Inferring Relevance Feedback from Server Logs:  
A Case Study in Online Recruitment

Rachael Rafter, Barry Smyth, Keith Bradley

Smart Media Institute, Dept. Computer Science,
University College Dublin, Belfield, Dublin4, Ireland.
{rachael.rafter, barry.smyth, keith.bradley}@ucd.ie

1 Introduction

Information overload is now a well-known problem facing Internet users, and the task of finding the right piece of information is becoming increasingly difficult. Indeed these problems are set to rise as the Internet continues to grow at an exponential rate. Recently, automatic content personalisation techniques and recommender systems have emerged as a potential solution to this problem. Such techniques avail of advanced user-profiling and intelligent information filtering methods to automatically learn about the likes and dislikes of individual users, and prioritise the delivery of information for these users based on these learned preferences.

Ultimately, the success of content personalisation technologies depends critically on the availability of comprehensive user profiles that accurately capture the interests of end-users. However, the automatic compilation of accurate profiles represents a complex learning task. In this paper we address the issues involved in learning user profiles that are needed by Internet content personalisation systems. In particular, we focus on a key assumption that has been adopted by some such systems, namely that accurate user profiles can be generated directly from analysing certain behaviours such as the click-stream (the links that users click) and realtime data (how long a user spends reading a given page) of Web users. Although this idea has been proposed for many recommender systems already, it has rarely been rigorously tested.

The CASPER project, [8] focuses on the application of intelligent personalisation and information filtering techniques in online systems. In particular CASPER looks at employing these technologies to enhance JobFinder (www.jobfinder.ie) - an online recruitment service that uses traditional database-driven, query-based search techniques, which make it prone to information overload problems. The CASPER project investigates a number of techniques such as case-based reasoning [1] and automated collaborative filtering [7], in addition to the user profiling techniques that we focus on in this paper.

Our goal in this paper then, is to evaluate the validity of the aforementioned assumption: that accurate user profiles can be generated by analysing user behaviour in the CASPER system. CASPER constructs user profiles by passively monitoring the click-stream and realtime behaviour of regular users. Specifically, we will argue that building accurate user profiles from click-stream and realtime data is far from straightforward. In particular, we will describe some of the techniques and metrics that are used in CASPER to generate high-quality graded user profiles from raw server logs. Finally, we will describe a comparative evaluation of these different techniques on real user data.
2 Background

In general, recommender systems rely on user profiles in some shape or form, and one of the most common strategies is to profile users by recording their level of interest in specific information items, thus generating user profiles that consist of content item identifiers and their associated ratings. Current research can be usefully classified along two dimensions: the content filtering strategy used versus the type of user profiling employed. Content filtering strategies are generally either content-based or collaborative, while user profiling strategies are either active or passive.

The different styles of content filtering methods employed in recommender systems both adopt familiar social recommendation policies albeit from different perspectives. Content-based techniques (such as case-based reasoning) seek to recommend similar items to the items that a user has already liked in the past. This necessarily involves comparing candidate content items to the content items listed in the user profile, preferring those items that are similar to items that the user has rated positively and dissimilar to the items that the user has rated negatively. In contrast, content-free techniques (such as collaborative filtering) seek to select items for a given user that other users (with similar tastes) have liked. This typically involves identifying users that have similar preferences to the target user, in the sense that they have rated some of the same content items in the same way; that is, users whose ratings correlate positively with the target user.

Active and passive user profiling techniques differ in the manner in which they collect profile information from users. Active user profiling techniques essentially transfer the profiling problem onto the user in the sense that the user is required to provide explicit profiling information, such as content ratings. For example, PTV - www.ptv.ie [10] generates personalised TV guides for individual users based on their profiled preferences which are gathered directly by encouraging the user to explicitly rate programmes that appear in their personalised guides as positive (liked) or negative (disliked). Thus, PTV can only learn about its users by obliging them to explicitly engage in the profiling process.

In contrast, passive user profiling techniques remove the burden associated with explicitly rating items from the user, and rather try to infer this information implicitly. To this end, these techniques monitor certain browsing/searching behaviours of the user which are thought to be indicative of that user's interest. For example, the GroupLens Project ([3,9]), which performs automated collaborative filtering for Usenet news articles, adopts the amount of time that a user spends reading a news article as an indication of her interest in that item.

Our main focus in this paper is the user profiling component of the CASPER recommender system which has been designed as a personalised information assistant for users of the Jobfinder (www.jobfinder.ie) recruitment site. One of the core motivations in CASPER is to investigate the implicit profiling of users based on their click-stream and runtime data. In this sense CASPER is related to other recommender systems such as ([3,9, 11,10]).

3 Mining User Interests in CASPER

User interests in CASPER are gathered by mining the JobFinder server logs which record details of the user interactions within the website, see Fig. 1. Essentially, each line of the log file records a single job access (user clicks on a hyperlink to see a job description) by a user, and encodes details like the time of access, the job and
user ids. In addition to this, any action that the user performed with respect to that job is recorded - JobFinder allows users to email a job to themselves for later consideration, or apply for it directly online.

The most basic form of a profile is simply a history of jobs that the user has looked at over time. In general though, a list of jobs that a user has accessed does not constitute a reliable picture of the user's interests, for example, a job that has been clicked on may turn out to be uninteresting in the end, once the user has seen its description. Thus a basic profile for JobFinder can be misleading as it depicts those jobs that the user thinks she might be interested in, prior to viewing. Moreover, a given user is likely to have different levels of interest in the jobs that she genuinely likes. Therefore, a more detailed profile representation is needed that also records relevance information to discriminate between those jobs that the user looks at or considers, and those that she is truly interested in.

Graded profiles, which are used in many collaborative recommender systems (e.g. [10]), supplement the basic profile representation with relevance indicators. These indicators are essentially the set of grades that measure how relevant each item is to that user. However, these indicators can take many shapes and forms, and it is here that the difference between explicit and implicit profiling becomes interesting. A system like PTV [10] explicitly requires users to grade profile items (in PTV - TV programs) according to their taste, and therefore the grades given to the profile items are actual grades. In contrast, an implicit profiling system like CASPER’s recommender system observes the users browsing behaviour to produce a set of grades and hence, an immediate advantage is that the system does not interfere with the user’s browsing and the user does not have to perform any extra task. It has also been documented [9] that attaining extra ratings implicitly can at least supplement explicit ratings, especially when the profile space is sparse (also [3, 7]) and there is a general lack of ratings within it. The 'grades' in an implicitly generated profile may correspond to the amount of time the user spent reading a particular item, whether she purchased the item, or whether she bookmarked it etc. [5]. In CASPER, these grades correspond to three main types of information: the number of revisits made to a job description, the amount of time spent reading a job description, and whether the user applied for a job or mailed it back to herself.

Fig. 1. From server logs to graded profiles
3.1 Revisit Data

The number of times that a user clicks on a job is thought to be an indicator of her interest in that job, in the sense that users will often return for a second or third read of an interesting job description while they are unlikely to revisit uninteresting jobs after an initial read, (see also, [2]). For this reason CASPER logs the number of times that each user clicks on a job. This measure of raw revisits gives us an idea of how often the user has clicked on the job to read its description. However, the number of times a user clicks on a job may not correlate with the number of times that user revisited the job. In fact, due to slow bandwidth problems, etc. many of these clicks are so-called "irritation clicks" due to a user repeatedly clicking on a job in frustration, while waiting for the description to download, and therefore do not constitute accurate revisit data. In order to deal with this misleading revisit data, CASPER employs a thresholding technique that counts repeated clicks on the same job as irritation clicks. For example, in Fig. 1, according to the server log, the user "Rachael" has repeatedly clicked on the job richjobs*809 three times in quick succession, presumably in irritation due to a slow download time. Thus, these clicks are collapsed into one. In contrast, the user has clicked on csjobs*0 twice with a 16 minute gap between clicks during which time she looked at other jobs. This is interpreted as a genuine revisit and thus both clicks are counted.

3.2 Read Time Data

Coarse-grained revisit data can be augmented with a more fine-grained measure of relevancy obtained from runtime data. The time a user spends reading a job description has been shown to correlate with that user's degree of interest, ([2-6,9]). For this reason, CASPER also calculates runtime information from the server-logs by noting the time difference between successive requests by the same user. Again, a suitable thresholding technique is necessary to eliminate spurious readtimes due to a user logging off or leaving her terminal. However, the nature of read time data makes this task more difficult than that applied to revisit data. There is no trivial way of identifying misleading read time data which can arise for many reasons: end-of-sessions, distracted users, multiple windows, search-bots, etc. and this is compounded by the nature of connection speeds etc.

In order to prevent spurious readtimes (some with values of days or weeks) interfering with the identification of relevant jobs within a profile we adopted a two-step process. Ideally we wanted some "average" value for the time it takes to read a job, and replace any readtimes that deviate from the average too much, with the average value. Obviously this approach is not perfect, but the assumption is that it will not interfere with the identification of relevant jobs within the profiles as it is a middling value. However, due to the extent of spurious readtimes within the profiles, the average realtime for a user or for a particular job was strongly influenced by these outliers and was thus impractical. Our approach instead involved using the median of median realtime values per individual job access (as opposed to the total realtime over a number of visits to the job) for both users and jobs to calculate a 'normal' realtime for the system (1). Surprisingly the median of medians for both users and jobs was 48 seconds which we took as a reasonable value for a 'normal' length of time to read a job description.
SystemNormRT = (average(median(median((p_1), \ldots, median(p_n))), median(median((j_1), \ldots, median(j_m)))))

where: p_i \in Profiles, j_i \in Jobs.

The second step in tuning the realtime data in the profiles was to find any readtimes (per job access) within the profiles that had a realtime greater or equal to twice the system median (2 * 48secs). This produced a set of adjusted readtimes (2) where all the realtime values are 'reasonable'.

\[ \forall \text{jobaccesses}, \text{adjustedRT} = \begin{cases} 
\text{SystemNormRT} & \text{if rawRT} > 2 \times \text{SystemNormRT} \\
\text{rawRT} & \text{otherwise}
\end{cases} \]

For each job in a profile, the total adjustedRT was the sum of all individual adjusted readtimes for that job. Graded readtimes (3)per job were then produced by calculating in each profile, the number of standard deviations above or below the user’s average adjusted realtime, each job’s total adjusted realtime was for that user.

\[ \forall \text{jobs, gradedRT} = (\text{adjustedRT} - \text{average RT})/\text{stdevRT} \]

<table>
<thead>
<tr>
<th>User A</th>
<th>activity</th>
<th>raw revisit</th>
<th>thresholded revisit</th>
<th>raw rt</th>
<th>graded rt</th>
</tr>
</thead>
<tbody>
<tr>
<td>howjobs*317</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1753136</td>
<td>3.07939889</td>
</tr>
<tr>
<td>resconjobs*300</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>830</td>
<td>5.9226916</td>
</tr>
<tr>
<td>icejobs*6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>43</td>
<td>-0.11205088</td>
</tr>
<tr>
<td>howjobs*247</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>51</td>
<td>0.35216</td>
</tr>
<tr>
<td>resconjobs*285</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>161</td>
<td>3.0213726</td>
</tr>
<tr>
<td>wpgijobs*90</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>1779164</td>
<td>10.332694</td>
</tr>
<tr>
<td>cpljobs*1431</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>55</td>
<td>0.5842654</td>
</tr>
<tr>
<td>wpgijobs*54</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2410168</td>
<td>7.0251913</td>
</tr>
<tr>
<td>cmcngaleng*15</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>491</td>
<td>5.057167</td>
</tr>
<tr>
<td>howjobs*336</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>77</td>
<td>1.8608453</td>
</tr>
<tr>
<td>sljobs*277</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>41</td>
<td>-0.2281036</td>
</tr>
<tr>
<td>howjobs*316</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>715985</td>
<td>0.23610727</td>
</tr>
<tr>
<td>hennessyjobs*162</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>7500782</td>
<td>-0.32104632</td>
</tr>
<tr>
<td>sljobs*275</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1107430</td>
<td>1.338081</td>
</tr>
<tr>
<td>fjsjobs*506</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>19134</td>
<td>-0.02981327</td>
</tr>
<tr>
<td>prsjobs*391</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>-1.6207362</td>
</tr>
<tr>
<td>prsjobs*465</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>65</td>
<td>1.164529</td>
</tr>
<tr>
<td>prsjobs*442</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>106</td>
<td>3.543699</td>
</tr>
</tbody>
</table>

**Fig. 2.** Details of a profile

Fig. 2 shows the detail of a profile with the following fields from left to right: jobID, activity data - whether the user read, mailed to herself or applied for the jobs (5, 4, 0 resp.), raw revisit data, thresholded revisit data (spurious revisits removed), raw realtime and graded realtime (spurious readtimes removed and grading method applied). The diagram depicts a common realtime pattern where a user goes through a few sessions with JobFinder, each ending with a large (misleading) raw realtime due to the time difference between the
sessions. In Fig. 2, it appears that user A has probably had approximately 4 sessions with JobFinder during this period of time, and jobs with large raw readtimes like wpqijobs*90 and wpqijobs*54 are the last jobs the user looked at within the different sessions.

Fig. 3 focuses on the runtime data within the profile for user A, and shows the jobs sorted by raw runtime (left) and graded runtime (right). The diagram shows that when the jobs in the profile are ordered by the raw runtime data the precision of retrieving the most relevant jobs (those that are applied for, or to a lesser degree those that the user has mailed back to herself) is affected by other jobs with high spurious readtimes, for example hennesseyjobs*162, appearing at the top of the list. The recall is also affected, because jobs like resconjobs*300 and cmcngaleng*15 are pushed down the list by the jobs with high misleading readtimes.

However, the improvement is clear when the runtime data is transformed into graded runtime data which helps to remove erroneous readtimes. In fact, all the relevant jobs are correctly captured by this data, except for one (prsijobs*465).

![Fig. 3. Readtime details](image)

### 3.3 Activity Data

The final and perhaps most reliable way to judge user interest in a particular job is to make use of JobFinder’s online application or email facility. Briefly, a JobFinder user can either email a job description to themselves, for later consideration, or apply for the job directly online. These actions indicate a more serious interest in a job than a simple reading of the job description and for this reason CASPER takes note of the user activity (read(5), apply(0) or email(4)). For example, in Fig. 3, we can see user A has read 12 job descriptions, mailed 2 back to herself, and applied for 4 jobs online.

Obviously, a job will be highly relevant to a user if she applies for it online. However, users tend to do this infrequently, or not at all, resulting in insufficient data to exclusively base relevancy predictions on. It is
therefore necessary to consider these other relevancy measures, (readtime, revisits) to supplement this data, and it is interesting as these measures are common across a wide range of other web-based information filtering domains too.

4 Experimental Evaluation

So far, we have described the techniques used in CASPER to produce graded profiles that capture the job preferences of users. One of our central objectives, is to evaluate how these profiles (containing the implicit relevancy information) perform within the collaborative recommendation task [7], against simple profiles with no relevancy information, and ideally against profiles with relevancy information that has been obtained explicitly from the users. Given the difficulty however, in performing such an evaluation without access to a large group of users, we restrict ourselves here to a set of preliminary evaluations to test the value of CASPER’s implicit relevancy information, and in particular the measures of revisit data and readtime data.

The experimental study is based on the user profiles generated from server logs between 2/6/98 and 22/9/98, which contained 233,011 job accesses by 5132 different users. These profiles spanned 8248 unique jobs with an average profile size of 14.6 jobs.

Our evaluation of readtime and revisit data as relevancy predictors is based on the activity data of users. We assume that the action of a user applying for a particular job online is a reliable indicator of their interest in that job, and evaluate the readtime and revisit data based on how well it correlates with this information. In other words we are examining how well revisit and readtime data perform at predicting whether a user applied for a job or not. We are also testing in these experiments whether our improved measures of revisit and readtime (using thresholding techniques) data actually improve prediction performance. The experiments were therefore restricted to the set of those users who had applied for at least one job. Furthermore, we only took users with a profile size (number of jobs in profile) of 15 or greater. These users numbered 412 in total and were used as the profile base for the experiments.

4.1 Predictions with Revisit Data

For each user in the profile base we produced two sets of predictions for the jobs that the user applied for, based on the two kinds of revisit data: raw revisit predictions, and thresholded revisit predictions. Basically, the jobs in the profile were ordered with jobs that the user had visited most appearing at the top of the lists. For each set of predictions then, we produced 5 lists of the top k predicted jobs, for k = {1, 2, 5, 10, 15} for each user.

We then measured the precision (4) and recall (5) of each list:

\[
\text{precision} = \frac{\text{no. predicted jobs applied for}}{\text{no. predicted jobs}}
\]
\[
\text{recall} = \frac{\text{no. predicted jobs applied for}}{\text{no. jobs applied for}}
\]

Fig. 4 shows the results as precision against recall, each node on the trendline representing the different values for k, so we can see how these values vary as the size of the list increases. For each value of k (size of list) the results are averaged across the 412 users.
The graph shows that there is a clear correlation between revisit data and activity data (specifically jobs that the user has applied for online). It also shows that this data can be more finely tuned by removing the spurious irritation clicks to produce the thresholded version.

4.2 Predictions with Readtime Data

The readtime prediction experiments proceeded in a similar way to those for the revisit data - two sets of predictions were made one based on the raw readtime data, and the other based on the graded readtime data. For each set of predictions, 5 lists of the top k (={1, 2, 5, 10, 15}) predicted jobs were produced again, and the results averaged over the 412 users. The prediction quality was again measured by precision (4) and recall (5).

The results are shown in Fig. 5 as precision against recall for the different values of k.

The graph shows that there is only a loose relationship between raw readtime and activity data. This is because of the large amount of noise that this type of data is subject to, such as a user logging out or leaving her terminal. However, a significant improvement in the correlation between readtime and applying for a job is gained, by refining the data into graded readtimes that eliminate some of the erroneous information.
Finally, Fig. 6 compares the results obtained with revisit data and realtime data. It shows that the thresholded revisit data performs the best, followed by the graded realtime data. We believe the revisit data performs better because it is less subject to noise than the realtime data which shows little correlation to the activity data in its most raw form.

![Precision V Recall: raw realtime vs graded realtime vs raw revisits vs thresholded revisits](image)

**Fig. 6.** Relevancy prediction quality of realtime data and revisit data

5 Future Work

So far we have shown how the most relevant items in a user profile can be identified using measures of revisit data and realtime data. Ultimately, the goal is to use this information in the collaborative recommendation task [7], i.e. where we are not only concerned with the most relevant jobs for a single user, but rather with the similarity between two users based on this relevancy information. This presents another important issue to be considered when analysing the value of revisit and realtime data. As an example, consider that a user has accessed jobA 10 times; is that job 5 times as relevant to that user as jobB that she has looked at twice? It is our hypothesis that although jobA is more relevant to the user, it is not 5 times more relevant. Therefore, we believe that rather than using a linear model of revisit and realtime relevancy, a logarithmic type of model would be more appropriate, Fig. 7. In the future, we plan to investigate this hypothesis, and fully integrate and evaluate the relevancy measures of revisit and realtime data into CASPER’s collaborative recommendation system.

![Linear V logarithmic model of revisit relevancy data](image)

**Fig. 7.** Linear Revisit Model
6 Conclusions

One of the basic assumptions underlying recent research focused on developing user profiling applications for the Internet, is that it is possible to derive user preferences by monitoring measures like the click-stream and realtime data generated in server logs from user sessions. However, although this assumption seems broadly accepted there is relatively little published work that tests the validity of this assumption, [5].

In this paper we have describe such a study in the context of the CASPER project, which is concerned with developing user profiling techniques for online services like the JobFinder recruitment service. We have described how JobFinder’s server logs are mined to generate different types of user profiling information derived from normalised click-stream and realtime data. The results although preliminary, are promising as they indicate that both realtime and revisit data are useful in predicting jobs that individual users have applied for, which we believe is indicative of serious user interest. Furthermore, we have provided a comparative evaluation to demonstrate the differential performance of these techniques.

References


