<table>
<thead>
<tr>
<th>Title</th>
<th>Personalised Retrieval for Online Recruitment Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors(s)</td>
<td>Rafter, Rachael; Bradley, Keith; Smyth, Barry</td>
</tr>
<tr>
<td>Publication date</td>
<td>2000-04-05</td>
</tr>
<tr>
<td>Conference details</td>
<td>The BCS/ IRSG 22nd Annual Colloquium on Information Retrieval (IRSG 2000), Cambridge, UK, 5-7 April, 2000</td>
</tr>
<tr>
<td>Item record/more information</td>
<td><a href="http://hdl.handle.net/10197/4636">http://hdl.handle.net/10197/4636</a></td>
</tr>
</tbody>
</table>
Personalised Retrieval for Online Recruitment Services

Rachael Rafter, Keith Bradley, Barry Smyth
Department of Computer Science, University College Dublin Belfield, Dublin 4, Ireland

Abstract

Internet search technology is largely based on exact-match retrieval. Such systems rely on the user to provide an adequate description of their requirements. However, most users submit poorly specified queries, leading to imprecise search results. Furthermore, there is no facility for personalising searches to reflect implicit likes and dislikes of users. We describe two complementary solutions to these problems as implemented in CASPER, an intelligent online recruitment service. We describe an approach to personalised similarity-based retrieval, and a queryless collaborative filtering recommendation technique.

1 Introduction

The Internet currently consists of approximately 800 million pages of information across more than 8 million Web sites, and there are over 200 million regular users from all walks of life with varying degrees of computer literacy and a wide range of interests and preferences. In particular Online Recruitment services have proved to be one of the most successful and popular information services on the Internet. Typically these sites provide job seekers with a comprehensive database of categorised jobs, a dedicated search engine, and the ability to submit resumes and apply for jobs online. The award winning Irish site, JobFinder is a good example of one such service.

However, retrieving the right information for a particular user given some target query is once of the most difficult information retrieval tasks imaginable. Moreover, our current information retrieval techniques (as they are used in modern search engines) fall short of the mark in providing an effective information retrieval service.

There are several fundamental problems with current approaches. Most search engines rely on exact-match retrieval constraints, are limited to a client pull retrieval model and do not provide a personalised retrieval service that accounts for the implicit likes and dislikes of a particular user. Furthermore, the fact that the average Internet user is not an information retrieval expert means that the average search query is poorly constructed and either too broad or too specific; in fact recent studies indicate that the average search query contains only two or three search terms. The inevitable outcome is that the average user is faced with search results that run into thousands or even tens of thousands of supposedly relevant documents. This of course is totally impractical, and the average user rarely gets beyond the first page or two of search results, and often fails to find the document that they are looking for.

In this paper we describe the work carried out in the CASPER (Case-Based Profiling for Electronic Recruitment) project. Very briefly, the aim of CASPER is to build a more intelligent search engine for use in specialised Internet information retrieval applications – specifically our application test-bed is the JobFinder web site. We will present both these approaches in this paper, giving a detailed description of their design, as well as some preliminary experimental results.

2 The CASPER System Architecture

The CASPER system was designed to run as a complementary service to the original Job Finder search engine. The results produced by CASPER were to enhance or augment the original search results. Both of the CASPER Systems are shown in Figure 1. Initial data for both of the CASPER systems was drawn from the job database and the log files of the Job Finder web server. It is worth noting at this point that the point at which a user begins interacting with CASPER differs for each of the CASPER systems. This reflects the different approaches to personalisation taken by both CASPER Automated Collaborative Filtering (ACF) and CASPER Personalised Case Retrieval (PCR).

CASPER ACF is a content free system that starts tracking the user from the moment that they login to the web site. It combines two server side components: a User Profiling System that constructs a user profile of each user’s preferences, gathered by monitoring their behaviour within the JobFinder site, and a queryless Automated Collaborative Filtering engine, which uses this profile information to generate personalized recommendations to a user based on what similar users like.

CASPER PCR uses a two stage personalisation process. The first stage uses a server side similarity based retrieval engine to provide an intelligent database query system. The second stage is a client-side content based
personalisation engine which only uses an individual users history to personalise results obtained during the first stage while benefiting from improved privacy due to its client side profiling technique (e.g., [4, 5, 10, 11, 12, 13, 14, 16]).

![CASPERS System Architecture](image)

**Figure 1. CASPER System Architecture**

### 3 Automated Collaborative Filtering in CASPER

In this section, we will look at the role of ACF (Automated Collaborative Filtering) within CASPER as the core component of its query-less recommendation service. In many ways ACF techniques are uniquely suited to the web where it is possible to automatically collect useful profile data on a large number of users. Briefly, ACF is a feature-free recommendation technique that selects information items to recommend for a particular user on the grounds that other similar users have previously expressed an interest in these items (e.g. [1, 2, 3, 6, 8, 11, 15]). In this sense ACF implements a computational model of a familiar social recommendation policy. The success of ACF depends critically on the availability of high-quality user profiles, which contain graded lists of information items, and the technique is shown to work well when there is a high expected overlap between related profiles. However, JobFinder’s large dynamic database of jobs, each with a limited life span, means that the expected overlap between similar users can be vanishingly small (see also [2]). In the following section, we will discuss how this characteristic presents a problem to one common form of ACF and we describe the solution being investigated in CASPER.

#### 3.1 User Profiling in CASPER

User profiles are the primary source of knowledge in an ACF system. In CASPER, user profiles are automatically and passively constructed by mining JobFinder’s server logs for the relevant information. One of the central objectives is the construction of user profiles without interfering with the way that users interact with JobFinder. Hence, unlike many related systems CASPER does not require explicit graded feedback from users to construct its user profiles [6, 12, 13].

Figure 2 shows a small part of a JobFinder server-log. Each line records a single job access by a user and encodes details such as the time of the access, the type of access, and the job and user ids. The basic form of a user profile is simply a list of jobs that a given user has looked at over the course of their entire interaction with the JobFinder system. Of course, a simple list of job ids will not provide a very accurate picture of a user’s preferences. In general, a given user is likely to have different levels of interest in the different jobs they have looked at, and this implicit relevancy information can be added to the profile representation in a number of ways. CASPER collects three types of relevancy information – revisit data, read time data, and activity data.
Revisit Data: The number of times that a user clicks on a job is thought to be an indicator of their 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. For this reason CASPER logs the number of times that each user clicks on a job – a thresholding technique is used to identify any misleading data e.g. so-called "irritation clicks" where a user repeatedly clicks on a job in quick succession while waiting for a description to download, e.g. in Figure 2 we see how 3 clicks on richjobs*869 in quick succession are collapsed into one in the profile.

Read Time Data: Coarse-grained revisit data can be augmented with a more fine-grained measure of relevancy obtained from read-time data [6, 9]. The time a user spends reading a job description should correlate with that user’s degree of interest. For this reason, CASPER also calculates read-time information from the server-logs by noting the time difference between successive requests by the same user. Again, a suitable thresholding technique is used to eliminate spurious read-times due to a user logging off or leaving her terminal. For example, in Figure 2 we can see how the read time for five different jobs is calculated from the server logs.

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. 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, apply or email); For example, in Figure 2 we can see that user "Rachael" has read three jobs, emailed a copy of another, and then actually applied online for another, (encoded as 5, 4, and 0 respectively in log file).

Further, in our research so far we have examined each of these factors as a measure of relevance. However, to date we have not found any significant improvements over the simple ungraded profile representation. It seems that unlike other domains [9], simple measures of clicks or readtimes do not act as accurate relevancy indicators and we believe that a more complex measure that combines clicks, readtime and activity data is needed – however, this is beyond the scope of this paper.

What is perhaps more important in this paper is the nature of CASPER’s profile space in relation to the expected overlap between individual user profiles. There are a total of 5132 active profiles in the system covering 8248 unique job descriptions. Moreover, on average, a profile will contain approximately 14 jobs and therefore the expected overlap between any two profiles will be very low. Even the closest matching profiles will typically overlap by only a few jobs. We will argue in the next section that this presents a significant problem for at least one common type of collaborative filtering technique.
3.2 Collaborative Filtering and the Problem with Relationships

Collaborative filtering is a recommendation technique that draws its recommendations (for a given target user) from the profiles of other related users in a two-step process. First, a set of users that are related to the target user are identified, and second, profile items from the related users, that are not present in the target profile, are ranked for recommendation to the target user. The success of collaborative filtering as a recommendation technique depends critically on the ability to recognise those users that are genuinely related to a given target user in the sense that they will act as high quality recommendation partners. In this section we will introduce two different flavours of collaborative filtering (memory-based collaborative filtering and cluster-based collaborative filtering) that rely on different techniques for identifying relevant users. In addition, we will argue that only one of these flavours is likely to be useful within the CASPER context due to the sparseness of CASPER’s profile space.

3.2.1 Memory-Based Collaborative Filtering and Direct Relationships:

Memory-based collaborative filtering is probably the simplest form of the general collaborative filtering approach. Users are related on the basis of a direct similarity between their profiles, for example, by measuring the degree of overlap between their profile items, or by measuring the correlation coefficient between their grading lists [2, 12, 13]. This leads to a lazy (in the machine learning sense) form of collaborative filtering whereby the target user is used to select the k nearest profiles. Currently CASPER uses a simple overlap metric (Definition 1) to determine profile similarity.

\[
\text{Overlap}(t, p) = \frac{|\text{Items}(t) \cap \text{Items}(p)|}{|\text{Items}(t) \cup \text{Items}(p)|}
\]

\[
\text{Quality}(j, t, P) = \sum_{p \in P, j \in P} \text{Overlap}(t, p)
\]

**Definition 1:** Overlap {where: t and p are profiles (t being the target profile) and j is a job}

**Definition 2:** Quality {where: t and p are profiles (t being the target profile) and j is a job}

Once the k nearest users (base profiles) have been identified, their recommendable items (jobs that are not present in the target profile) are ranked according to the metric shown in Definition 2. This measure biases jobs in two ways: jobs that occur more frequently in the base profiles are preferred over jobs that occur less frequently, and jobs that occur in profiles that are very similar to the target are preferred over jobs that occurs in less similar profiles.

![Figure 3. Direct vs. Indirect User Relationships.](image-url)
indirect relationship. In short, C may be a good recommendation partner but this cannot be recognised in this form of collaborative filtering.

In many sparse-profile domains, such as CASPER’s, the expected overlap (similarity) between two user profiles is likely to be vanishingly small and thus the number of other users with significant overlap with a given target user is also likely to be very low [2, 10]. As a result, the number of users with a direct relationship with a target can be very low indeed, and the target user’s recommendations may be based on a very small number of profiles with very low degrees of similarity. At best this may compromise the quality of the recommendations, and at worst it may result in no recommendations at all for a user.

### 3.2.2 Cluster-Based Collaborative Filtering and Indirect Relationships

Fortunately there is an alternative collaborative filtering method that relies on the detection of both direct and indirect user relationships. The solution is to make use of (eager) profile clustering techniques in order to group users prior to recommendation – profiles are clustered into virtual communities such that all of the users in a given community are related, (e.g., [6]).

In Figure 3(b) we demonstrate a common scenario where users A and B are directly related, as are users B and C. There is no direct relationship between A and C, however there may be an indirect one. Consider that A uses JobFinder during March and C uses JobFinder during June, and that both users are similar, i.e. they both search for similar types of jobs e.g. Java programming jobs in Dublin. Due to the dynamic nature of the job market, the jobs that are available to A in March, and the jobs that are available to C in June will be different, and therefore their profiles will contain different jobs. If relationships are measured based purely on direct relationships the indirect similarity between A and C will be lost. However, allowing indirect relationships using virtual communities, if B has used JobFinder through March and June, then the transitive relationship between A and C through B will be captured too. As user relatedness is transitive, users A, B, and C should belong to the same virtual community along with all other users that are directly or indirectly related to A, B, or C. In order to specifically exploit this form of indirect relationship the single-link clustering technique can be used with a thresholded version of the similarity metric from Definition 1; essentially each community is a maximal set of users such that every user has a similarity value greater than the threshold with at least one other community member.

Once communities of related users are constructed the recommendation process can then proceed in a way that is analogous to the memory-based approach, except that instead of selecting k neighbours for the target profile, we select the members of the target profile’s community. Of course, the immediate benefit of this cluster-based approach is that it is possible to identify larger groups of users that are related to the target user and thus provide a richer recommendation base.

$$Quality(j, P) = \frac{|\{p \in P : p \text{ contains } j\}|}{|P|}$$

**Definition 3:** Quality (where j is a job and P is a community of profiles)

A disadvantage of this cluster-based method is linked to the fact that it is no longer possible to judge the direct similarity between all pairs of profiles in a community, as there may be no direct relationship between member pairs. For this reason, the grading of items for recommendation must be based solely on their frequency in the community only (Definition 3) and no priority can be given to the recommendations of particular users.

### 3.3 Experimental Analysis

At this point, we have described two basic collaborative recommendation strategies for use in CASPER. In this section, we describe a preliminary evaluation to test the quality of CASPER’s ACF component. Specifically we evaluate the quality of the job recommendations produced by each flavour of ACF, the lazy, k nearest neighbour version and the eager, cluster-based approach.

The experimental study is based on the user profiles generated from server logs between 2/6/98 and 22/9/98. These logs contained a total of 233,011 job accesses from 5132 different users. These profiles spanned a total of 8248 unique jobs with an average profile size of approximately 14 jobs and nearly 3000 profiles containing less than 10 jobs – and indication of CASPER’s extremely sparse profile space.

Unfortunately, we had no way of automatically evaluating the recommendations produced by the two ACF versions. Instead the evaluation had to be carried out by hand and for this reason we restricted our evaluation to a small set of users. Ten target users were selected, each from a different virtual community. Furthermore, the communities to which these target users belong were chosen to cover a range of different community sizes (from
small to large). For each target user, we produced two recommendation lists containing ten jobs:

**Memory (ACF-NN):** The list of jobs recommended according to the memory-based version of ACF. That is, each target user is associated with its k nearest users (k=10 in this experiment) and a ranked recommendation list based on the equation in Definition 2 is produced.

**Cluster (ACF-Cluster):** The list of jobs recommended according to the cluster-based ACF approach. That is, each target user is recommended the most frequent jobs in its virtual community (Definition 3).

Both sets of results for each target user were then manually graded as good, satisfactory, or poor (mapped on to a numeric value of 3,2, or 1 respectively), based on how similar the recommended jobs were to the existing jobs in each target user profile – for these preliminary experiments the grading was performed by the researchers involved in the project, a more elaborate evaluation is planned and will include a range of different graders that are not associated with the CASPER project. Therefore, every target user receives a cumulative grading score across the 10 recommended jobs from each ACF technique.

Each grading score is normalised by dividing by the maximum cumulative grade of 30 and presented in Figure 4 for each target user. Figure 4 also encodes a measure of cluster size so that we can see how the recommendation quality behaves for different cluster sizes using the cluster-based ACF method.

It is clear from the results that the cluster-based method is outperforming the memory-based version for all target users (except those that hail from the smallest virtual communities). We believe that the reason has to do with the sparseness of the profile space. As explained in Section 3.2, the expected overlap between users is very small and therefore many of the 10 nearest users for a given target chosen by ACF-NN may exhibit only very low degrees of similarity to that target. In fact, many of these nearest users may overlap with the target profile by just 2 or 3 jobs and therefore may not constitute a reliable recommendation partner for the target user. This means that we can expect part of the recommendations for the ACF-NN method to come from unreliable recommendation sources.

![Figure 4. Recommendation quality for memory-based and the cluster-based techniques.](image.png)

In contrast, the ACF-Cluster method is basing its recommendation on potentially more reliable measures of indirect relationships between profiles. In the case of this experiment the similarity threshold used to construct the virtual communities was set at 10 and therefore implicit relationship are based on a transitive overlap of 10 between community members. However, the ACF-Cluster method does show quality degradation for small cluster sizes. Again, this is to be expected as these small clusters do not provide a rich enough recommendation source for their target users.

### 3.4 Discussion

In this section we have looked at enhancing the JobFinder system with CASPER’s collaborative filtering component - a personalized recommendation facility. Specifically, we have described how implicit user profiles can be automatically generated from JobFinder server logs and how personalised recommendations can be formed using two different collaborative filtering approaches, one based on a k nearest neighbour technique (memory-based collaborative filtering) and another based on an explicit clustering technique (cluster-based collaborative filtering).

We have argued that the sparseness of CASPER’s profile space presents significant problems for the memory-based k nearest-neighbour approach to collaborative filtering because the expected overlap between profiles in CASPER is likely to be very low. The cluster-based approach is presented as a solution to this problem as it can
exploit transitive overlap relationships between profiles that otherwise do not overlap. This means that it is possible to recognise related users even if their profiles contain no overlapping items. We have argued that this property makes the cluster based technique especially appropriate in the case of CASPER. This hypothesis has been backed up by some preliminary experimental results and we expect to add a more comprehensive evaluation in the near future.

4 Personalised Case Retrieval in CASPER

Case-based reasoning research has been successfully applied to implement a more flexible and powerful information retrieval service in a wide variety of industrial applications in recent years, and CBR systems are now becoming an important component of many advanced e-commerce Web sites [5, 7, 12, 13, 14, 16]. CASPER PCR pushes the envelope even further by extending the case-based reasoning strategy to implement a form of personalised case retrieval that, we argue, constitutes a significant step forward in terms of Internet search technology.

JobFinder’s large-scale and reliance on exact-match retrieval makes it an ideal test-bed for CASPER’s similarity-based personalisation methods. Our principal aim then is to enhance JobFinder’s conventional browse and search functionality with personalised search and recommendation facilities. For example, a JobFinder user might specify that they are looking for a software development job with a salary of £25k. However they may actually be looking for a permanent C++ or Java software development job, with a salary in the region of £25k (preferably greater), in the Dublin or Wicklow area. In other words, the supplied query is usually only a very small part of the equation. Furthermore, two users with the same queries will receive the same search results even though they may both have very different personal preferences – a second user, looking for a “software development job in the region of £25k”, may actually be looking for a Visual Basic job in Cork, and not want to see information on Dublin, C++ or Java jobs.

In this section we will describe and illustrate the mechanics of the CASPER PCR component. We explain how structured job cases are automatically constructed from ill-structured raw data and how the PCR retrieval system uses a two stage retrieval process to first select similar jobs to a target query and second to personalise these retrieval results according to their relevance to a given user’s profile.

4.1 Sub-System Architecture

The CASPER PCR component is shown in Figure 5 and comprises of 2 main sub-systems. The server-side similarity based retrieval engine manages the similarity assessment and retrieval of job cases from the job case-base; essentially this CBR engine is a replacement (supplement) for the existing JobFinder retrieval system. In addition, there is a client-side personalisation component responsible for managing individual user profiles and for using these profiles to rank retrieved cases in a way that reflects not only their similarity to the target query, but also their similarity to a user’s profile.

![Figure 5. CASPER PCR System Architecture](image-url)
4.2 Job Case Representation

Before describing the details of CASPER’s retrieval and personalisation components we will focus on the nature and representation of its case-base. In the original JobFinder system each job is represented as a single database record with various fields detailing different aspects of the job in question. For example, there are fields for the job type, salary, location, required experience etc. However, JobFinder receives its data feed from many different recruitment agencies and at the present time there are minimal guidelines regarding the vocabulary to use when describing new jobs. No restrictions or guidelines for information content or style are given to the job authors. A wide variety of expressions could therefore be used to represent the same information. Furthermore no checks for misspellings, incompatible data (text instead of digits) or data entry in incorrect fields (description in the salary field) are performed. The end result is a set of error filled jobs described with various mixed terms.

In order to explore a case-based reasoning option these semi-structured job records must be translated into a more structured case format. This is carried out using an automatic translator, which is essentially a rule-based system for translating feature values. The first stage in the translation process is to identify relevant data from the job description. The other information is discarded, as it is not relevant to the search methods used during retrieval; however a complete original description of the job is still retained for display to the user. The series of translators employed also standardise the information extracted from the fields. For example the original salary field contains various descriptions such as 23, 23000, 23k etc. The salary period corresponding to theses figures also fluctuates. These various forms are all translated into an annual salary range description, for example 20000 – 30000, to allow for the easy comparison of these fields. Domain specific thesauri are also employed to standardise known job skills and locations. Compound fields such as skills are also parsed into their individual components. A final structured case is shown in Table 1.

<table>
<thead>
<tr>
<th>Description</th>
<th>“Original Job Text”</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID Reference</td>
<td>Management Team Leaders</td>
</tr>
<tr>
<td>Work Type</td>
<td>Permanent</td>
</tr>
<tr>
<td>Duration</td>
<td>0</td>
</tr>
<tr>
<td>Location</td>
<td>Dublin</td>
</tr>
<tr>
<td>Salary</td>
<td>20000 – 35000</td>
</tr>
<tr>
<td>Minimum Experience</td>
<td>2</td>
</tr>
<tr>
<td>Key Skills</td>
<td>Java, Internet, WWW</td>
</tr>
</tbody>
</table>

Table 1. The transformed structured job case suitable for similarity matching.

4.3 Stage 1: Similarity-Based Retrieval

CASPER PCR retrieves candidate jobs that it considers suitable matches for a target job description provided by the user. Job suitability is determined by the overall similarity of the job case in question to the target query. As mentioned in the previous section, each target and job case is made up of a number of different features (see sample case insert in Figure 5 and also Table 1) and during retrieval the similarity of a target (t) to a current job case (j) is calculated as the weighted sum of the individual feature similarities (t, and j, respectively) (see Definition 4).

\[ Sim(t, j) = \sum_{i=1}^{n} similarity(t_i, j_i) \times w_i \]

Definition 4 Similarity of two jobs t, and j (where \( w_i \) is the weight of the ith target and case feature \( t_i \) and \( j_i \), respectively)

Since a job case will contain different feature types (e.g., numeric values, numeric ranges, and symbolic values) CASPER employs a range of different feature-similarity functions – numeric values and ranges are dealt with by standard value difference metrics while symbolic values are compared by measuring concept distances in a concept-tree. As an example we will consider the similarity between a target query and a candidate retrieval job; shown in Table 2(A) and Table 2(B) respectively; looking at the similarity function used to compare each field in turn.
Table 2(A): A sample search query, (B): a retrieval candidate

The Work Type field uses a simple exact-match string comparison to determine whether or not both jobs are permanent or contract. If a match is found this field gets a similarity score of one, if not they get a score of zero.

Integer ranges represent salaries. Salaries can be represented by atomic values e.g. 30000, or as ranges e.g. 30000 - 40000 etc. Similarity here is determined by whether or not a value falls into the relevant range. If a target salary falls above the target range it is also taken to be an exact match since it’s actually a better offer than the user was looking for.

CASPHER performs two types of similarity-based matching based on concept trees. These techniques are only applied to those fields about which some background knowledge is available. This background knowledge is encoded in the form of concept trees as shown in Figure 6(A). These hierarchical representations of domain knowledge show the relationships between concepts in a domain.

The first form of similarity based matching is called subsumption matching (see Figure 6(B)) and allows for different granularities of abstraction to be used in a search query. For example in Figure 6(B) the user has submitted a query for OOP or Object Oriented Programming based jobs. Any job containing a concept that is a descendant of OOP in the concept tree is taken to be an exact match. That is any job that contains say Java or C++ is taken to be an exact match for an OOP job. This is logical since all descendants of OOP in the tree are object oriented programming skills.

CASPHER also determines concept similarity based on concept proximity as shown in Figure 7(A) and 7(B). In other words the closer two concepts are in the tree the more similar they are taken to be. As an example in Figure 7(A) the distance between C++ and Java is 2. These concepts are therefore taken to be quite similar. Distance is measured in terms of the number of inter nodal edges or links between the two concepts. By contrast the similarity between Java and Pascal is judged to be quite low as shown in Figure 7(B) since there are four edges between them.
It is quite possible for concepts to recur within a given concept tree, that is, for a concept to have multiple parents. The final similarity score is then taken to be the best similarity score from all of the permutations of candidate / target available.

4.4 Stage 2: Personalising Retrieval Results

As it stands CASPER’s CBR engine allows job cases to be retrieved on the basis of a fluid feature-based similarity assessment procedure. The advantage for the end-user is a more flexible retrieval system that is capable of identifying partial and inexact matches to target queries. However, the CBR component cannot personalise the search for individual users; that is, two users entering the same search query will receive the same results, ranked in the same way. We believe that this is fundamentally flawed. Clearly, different users can have different tastes and preferences and, even though the user may not explicitly provide this information, it should play a role in the retrieval process.

CASPER PCR uses a novel approach to personalising retrieval. Like many related approaches it bases its personalisation on a profile of the user that built up over time as the user interacts with the system [1, 2, 3, 4, 6, 8, 15]. As users use the system and receive search results, they are given an opportunity to rate the individual jobs that are retrieved for a particular query (as being either a liked or disliked job), and these grades, together with the relevant job cases are stored in the user’s profile. However, unlike many traditional personalisation techniques (such as collaborative filtering), these profiles are stored and manipulated on the user’s (client) machine, and personalisation is carried out at the client-side rather than the server-side. This offers significant benefits when it comes to security, privacy, and efficiency. For a start, personal user information does not have to be passed to a remote server, and so individual users can feel secure that their personal information is not being collected or exploited from some central repository. In addition, the computational load is shifted from the server-side to the client-side thereby improving the overall efficiency of the personalisation system.
The key to CASPER PCR’s personalisation facility is the idea that jobs should not just be retrieved because they are good matches to the target query, they should also match the user’s personal preferences. In our system these preferences are represented by previously graded jobs. We view this as a version of the traditional classification problem. A candidate retrieval job is compared to those already in a users profile. CASPER PCR uses a form of k nearest neighbour matching to calculate relevance. The candidate job’s k nearest neighbours (where k is an odd number) are then used to classify the candidate as either relevant or irrelevant as shown in Figure 8. A value for k varies depending on the domain being examined and must be obtained experimentally. Each entry in the users profile therefore influences the classification of the retrieval candidate. K nearest neighbour methods differ in how much influence each profile entry has over the final classification. CASPER PCR uses a weighted metric whereby the similarity of a profile entry to the retrieval candidate and the users classification of that profile element determine how much influence that profile entry has in the overall calculation. The metric for computing relevance is defined in

\[
\text{Rel}(j,P) = \begin{cases} 
1 & \text{if } \left( \sum_{p \in knn(j,P)} \text{Sim}(j_p, p).\text{classification}(p) \right) > 0 \\
-1 & \text{otherwise} 
\end{cases}
\]

**Definition 5 Relevance Formula.** (where j is a retrieval candidate, P the users profile, classification(p) a function which yields the user’s classification for a job p, and knn(j, P) a function that returns the k nearest neighbours to a target job j in a user profile P)

In this way we now have two techniques for ordering search results for the user, by their similarity to the target query, or according to their relevance to the target user. Thus, CASPER’s PCR personalisation client ranks, filters and sorts retrieved jobs in terms of their similarity to the target and their relevancy to the user. Furthermore, as each user’s profile evolves the personalisation facility will automatically adjust to that user’s true preferences. Moreover, if a user’s preferences change over time, her profile will automatically change to reflect this (as additional gradings are taken into account), and so too the personalised retrieval results will adapt.

**4.5 User Trials**

Some preliminary experiments were carried out in order to determine a value for K and to test the classification accuracy of the personalisation engine. A set of approximately 60 jobs was drawn from a larger casebase of 3800 jobs so as to provide a reasonable spread of job features. Eleven subjects were asked to classify a set of approximately 60 jobs as to whether or not they liked or disliked them.

A form of leave one out testing was then used to analyse the accuracy of the personalisation. Each job from the set was used in turn as a retrieval candidate. This candidate was used to simulate a job suggested by the first stage similarity based retrieval engine. The remaining user classified jobs were used as a profile. The personalisation engine used this profile to classify the retrieval candidate. The accuracy of the personalisation was determined by comparing this automatically generated classification with that originally given to the retrieval candidate by the user.

The value for K used in the personalisation process was varied between one and fifteen in order to obtain a value of K which maximised the accuracy of the personalisation engine. The average accuracy across all the users – approximately 600 classifications in total, plateaued at values of k between 7 and 11. A graph of the classification accuracy for each of the eleven users, with a value of k equal to 7 is shown in Figure 9.
This clearly shows that the personalisation engine consistently achieved a high level of accuracy in predicting users’ likes and dislikes when 7 nearest neighbours were taken into account.

5 Conclusions

We have presented the CAPSER project – a personalised intelligent information retrieval service for specialist search engines like the JobFinder online recruitment system. Within CASPER, we describe two main systems: CASPER ACF and CASPER PCR, which address the task of making the information retrieval process more personalized and intelligent in two different but complementary ways.

The ACF system provides a queryless, content-free recommendation server, which tracks the user as they search and recommends new items to the user based on what similar users have liked before. Specifically we have described how implicit user profiles are automatically generated by the User Profiling component, and how these profiles then supply an Automated Collaborative Filtering engine, which uses them to produce personalized recommendations.

We have shown how CASPER PCR provides a two stage retrieval system by augmenting a server-side similarity based retrieval engine with a post processing client-side personalisation engine. This means that retrievals not only match the details of a user’s query, but also their implicit preferences. The personalisation approach is generally applicable, since it relies on standard attribute-value type content representation. Moreover, we argue that the approach derives significant security and privacy benefits because of its client side implementation. Preliminary user trials suggest that the system consistently achieves a high level of accuracy in predicting user preferences.

References

5. Kolodner J. Case-Based Reasoning. Morgan Kaufmann. 1993


