. In other words, it appears that each page links to nearly every other page, while in reality, many URLs in the source code are not directly accessible. Such features introduce a very large noise in the link structure analysis, and it would mislead HITS algorithm by making every page has the same links (i.e. links to nearly all the pages).

To deal with this severe problem, we devised a novel technique. Since the drop-down menu appears in every page we could view it as a basic feature that are not to be taken into account when comparing pages hub scores. Then, if every page (or most pages) have a certain number of links to another page (say, 3 links), and the remaining pages have a larger number of links to that page (say, 4 or 5 links), then it is almost certain that those 3 links are produced by the menu, or other common features, that should be ignored. In this way, we have artificially removed the redundancies while preserving those links that distinguish one page from the others.

We give an example. For the page http://www.pmb.ox.ac.uk/about-pembroke/freedom-information, there are 824 out of 2500 pages have 3 links to it, 631 pages have 4 links to it. One of the pages that is ‘claimed’ to have 3 links to it is ’http://www.pmb.ox.ac.uk/content/ella-st-george-carey-and-mike-joseph-history-undergraduate-course’. If we open these pages in the browser, we will actually find no link from the latter to the former! And it was reported they have three links! This is mainly due to the drop-down menu bar described before, and it would only be sensible to remove those 3 links from the count.

We give an implementation of the above idea in the Code 6.6.

Even with the above problem solved, we still get unsatisfactory results.
The algorithm gives very similar HubScores to a large group of pages. For example, we give a histogram plot of the distribution of HubScores for one of the eigen - users. From the histogram, we see that although the algorithm does some work of selecting the most important hubs, it fails to distinguish between a large number of pages, i.e. there are several hundred pages that have HubScores similar to maximum values. A further investigation shows that those pages with the highest HubScores are all pages with a ‘About Pembroke’ menu bar in the left, for example, the page ‘http://www.pmb.ox.ac.uk/content/art-gallery-exhibition-opening-salt-room-wanderings-when-night-falls’.

Figure 3.2

This failure of HITS is due to the fact that those pages are Densely Linked,
i.e. in this subgroup of pages, nearly every entry of the matrix representing their link structure is non-zero. (Recall the definition of such matrix in chapter 2). This kind of failure is detailed in the paper by S Nomura, S Oyama, T Hayamizu [8].

3.7 Effective Linkage structure and HITS

The *Effective Linkage Structure* include links that are followed least once in our records of website traffic. By using *Effective Linkage Structure*, we solve the problem of distinguishing links with different visibility, and moreover, we eliminate the misleading effects of *Densely Linked* page groups. We use Code 6.7 to extract the effective link structure from our original web-traffic data (in the numeric form, but haven’t been constructed into page-session form yet).

We run the HITS algorithm on this Linkage structure, and we got satisfactory results. This ends our implementation of HITS, and we will discuss results in the next chapter.
Chapter 4

Results and Analysis

4.1 Eigen-Users and Analysis

After the matrix completion and SVD, we have already got quite a lot of insights.

The first 20 singular values are 133.6694, 51.9148, 45.9313, 23.1636, 10.7616, 8.9796, 7.6903, 5.2978, 4.9687, 4.3707, 3.7162, 3.4864, 3.2237, 2.7994, 2.4566, 2.0525, 1.6972, 1.6049, 1.4357. We should think of the square of them as the relative weight of each singular vectors. From the data, we see that the first four singular vectors will do a good job at representing the overall data, i.e. they are the most common type of usage of the website. And we will analyse what do they mean.

The first singular vector is particularly dominating with its large singular values, and represent those behaviours that everyone has. We can think about it as a virtual user, whose behaviour could be found in almost all users. The webpages that this virtual user likes the most are (in the order of decreasing significance):

- '/vacancies/communications-designer-15-hours-week-0'
- '/accommodation'
- '/contact-us/buildings-maintenance'
- '/alumni-benefactors/alumni-dining-college'
- '/welfare'
- '/node/1726', Note, this is the rainbow picture that appears on the homepage.
- '/alumni/meet-team'
- '/fellows-staff/profiles/dr-peter-ditmanson-0'
One surprising thing to notice that many of those pages are by no means easy to find, starting from the homepage. For example, a lot of the users seem to be particularly interested in vacancies and accommodation, but instead, they have to spend a long time finding those relevant pages. As it is by no means clear how would you find information about accommodation and vacancies starting from the homepage.

The users’ need for information about accommodation is crystal clear, when we look at the pages that the sampled users have searched (about 10000 in total). There are:

- '/search/node/acmodation'
- '/search/node/accomadation'
- '/search/node/accomdaiton'
- '/search/node/accomidation'
- '/search/node/accommodation'
- '/search/node/Accommodation'
- '/search/node/accommodation costs'
- '/search/node/accommodation department'
- '/search/node/accomodations fees'
- '/search/node/accomodation'
- '/search/node/Acomodation'
- '/search/node/acmodations fees'
- '/search/node/housing'
- '/search/node/housing fee'

We can see there are a huge variety of spellings for the search term 'accommodation’. The other search term in general only have one spelling. And if we assume that there is a fixed probability that a user will misspell a term, or use another expression for a term, when he is searching, then since we have such a large variety of search terms related to accommodation, there must be an even large number of users who have searched something related to accommodation. Note that this is consistent with what we have described before, that a large number of users are interested in accommodation, but they cannot find the relevant page on the homepage, so they would have to perform a search, hence the large number of searches.

Similar phenomenon can be observed related to vacancies and jobs. There are:
This supports the view that a large number of users are interested in jobs offered by the college.

From the singular vectors, we also find that:

- The newly added alumni dining activities is very popular, and this also suggests that there are a large number of alumni users.

- Instead of complete the maintenance form in the intranet, many people still prefer to find the contact information of maintenance people and directly contact them.

- The rainbow photo is really beautiful, and people do love it.

- A surprising fact is that people are actually very interested in welfare

- Note that this virtual user is mainly a combination of alumni, accommodation seeker and job seekers, represented by its interest in accommodation, alumni dining and vacancies. And it turns out that our current students use the webpage rather infrequently.

Now we turn to the second most important virtual user, their main interests are:

- '/news-events’
- '/vacancies/administrative-assistant-part-time-fixed-term-0’
- '/alumni-benefactors/alumni-dining-college’
- '/intranet’
We see this virtual user is predominantly interested in looking for news. And hence we conclude that a lot of people come to the Pembroke college website looking for the latest news/events related to the college. Note that vacancy page still ranks high (although it is another vacancy), which confirms that there are indeed lots of job seekers. There are also a significant interest for courses offered by Pembroke, which represents the perspective students.

For the third and fourth singular users, vacancies and accommodations appear again, and also conferences. So we conclude that job seekers, news seekers, accommodation seekers, perspective students and alumni are the most frequent users of the Pembroke college website, in direct contract with our usual perception that there must be a large number of current students and academics using the website.

4.2 Results of HITS and Analysis

The top hubs for the first five (most important) $Eigen - Users$ can be found in the below table. We notice that there are 28 pages given. The pages in the table can be broadly divided in four categories: general college information, prospective students information, contacts and accommodation, conferences and alumni. These groups are largely in line with our expectation, and similar to results provided by PCA.

Note some of the hubs are easily accessible, for example, the homepage, or the page ‘http://www.pmb.ox.ac.uk/study-pembroke/courses’, which could
be reached by the conspicuous link ‘undegratuate courses’ from the homepage. Sadly, some of the hubs are rather hard to find. For example, the page http://www.pmb.ox.ac.uk/contact-us. Although there are two links to this page on the homepage, they are not easy to find (one is at the very top, and one is at the very bottom). Still, users need this contact information desperately. The number of clicks (totaled more than 350) that those ‘hard-to-find’ links received ranked top 10 among all pages that are linked by the homepage. Similarly, links to pages related to accommodations such as http://www.pmb.ox.ac.uk/accommodation', 'http://www.pmb.ox.ac.uk/accommodation-undergraduate’ are also most clicked links from the homepage, but it is not easy to find such links. In contrast, links to http://www.pmb.ox.ac.uk/students/visiting-students (the ‘Visiting Student’ big box in the homepage) have received no click over the whole sampling period. It would be hugely beneficial for users if we put links to accommodations, contacts, etc in the place of such a nearly useless box.

We want to emphasise once again that all these pages are excellent hubs, and we really need to take a close look at each one of them. We want to make sure that each of them is easily accessible from the homepage.
Table 4.1: Excellent Hubs

<table>
<thead>
<tr>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/">http://www.pmb.ox.ac.uk/</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/about/buildings">http://www.pmb.ox.ac.uk/about/buildings</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/about/history/alumni-and-pembrokians">http://www.pmb.ox.ac.uk/about/history/alumni-and-pembrokians</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/about/information-regulations">http://www.pmb.ox.ac.uk/about/information-regulations</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/academics">http://www.pmb.ox.ac.uk/academics</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/accommodation">http://www.pmb.ox.ac.uk/accommodation</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/alumni-benefactors/alumni-dining-college">http://www.pmb.ox.ac.uk/alumni-benefactors/alumni-dining-college</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/catering">http://www.pmb.ox.ac.uk/catering</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/conferences-and-events">http://www.pmb.ox.ac.uk/conferences-and-events</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/conferences/accommodation">http://www.pmb.ox.ac.uk/conferences/accommodation</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/conferences/dining">http://www.pmb.ox.ac.uk/conferences/dining</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/contact-us">http://www.pmb.ox.ac.uk/contact-us</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/contact-us/getting-pembroke">http://www.pmb.ox.ac.uk/contact-us/getting-pembroke</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/discover-pembroke/news-events/college-events">http://www.pmb.ox.ac.uk/discover-pembroke/news-events/college-events</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/discover-pembroke/vacancies">http://www.pmb.ox.ac.uk/discover-pembroke/vacancies</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/fellows-staff/profiles/dame-lynne-brindley-dbe">http://www.pmb.ox.ac.uk/fellows-staff/profiles/dame-lynne-brindley-dbe</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/fellows-staff/profiles/dr-elisabeth-kendall">http://www.pmb.ox.ac.uk/fellows-staff/profiles/dr-elisabeth-kendall</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/intranet">http://www.pmb.ox.ac.uk/intranet</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/news-events">http://www.pmb.ox.ac.uk/news-events</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/node/1726">http://www.pmb.ox.ac.uk/node/1726</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/node/2805">http://www.pmb.ox.ac.uk/node/2805</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/admissions/open-days">http://www.pmb.ox.ac.uk/students/admissions/open-days</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/clubs-and-societies">http://www.pmb.ox.ac.uk/students/clubs-and-societies</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/graduate-students">http://www.pmb.ox.ac.uk/students/graduate-students</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/study-guide/tutorials-lectures">http://www.pmb.ox.ac.uk/students/study-guide/tutorials-lectures</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/students/undergraduate-students">http://www.pmb.ox.ac.uk/students/undergraduate-students</a></td>
</tr>
<tr>
<td><a href="http://www.pmb.ox.ac.uk/study-pembroke/courses">http://www.pmb.ox.ac.uk/study-pembroke/courses</a></td>
</tr>
</tbody>
</table>
Chapter 5

Conclusion

In this report, four major mathematical methods are used, which are Winsorization, Low Rank Matrix Completion, Singular Value Decomposition and Hyperlink Induced Topic Search algorithm. The implementation of those mathematical models are not always straight forward. Data needs to be gathered at the first place, and can only be used after preprocessing. More importantly, there are circumstances that HITS can fail, and different method of constructing the linkage structure has to be used.

Analysing the web-traffic using PCA gives surprising results. Although we expect many people come to a college website mainly looking for academic information, real time users are more interested in jobs, accommodation and finding contacts(although there is indeed considerable interest in academics as well).

Using the HITS algorithm on the effective link structure, a short list of top hubs for all user groups of the website are found. Some of these hubs are easy to find, given the existing linkage structure, while others take more time to find, and as a result, it takes more time to find information that they point to as hubs. We recommend inserting a clear link to all of the hubs in the list.
Chapter 6

Appendix

Code 6.1: Dictionary creation

```matlab
%%This script takes a vector of strings of length x as
input (Not Assume sorted!), and creates a
dictionary.
%%I.e. it returns a x*2 matrix as labelled vector, and
another m*1 vector
%%as the dictionary, where m is the number of entries.

function [lab, dic] = cdic1(v)
x = length(v);
lab = cell(x,1);
dic = cell(x,2);

dic{1,1} = v(1,1); dic{1,2} = 1; lab{1,1} = 1; % Initialization
d = 1; % d counts the number of non zero entries in
the dictionary
for k = 1:x-1
    f=0;
    b = v(k+1,1);
    for l = 1:k
        a = v(l,1);
        if strcmp(a,b)==1
            f = 1;
        end
    end
    d = d+1;
    dic{d,1} = b;
```
Code 6.2: Convert strings into numerical format

```matlab
% The purpose of this script is to take the vector of timestamp strings as
% the input, produce a x*8 double matrices. the first column contains year,
% the second contains month, third, date, fourth, hour, fifth, minute,
% sixth, second, last, time zone, indicated by a single number between −12 to +12.
% Assume the input is 'time'
x = length(time);
m = cell(x, 8);
for k = 1:x
    m(k,:) = strsplit(time(k,:),{'-','T',';','+'});
end
```

Code 6.3: construct page-session matrix

```matlab
m = zeros(35838,1650); % Last column going to be used to store user id
m = sparse(m);
y = size(m,1); x = size(m,2)−1;
countm = 0; % countm, counts where is the next row in m to be filled.
```
% create an empty matrix, we know there are 1649 pages in our record.
% Create submatrix (per user basis) (subm1) in which we extract sessions.
count = 1; %count, counts where we are in the matrix
doublelab
while count˜=(y+1)
    a = doublelab(count,2);
    k = count;
    while doublelab(k,2) == a && k˜=y
        k = k+1;
    end
    if k == y
        k = k+1;
    end
    subm = doublelab(count:(k-1),:);
count = k;
uid = a;
l = size(subm,1);
end
% To construct sessions, we sort the submatrix by date. And Create submatrix (per date basis) (subm1)
subm = sortrows(subm,[4:8]);
% Extract further submatrices, until we exhaust the original matrix
count1 = 1; %count1, counts where we are in subm
while count1 ˜=l+1
    a1 = subm(count1,5);
    k1 = count1;
    while k1˜=l && subm(k1,5)==a1
        k1 = k1+1;
    end
    subm1 = subm(count1:k1,:);
count1 = k1+1;
% Now we have a single day's visiting records
% construct sessions
% First, we convert hours and minuites to seconds and add them
together, to the seconds column
subm1(:,8) = subm1(:,8) + subm1(:,6)*3600 + subm1(:,7)*60;
% Note the 8th column is already sorted by procedure before
%Now we construct sessions

%To this purpose, we scan down the list, for each entry, we try to
%’paste’ together the timestamp using the duration given. If it is
%within 1 second difference, we count it as a single session (i.e.
%user path), otherwise we start a new session.
%The last row in the list will result a termination of the process,
%and be put as the termination webpage
%The first row always signal a new session

l1 = size(subm1,1);
count2 = 1;
countm = countm+1; %Always start a new row, when initializing
m(countm,1650) = uid; %The first column represents user id
pid = subm1(1,1); %Subtract starting page id
m(countm,pid) = subm1(1,9); %Put duration.

while count2~l1
    f = 0; % flag
    diff = subm1(count2+1,8) - subm1(count2,8) ; % time diff by timestamp
    tdiff = subm1(count2,9); % recorded time diff
    if abs(tdiff-diff)<=900 % the two above approximately agrees
        f = 1; % flag sets to 1
    end

    if f ==1 % session continues
        pid = subm1(count2+1,1); % extract pid
        dur = subm1(count2+1,9); % extract duration
        m(countm,pid) = dur+m(countm,pid); % put the data in
    end

    if f ==0 % session discontinues
        countm = countm+1; % Start a new
session
m(countm,1650) = uid; % put in user id
pid = subm1(count2+1,1); %extract pid
dur = subm1(count2+1,9); %extract
duration
m(countm,pid) = dur+m(countm,pid); %
put the data in (with multiple page
visits, we take arithmetic sum)
end
count2 = count2+1; %continues the loop;
end
end

Code 6.4: Create Map as Dictionary

function Newmap = append_map(Oldmap,list)
%% Given our old map, oldmap, and our new page list, how to we update the map?
%% Note if we don’t have old map, we could take it as an empty map
valueSet = values(Oldmap);
valueSet = cell2mat(valueSet);
m = max(valueSet);
Newmap = Oldmap;
for k = 1:size(list,1)
    a = list{k,1};
    flag = isKey(Newmap,char(a));
    if flag ==0
        Newmap(char(a)) = m+1;
        m = m+1;
    end
end
end

Code 6.5: Create Linkage structure

function Link_str = createlinkagestructure(data,map)
%Data would be a matrix, first column is source, second column is destination
%Map is the dictionary of pages
36
Create an empty matrix

\[
\text{Link}\_\text{str} = \text{zeros(size(map,1),size(map,1))}
\]

By the creation of the map, all the data should be in there so no need to check the key

if (i,j) entry has numerical value k, then there are k links from i to j.

```
for k = 1:size(data,1)
    source = data(k,1);
    destination = data(k,2);
    if isKey(map, source) && isKey(map, destination) == 0
        k
        % Automatically ignore external URLs
    else
        v1 = values(map, source);
        v2 = values(map, destination);
        Link\_str(v1{1},v2{1}) = Link\_str(v1{1},v2{1})+1;
    end
end
```

Code 6.6: Remove Fake Links

```
function [consolidated_link] = consolidate_link(links)

k = 2;
while k <= 2536
    v = links(:,k);
    a = unique(v);
    out = [a, histc(v(:), a)]
    for m = 1:size(out,1)
        numb = out(m,1)
        freq = out(m,2)
        if freq > 800 && numb > 0
            links(:,k) = max(links(:,k) - numb, 0);
            k = k - 1;
            break
        end
    end
    k = k + 1;
end

end
```
consolidated_link = links;
end

Code 6.7: Construct Effective Link Structure

link_str0810 = zeros(2536,2536);
for k = 1:1649
    page = dic(k,1);
    sourceindex = values(Dictionary_2609,page{1,1});
    s = sourceindex;
    for j = 1:1649
        page = dic(j,1);
        destinationindex = values(Dictionary_2609,page{1,1});
        d = destinationindex;
        link_str0810(s{1,1},d{1,1}) = link_struc(k,j) + link_str0810(s{1,1},d{1,1});
    end
end

Code 6.8: FPCA

tic;
X0 = psmat_us;
s = size(X0);
X = rand(size(X0)) * 0.5;
%X = X0;
Xori = X;
for mu = [1000,800,600,500,400,300,200,100,50]
    mu = mu*0.001;
    tau = 0.75;
    L = X0>0; %L is the logical index
    X0 is the data matrix
    X is initial guess
end

% Take the difference between X and X0
% we only need those indexed by L, to take L as the argument
% It returns a column vector
% we multiply the column vector by a percentage number
% then bring it back
% to its original form
% Then we add it back, to update X
% This is a form of gradient descent to minimise A(X)−b
while 1̸=0
Xbefore = X;
diff = X−X0;
modi = diff(1)∗tau;
oriorm = zeros(s);
oriorm(1) = modi;
X = X − oriorm;

%Then we do the SVD to control the rank
[U,S,V] = svds(X,20);
X = U*Sreduced(tau,mu,S)*V’;
mu;
end
end
toc

Code 6.9: HITS

function [info_score, point_score] =
    Uniform_propogation7(link_s, V, n, mu, tol, flag)
%% This version implements the detailed subgraph idea, 0910.

%% This version focus on a small subgraph instead of the whole graph.
%% The focused graph consists of the root pages and the pages that link to them.
link_s = double(link_s > 0);
clear info_score point_score

%flag = 1 means taking the positive, flag = 0 means taking the negative

%This function takes a Link structure matrix, link_s, V, singular vector matrix, n, the number of a
singular vector

% It uses uniform propagation to give each page two
% scores, one information score and one pointing score.
% This will be stored
%
% Note, link_s is a square matrix, v is a column vector
% . length of v equal
% to number of rows equal to number of columns of
% link_s
% 
% This should converge, but we don’t intend to prove it
% at the moment.

s = size(link_s,1);
[posi,nega] = dividing_singular_vec(V,n);
size_diff = s-size(posi,1);
posi_full = [posi;zeros(size_diff,1)];
nega_full = [nega;zeros(size_diff,1)];

% Extract relevant links, assign them an initial
% score of 1

if flag==1
    root_index= find(posi>0) ’;
    linked = zeros(s,1);
    for k=root_index
        links = link_s(:,k)>0;
        linked = linked+double(links);
    end
    old_point=double(linked>0);
    old_point = old_point./norm(old_point,1);
    point_root = find(linked>0);
end

if flag==0
    root_index= find(nega>0) ’;
    linked = zeros(s,1);
    for k=root_index
        links = link_s(:,k)>0;
        linked = linked+double(links);
    end
    old_point=double(linked>0);
    old_point = old_point./norm(old_point,1);
point_root = find(linked > 0);

if flag == 1
    old_info = posi_full ./ norm(posi_full, 1);
    info_root = find(old_info > 0);
    %weight = [posi; zeros(size_diff, 1)];
    %info_score = weight;
    %point_score = zeros(s, 1);
    %old_info = info_score ./ norm(info_score, 1);
    %old_point = point_score;
end

if flag == 0
    old_info = nega_full ./ norm(nega_full, 1);
    info_root = find(old_info > 0);
    %weight = [nega; zeros(size_diff, 1)];
    %info_score = weight;
    %point_score = zeros(s, 1);
    %old_info = info_score ./ norm(info_score, 1);
    %old_point = point_score;
end

% Construct base set

base_set = zeros(s, 1);
for k = info_root
    base_set(k, 1) = 1;
end
for k = point_root
    base_set(k, 1) = 1;
end
info_root = find(base_set > 0);
point_root = find(base_set>0);
old_info_copy = old_info;
old_point_copy = old_point;

%%% Main Loop
while 0==0
    %One overall iteration
    %% Update Info score, only for root_info
    for k = info_root'
        %if sum(isnan(point_score))>0 || sum(isnan(info_score))>0
        % k
        %   %error(num2str(k));
        %end
        %% Update infoscore (Only for pages that in the positive group)
        if flag==1 %We are calculating positives
            index = find(link_s(:,k)); %Those pages that link to page k
            scores = 0;
            for page_num = index'
                score_transfer = old_point(page_num)*
                               link_s(page_num,k);
                scores = scores+score_transfer;
            end
            update = mu*scores;
            %if isnan(update) ==0
            old_info(k) = old_info(k)+update;
            %end
        %end
    end
end

42
if flag==0 %In the negative range
    index = find(link_s(:,k)); %Those pages that link to page k
    scores = 0;
    for page_num = index
        score_transfer = old_point(page_num)*link_s(page_num,k);
        scores = scores+score_transfer;
    end
    update = mu*scores;
    %if isnan(update) ==0
    old_info(k) = old_info(k)+update;
    %end
    %Do normalization at the end
    %if norm(info_score)>0
    %info_score = info_score./norm(info_score,1);
    %end
end
old_info = old_info./norm(old_info,1); %% Update pointscore

for k = point_root'
    index = find(link_s(k,:)); %Those pages that page k links to
    scores = 0;
    for page_num = index
        score_transfer = old_info(page_num)*link_s(k,page_num);
        scores = scores+score_transfer;
    end
    update = mu*scores;
    %if isnan(update) ==0
    old_point(k) = old_point(k)+update;
    %end
    % if abs(norm(info_score,1)-1)>1e-15
    % k
% end
end
if norm(old_point) > 0
    old_point = old_point ./ norm(old_point, 1);
end

%% Test for convergence

diff1 = norm(old_info_copy - old_info, 1)
diff2 = norm(old_point_copy - old_point, 1)
if diff1 < tol && diff2 < tol
    break;
else
    old_info_copy = old_info;
    old_point_copy = old_point;
end

%% Miscellaneous

info_score = old_info;
point_score = old_point;
try
    save(strcat('2709,n=', num2str(n), ',mu=', num2str(mu), ',tol=', num2str(tol), ',flag=', num2str(flag), '.mat'), 'info_score', 'point_score');
catch
    a = 0;
end
end
Bibliography


