Relevance and ranking in online dating systems

3 authors, including:

Fernando Díaz
Universidad de Valladolid
88 PUBLICATIONS **1,994** CITATIONS

SEE PROFILE

Sihem Amer-Yahia
225 PUBLICATIONS **4,358** CITATIONS

SEE PROFILE

Relevance and Ranking in Online Dating Systems

Fernando Diaz
Yahoo! Labs
4401 Great America Parkway
Santa Clara, CA 95054
diazf@yahoo-inc.com

Donald Metzler
Yahoo! Labs
4401 Great America Parkway
Santa Clara, CA 95054
metzler@yahoo-inc.com

Sihem Amer-Yahia
Yahoo! Labs
111 West 40th Street
17th Floor
New York, NY
sihem@yahoo-inc.com

ABSTRACT

Match-making systems refer to systems where users want to meet other individuals to satisfy some underlying need. Examples of match-making systems include dating services, resume/job bulletin boards, community based question answering, and consumer-to-consumer marketplaces. One fundamental component of a match-making system is the retrieval and ranking of candidate matches for a given user. We present the first in-depth study of information retrieval approaches applied to match-making systems. Specifically, we focus on retrieval for a dating service. This domain offers several unique problems not found in traditional information retrieval tasks. These include two-sided relevance, very subjective relevance, extremely few relevant matches, and structured queries. We propose a machine learned ranking function that makes use of features extracted from the uniquely rich user profiles that consist of both structured and unstructured attributes. An extensive evaluation carried out using data gathered from a real online dating service shows the benefits of our proposed methodology with respect to traditional match-making baseline systems. Our analysis also provides deep insights into the aspects of match-making that are particularly important for producing highly relevant matches.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms,Experimentation

Keywords

dating systems, relevance

1. INTRODUCTION

A match-making system is a bulletin board where people seek to meet other individuals in order to satisfy a particular need. Many match-making systems exist today including dating services, resume/job bulletin boards, community-based question answering systems, and consumer-to-consumer marketplaces. Despite the popularity of such systems, relatively little research has been conducted on the design of information retrieval models particularly suited for the match-making task.

Typically, in a match-making system, each user is associated with a profile that includes general information about the user. For example, in an online dating service, the profile will include the user's location, physical attributes (e.g., hair color), and political affiliation. On a job seeking site, the profile may contain the job seeker's education, years of experience, and desired salary range. It is also common for users of these systems to be able to define the attributes they would like matches to satisfy. In match-making systems, these are often called *target profiles*. In information retrieval terminology, the target profile can be considered the user's information need, or query. A key aspect of match-making systems is the ranking of candidate matches for a given user. Different ranking functions are used to compute the relevance of a match to a user. In this paper, we study information retrieval-based ranking functions in this context. To the best of our knowledge, this is the first in-depth study of retrieval and ranking in a match-making system.

We use an online dating service as our main application. One key observation is that in such systems, *two-sided* relevance is a natural way of ranking matches. Intuitively, matches that satisfy a given user's target profile and *whose target profile is also satisfied by the given user own profile*, are preferred to matches whose target profile is not satisfied by the given user. Consider the situation where user u is interested in someone with attributes similar to those of user v but v is not interested in someone with attributes similar to those of user u. In this case, we argue that it is undesirable for a retrieval system to rank v highly for u. There are two reasons for this. First, we would like to avoid v being contacted by undesirable candidates. Second, we would like to maximize the likelihood that u receives a reply. Another interesting aspect of the dating domain is subjective relevance. Understanding the relevance of a pair of individuals often requires a complex understanding of the intents of both users. This makes a good match more difficult to detect than document relevance and as a result is particularly interesting from an information retrieval perspective.

This paper has three primary contributions. First, we formulate the match-making task as an information retrieval problem, whereupon user profiles are ranked with respect to a given target profile using a machine learned ranking function. Second, we exploit the uniquely rich nature of user profiles by extracting novel features based on their structured (e.g., age, income) and unstructured (e.g., description) attributes. Finally, we undertake an extensive evaluation using the data from a real-life online data site in order to determine the level of interest between two users. Our experiments also provide interesting insights into the types of features and models that are the most useful for these types of systems.

The rest of this paper is laid out as follows. First, Section 2 provides an overview of previous work related to match-making systems. Then, in Section 3 we formally define the match-making problem from an information retrieval perspective. Section 4 discusses our proposed machine learned ranking framework. Sections 5 and 6 detail our experimental methodology and present our experimental evaluation, respectively. Finally, Sections 7 and 8 conclude the paper with some discussions and directions for future work.

2. RELATED WORK

There is a large body of research related to match-making systems. Previous research has looked at match-making from algorithmic, empirical, and even psychological perspectives. We now highlight the previous research that is most related to our study.

First, in mathematics there is the well-known stable marriage problem. This problem aims to find a matching between all pairs of men and women such that it cannot be the case that two potential partners both prefer each other to their current partner [11]. When there is no distinction between sets of users, this is the stable roommate problem [13]. In combinatorics, the assignment problem refers to matching tasks and agents so as to minimize some cost. In our situation, we are not required to provide a hard matching between users, but rather to rank possible partners for a given user. This distinction is necessitated by the interactive nature of online match-making systems.

There have also been a number of studies that have specifically looked at dating systems. Hitsch *et al.* provide a very detailed exploration into factors influencing match-making in online dating systems [12]. Ranganath *et al.* explore how language used during speed dating sessions can be used to predict potential matches [24].

The combination of information retrieval and database approaches has been a fertile area of research recently. Such research is related to our work because of the structured nature of user profiles in match-making systems. Several previous studies have investigated the use of free text retrieval for a structured database [22, 16]. An important side effect of this is a ranking of records based on predicted relevance. Lavrenko *et al.* proposed structured relevance models for retrieving records from semi-structured data sets that have missing field information[18]. Follow-up work by Yi *et al.* used the structured relevance model for matching resumes to recruiter queries [26]. Several other methods have also been proposed for ranking database results [7, 6, 19] and for using explicit relevance feedback within database systems [21, 25, 3]. Our approach is unique in that we take

a feature-oriented, machine learning-based approach to the problem that provides a great deal of flexibility and a strong level of effectiveness.

Finally, the task of finding users (rather than documents) that satisfy an information need has been addressed in a variety of contexts. For example, in the TREC Enterprise Track, the Expert Search task requires participants to rank human experts with respect to specific topic [1]. Expert Search situations occur when a user seeks an expert. However, the expert has no constraint over who contacts her. One other difference is that users, both queriers and experts, usually do not have semistructured profiles. Completely decentralized search algorithms for individuals have also been studied in the context of social network analysis [17]. Furthermore, the reviewer assignment problem also involves a search over a set of users [14, 15]. While related, none of these tasks exhibit the unique properties exhibited by the online dating match-making task.

3. PROBLEM DEFINITION

We consider a retrieval scenario consisting of a set of users, $\mathcal{U}$, each of whom maintains a self-description and a query. For a user $u \in \mathcal{U}$, a description, d_u, consists of a set of descriptive attributes. These attributes may be scalar, categorical, or free text. In this paper we consider the attributes presented in Table 1. A query, q_u consists of a set of constraints on candidate attribute values. In this paper, we consider binary and scalar preferences. Binary constraints indicate that a certain attribute be present in a candidate record. Scalar constraints indicate that a scalar attribute be a certain value or in a certain range.

For each user, u, we would like to rank all other users, $v \in \mathcal{U} - \{u\}$, such that relevant matches occur above non-relevant candidates.

4. RANKING FOR MATCH-MAKING SYS-TEMS

We adopt a machine learning approach to learning our ranking model [20]. Machine learned ranking models consist of three parts: ranking features, relevance, and a ranking function. Ranking features include all signals we observe which may influence the scoring of a candidate match. In our work, relevance refers to two users being an appropriate match (e.g. they want to meet). Finally, the ranking function is a model of relevance given observable ranking features. In this section, we be explain in some detail each of these components as they relate to the match-making task.

4.1 Ranking Features

Given a user's query, we are interested in ranking all other users in decreasing order of two-way relevance. We consider three types of features that we believe are predictive of relevance. All of the features that we consider are based on the match-making attributes listed in Table 1.

First, we extract a number of features from user profiles. The user profiles represent information about a given user. Profiles typically only specify a single value for a given attribute rather than multiple values. This is due to the fact that users have a single age, a single body type, a single astrological sign, and so on. When ranking for a particular user, u, the profile features of a candidate, v, can be thought of as independent of user u and her query. We represent a

scalar	categorical					
age	body type	education	hair color	marital status	religious activity	star sign
height	city	employment	humor style	new user	romantic style	state
income	country	ethnicity	interests	occupation	sexuality	subscription status
num children	desires more children	eye color	languages	personality type	smoking	television viewer
num photos	drinking	featured profile	living situation	political bent	social style	zip code
		gender		religion		

Table 1: Match-making Attributes. Users define a set of scalar and categorical attributes when building a profile. In addition, users write a textual description of who they are and what they are looking for. User queries define their ideal match in terms of constraints on scalar and categorical attributes.

candidate's profile features with the notation $\overleftarrow{d}$ (330 features), a querier's profile features with $\overrightarrow{d}$ (330 features), and the concatenation of both sets as $\overleftrightarrow{d}$ (660 features).

Our second set of features compares pairs of user profiles. We expect that match relevance will be influenced when some profile features are very different (e.g. age). We compare two profiles in a match by comparing the individual profile features, $\overrightarrow{d}_i$ and $\overleftarrow{d}_i$, as,

$$\delta_i = |\overrightarrow{d}_i - \overleftarrow{d}_i| \qquad \text{scalar features}$$
$$\delta_i = \overrightarrow{d}_i \oplus \overleftarrow{d}_i \qquad \text{binary features}$$

We also implemented a simple score comparing the similarity of pairs of users' text self-description. For each user u, we compute an ℓ_2-normalized tf.idf-based term vector, t_u, based on the self-description. Given a pair of users, the text comparison is,

$$\delta_{\text{text}} = \langle t_u, t_v \rangle$$

Notice that, unlike our non-text feature comparison, the text comparison is a similarity as opposed to a difference. This is a minor issue since our modeling should be able to support both flavors of features. We present a pair's comparison features with δ (331 features).

The final set of features represent attribute matches with respect to a user's query. These are the attributes that the querier desires a potential match to have. For the scalar attributes, users can specify a range of values of interest, such as age between 18 and 35. These ranges are transformed into two scalar features, one representing the minimum allowable value and the other representing the maximum value. In our age example, this would correspond to the features age_min = 18 and age_max = 35. With categorical attributes, users specify one or more desirable values for each attribute. These preferences are encoded as binary features, one for each possible attribute value. For example, if a user is interested in matches with red or blonde hair, then the features hair_red and hair_blonde would be set to true (1) and all other hair color features (e.g., hair_black) would be set to false (0). Finally, users can specify the importance of each attribute. The possible options are "must match", "nice to match", and "any match". Here, "must match" attributes are those that the querier requires to be satisfied for a match to be relevant, "nice to match" are those attributes that the user would like to matches, but does not require, and "any match" means the user would be satisfied with any match for the given attribute. These attribute importances are encoded as binary features (e.g., hair_must_match, hair_nice_to_match, and hair_any_match). Our set of match features represent how well each attribute matches between a user's query and a candidate profile *as well as* the attribute preferences of the querier. For exam-
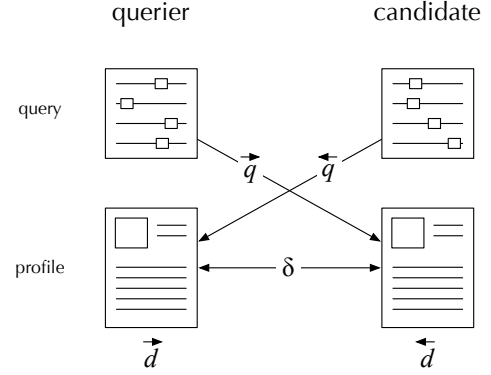
Figure 1: Graphical representation of the feature sets we consider in our experiments.

ple, if a querier is interested in matches with a given hair color, a match feature would encode whether a candidate's profile attribute satisfied that hair color *as well as* how important that attribute match was. Match features of this form are extracted for all query attributes. We represent a candidate's match features with respect to a querier's query with the notation $\overleftarrow{q}$ (156 features), a querier's match features with respect to the candidate's query with $\overrightarrow{q}$ (156 features), and the concatenation of both sets as $\overleftrightarrow{q}$ (312 features).

Figure 1 provides a graphical depiction of the symbols used to represent the different feature sets.

4.2 Relevance

Relevance in classic text retrieval tasks (e.g. TREC *ad hoc* retrieval tasks) is often interpreted as topical relevance. When interpreted this way, some degree of editorial assessment of document relevance can be performed. In a match-making system, queries are often constrained to a small set of structured attributes. Therefore, asking an editor to detect two-sided relevance can be a more time-consuming task; the editor would need to determine the intent of a querier given a structured query as well as a potentially rich profile in both directions. Furthermore, in many cases, the relevance of matches in a match-making system is more subjective than topical relevance. For example, there may be many subtle attributes in the profile or query which are important to the users but difficult to detect, even with query attribute preferences. Because of this, accurately determining intent and relevance may be impossible.

In order to address the difficulty with editorial relevance assessment, a retrieval system can use behavioral information of a running system to detect when users have found rel-

evant information. This approach is practiced in web search when relevance engineers monitor user clickthrough patterns [5, 23, 2]. Because salient behavioral information normally occurs after a query is issued, we refer to these behavioral signals as *post-presentation* signals. We are interested in tuning ranking function parameters using queries with post-presentation signals and generalizing to queries with no post-presentation signals.

Match-making systems provide a unique set of post-presentation relevance signals which can be used both for training and evaluation. For example, if two users exchange phone numbers, they are probably a good match. On the other hand, if one user's message never receives a reply, then they were probably a poor match. We present a complete list of our post-presentation features in Table 2. Instead of committing to a single feature to define relevance, we engineer a set of high precision rules to define relevance and non-relevance for a subset of matches. In our work, we consider as relevant any matches where users exchanged contact information; we consider as non-relevant any matches where at least one user inspected the other's profile but did not send a message as well as any matches where one user sent a message but the other did not reply. We refer to matches satisfying these rules as 'labeled' matches.

We often have post-presentation features likely to correlate with relevance but whose exact relationship we are unsure of. In this situation, because we have a small set of (automatically) labeled matches, we can train a model to predict the labels of these unlabeled matches. Specifically, we train a logistic regression model using the labeled set and the unlabeled features in Table 2 (i.e. we only use those features not used to infer relevance). We then use this model to label the unlabeled set with predicted relevance. For a match of users u and v, we use the notation $P(\mathcal{R}|u,v)$ to refer to the predicted relevance. We force labeled users to have $P(\mathcal{R}|u,v) \in \{0,1\}$ based on the automatic labeling.

We want to be clear that our predicted relevance is a noisy signal. For example, a successful interaction will not be detected if messages are exchanged through an external protocol. Furthermore, even when messages are exchanged within a system, it may be that the match is inappropriate. Messaging is also subject to false negatives if contacts are not initiated due a searcher's perceived probability of response. Nevertheless, we believe that basing relevance for a dating system on behavioral information is both more reliable and more efficient than editorial labeling.

4.3 Ranking Function

There are many different ways to define a ranking function for the match-making task. In this work, we make use of a machine learned ranking function based on gradient boosted decision trees (GBDTs) [10]. We chose to use GBDTs for several reasons. First, GBDTs can handle both numerical and categorical features, which is a good fit for the types of attributes found in match-making systems. Second, GBDT-based ranking functions can be trained using a wide variety of relevance sources, including manual labels and click-through data. Finally, these models have been shown to be highly effective for learning ranking functions [27].

We now provide a basic overview of GBDTs. Given a querier u and a candidate match v, we use GBDTs to compute a relevance score for the pair. As the name implies, GBDTs are boosted regression trees. Let $f_{u,v}$ be the fea-

phone_exchange	users both exchanged phone numbers
email_exchange	users both exchanged email addresses
regexp_match	users both exchanged information about meeting
num_exchanges	number of exchanges between users
message orphan	one user sent a message without a reply
message_disparity	difference in number of messages exchanged between users
exchange_timespan	duration of the message exchanges between users
message_density	the density of exchanges between the first and last messages exchanged
skip	one user saw the other's profile and did not send a message
num_view_exchanges	number profile views exchanged between users
view orphan	one user viewed another's profile and was not also viewed
view_disparity	difference in number of views exchanged between users
view_timespan	duration of views between users
view_density	the density of views between the first and last messages exchanged

Table 2: Post-Presentation Features. These features measure the interactions between users after they have seen a profile. Labeled features are bolded. Regular expressions used to detect meeting were taken from [12].

ture vector associated with the pair (u,v). A regression tree defines a function $T(u,v)$ by partitioning the space of feature values into disjoint regions R_j, $j = 1, 2, \ldots, J$, which are associated with the terminal nodes of the tree. Each region is assigned a value ϕ_j such that $T(u,v) = \phi_j$ if $f_{u,v} \in R_j$. Thus, the output of the regression tree is computed as:

$$T(u,v;\Theta) = \sum_{j=1}^{J} \phi_j I(f_{u,v} \in R_j), \quad (1)$$

where $\Theta = \{R_j, \phi_j\}_1^J$, are parameters and I is the indicator function. Given a pair of users, a single regression tree will return a single real-valued score for that pair of users. Precisely how the score is computed depends on the model parameters R_j and ϕ_j.

For a given loss function L these model parameters are estimated by minimizing the the total loss:

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{f_{u,v} \in R_j} L(y_{u,v}, \phi_j). \quad (2)$$

where $y_{u,v} = P(\mathcal{R}|u,v)$ is the actual or predicted relevance label for pair (u,v). Numerous heuristics exist for solving this optimization problem, the details of which are beyond the scope of this paper. In all of our experiments, we use mean squared error as our loss.

A boosted regression tree is an aggregate of such trees, each of which is computed in a sequence of stages. That is,

$$s_M(u,v) = \sum_{m=1}^{M} T(u,v;\Theta_m), \quad (3)$$

where at each stage m, Θ_m is estimated to fit the residuals

from stage $m - 1$ as follows:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{(u,v)} L(y_{u,v}, s_{m-1}(u,v) + \rho\phi_{j_m}). \qquad (4)$$

where ρ is a free parameter known as the shrinkage rate. Another important free parameter is the depth of the individual regression trees T. If the depth of the trees is greater than 1, then interactions between features are taken into account.

Thus, given a training set consisting of pairs of users and their associated relevance labels, we can learn a GBDT model model $\{\Theta_i\}_{i=1}^M$, which is an ensemble of regression trees. At test time, we use the learned model to score, and subsequently rank, candidate matches v with respect to queries u using $s_M(u,v)$.

Finally, the GBDT algorithm also provides what is called feature importance [10]. The importance is computed by keeping track of the reduction in the loss function at each feature variable split and then computing the total reduction of loss function along each feature variable. The feature importance can be useful for analyzing which features contribute the most to the learned model.

5. METHODS AND MATERIALS

5.1 Data

We collected the profiles and queries for users of a personals system during the fall of 2009. We collected user interaction data during this time period. After processing, we had 6,420,563 matches with features from Table 2 defined. We bootstrapped from labeled features to label unlabeled matches. We kept only users who had at least one relevant match with $P(\mathcal{R}) > 0$. After this processing, our data set consisted of 33,233 users (18,457 men and 14,776 women). We split our data into training, testing, and validation sets *by user*. This prevents us from evaluating on features observed in the training set. However, as a result, some matches cannot be observed because users are in different splits. Our data sets consisted of a training (7716 users, 17538 matches), validation (3697 users, 7543 matches), and testing set (4836 users, 10636 matches). Users whose only matches were broke by partitioning were removed from the experiments.

Text comparison features used idf statistics from the collection of user self-descriptions and the set of SMART stopwords.

5.2 Evaluation

We can compute standard information retrieval metrics, macro-averaged over users. That is, we iterate through all users, considering each user a 'querier' and all other users with labels or predictions 'candidates' to be ranked. Because we use both relevance predictions, we evaluate using metrics supporting estimates of the probability of relevance. In particular, we use the probabilistic versions of the following metrics: average precision (AP), precision at k documents (P@k), normalized discounted cumulative gain at rank k (NDCGk), and expected reciprocal rank (ERR). These are defined by,

$$AP_u = \frac{1}{\sum_v P(\mathcal{R}|u,v)} \sum_v \text{Prec}(v)P(\mathcal{R}|u,v)$$

$$P@k = \frac{1}{k} \sum_{v \in R_k} P(\mathcal{R}|u,v)$$

$$NDCGk = \sum_{i=1}^k \frac{2^{P(\mathcal{R}|u,v_i)} - 1}{\log(i+1)}$$

$$ERR = \sum_i \frac{P(\mathcal{R}|u,v_i)}{r} \prod_{j=1}^{i-1} (1 - P(\mathcal{R}|u,v_j))$$

The expected reciprocal rank metric was recently proposed by Chapelle et al. [4] and is designed for tasks with probabilistic relevance grades. The metric overcomes some of the position bias issues associated NDCG by incorporating a simple cascade-style user browsing model [8].

We define two sets of evaluation users as,

$$\mathcal{U}_1^{\mathcal{R}} = \{u \in \mathcal{U} : \exists v \in \mathcal{U}, P(\mathcal{R}|u,v) = 1\}$$

$$\mathcal{U}_0^{\mathcal{R}} = \{u \in \mathcal{U} : \exists v \in \mathcal{U}, P(\mathcal{R}|u,v) > 0\}$$

$\mathcal{U}_1^{\mathcal{R}}$ would only include those users who have at least one labeled relevant match. On the other hand, $\mathcal{U}_0^{\mathcal{R}}$ would only include those users who have at least one match with non-zero predicted relevance. Because the reliability and robustness of rank metrics is influenced by the number of candidates being ranked, we only evaluate over users with five or more matches with $P(\mathcal{R}|u,v)$ defined.

We will present results under two evaluation regimes: labeled and predicted. For the labeled evaluation, we use the set of users in $\mathcal{U}_1^{\mathcal{R}}$ as queriers and only the matches labeled by our rules (i.e. $P(\mathcal{R}|u,v) \in \{0,1\}$). In this case, our metrics reduce to their standard, non-probabilistic form. For the predicted evaluation, we use the set of users in $\mathcal{U}_0^{\mathcal{R}}$ as queries and all matches with $P(\mathcal{R}|u,v)$ defined.

To identify statistically significant differences between two ranking functions with respect to a given retrieval metric we use a paired, one-tailed non-parametric bootstrap test [9]. We adopt this significance test because it allows us to sample users non-uniformly. We sample a user u by $\max_v P(\mathcal{R}|u,v)$. For $\mathcal{U}_1^{\mathcal{R}}$, this results in uniform sampling. For $\mathcal{U}_1^{\mathcal{R}}$, this results in sampling proportional to the most relevant match. All significance tests are reported at the $p < 0.05$ level.

5.3 Training

We estimated GBDT models using different subsets and combinations of features regressing against $P(\mathcal{R}|u,v)$. These runs are labeled $\overrightarrow{q}$ (match only), $\overleftarrow{d}$ (candidate profile only), δ (profile similarity only), and $\overleftrightarrow{q}$ (two-way match only). We also experiment with a runs which combine sets of features. Because of unbalanced class distributions, we weighed instances according to,

$$w_{u,v} = \begin{cases} \frac{1}{|\mathcal{U}_{P(\mathcal{R}|u,v)=1}|} & P(\mathcal{R}|u,v) = 1 \\ \frac{1}{|\mathcal{U}_{P(\mathcal{R}|u,v)=0}|} & P(\mathcal{R}|u,v) = 0 \\ \frac{1}{|\mathcal{U}_{0<P(\mathcal{R}|u,v)<1}|} & 0 < P(\mathcal{R}|u,v) < 1 \end{cases}$$

so that we give a match weight according the source of its label.

Gradient-boosted decision trees have several free parameters: number of trees, number of nodes, and shrinkage. We

trained our decision trees using the training partition and selected free model parameters using the validation set, exploring ranges of free parameter values,

number of nodes	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50\}$
number of trees	$\{5, 10, 50, 75, 100, 250, 500, 1000\}$
shrinkage	$\{0.01, 0.025, 0.05, 0.075, 0.10, 0.15, 0.25, 0.50\}$

Although we train on the full set of matches, we validate using the pool depth restriction we use in evaluation.

6. RESULTS

The results of our experimental evaluation are shown in Table 3. We can make several observations about our runs from these results. First, and perhaps most strikingly, we see that using $\overleftarrow{d}$ features alone tends to perform very well across metrics. Recall that the $\overleftrightarrow{d}$ features are query-independent, and thus we are essentially learning nothing more than a static ranking of the profiles. In fact, there are no statistical differences between the strongest runs and $\overleftarrow{d}$.

Second, features comparing both users, δ, although effective, do not yield significant improvements over the query independent ranking (and actually results in a statistically significant drop for P@5 under both evaluation sets).

Using only match features, both one-way and two-way, results in the worst performance across metrics. These drops in performance are statistically significant across metrics compared to the next best run, δ, except P@5 and P@10 for the labeled relevance set and P@5 for the predicted relevance set. When we compare $\overrightarrow{q}$ and $\overleftarrow{q}$ runs, we observe statistically significant gains for four of the eight metrics for the labeled relevance set (AP, ERR, NDCG5, NDCG10) and four of the eight metrics for the predicted relevance set (AP, ERR, NDCG5, NDCG10).

We note that, when combining δ and $\overleftrightarrow{q}$ features, we see improvements compared to only using δ or $\overleftrightarrow{q}$. Compared to using only δ features, statistically significant improvements are isolated to one metric for labeled relevance set (P@5) and three metrics for the predicted relevance set (NDCG5, NDCG10, P@10). All improvements are statistically significant compared to using the $\overleftrightarrow{q}$ alone.

Finally, combining $\overleftrightarrow{d}$ and $\overleftrightarrow{q}$ provides strong performance, although not as strong as our other combination ($\delta \overleftrightarrow{q}$). Furthermore, this run degrades performance compared to using only $\overleftarrow{d}$ features, although these degradations are never statistically significant.

In Table 4, we show the top fifteen features, as measure by their feature importance as described in Section 4.3, for each run we evaluated. As the results indicate, the most important features across runs tend to be those associated with distance, age, height, the profile having been featured, and the user's subscription status. When considering match features ($\overrightarrow{q}$, $\overleftarrow{q}$), we notice the selection of both match values as well as constraints, suggesting that the algorithm is indeed taking advantage of the users' preferences, not just the match value. When we consider two-way match features, match features in both directions are selected, indicating that both users' match preferences are important for ranking. Finally, we note that the text similarity measure, when available, often ranks highly in terms of feature importance, suggesting that non-relational attributes also play an important role.

7. DISCUSSION

We were surprised that that model trained only using $\overleftarrow{d}$ features resulted in such strong performance. Does this imply that attributes specific to the querier are irrelevant? Most likely not. There could be several other reasons for this. For example, the expressiveness of the query representation may be limited. Even though users are provided with the ability to indicate the importance of specific traits and their values, the importance value is still discrete; these importances are unlikely to be as expressive as real-valued weights applied in the machine learned model. Although it may seem that the attributes in Table 1 are not informative enough to rank candidates, this hypothesis is unlikely to be supported since the $\overleftarrow{d}$ features are constrained to the same representation. Finally, it might be that the user is just not agile enough with the query interface to produce effective queries. This suggests that alternative interfaces or more intuitive features might be better at eliciting effective queries from the users for this particular task.

The strength of the $\overleftarrow{d}$ features may also be the result of our data gathering process. We only experiment with pairs of users who have interaction features defined, in the form of profile views or message exchanges. Users may become cognizant of how their own attributes compare to the preferences associated with the profiles they view. This is especially true since profile displays include information about the user's query. There is likely a self-censoring process involved whereby a user only sends a message if they notice that they match the profile's preferences.

The conjunctions of features noticeably did not provide statistically significant improvements for many metrics. This result likely follows from the number of features considered in the aggregated set compared to the training set size. For example, The combination of $\overleftrightarrow{d}$ and $\overleftrightarrow{q}$ features results in a vector of 1614 features. With only 17,538 training instances, it is unlikely that the model had enough data to learn an accurate predictor, thereby degrading ranking effectiveness. We believe that, with more training data, our combined runs would significantly outperform the simpler baselines.

In addition to physical attributes related to age and height, those attributes with the strongest predictive power were distance, subscription status, and whether a user's profile had been featured. While the distance constraint is understandable to many (e.g. individuals are looking for matches within their city or state), the others require some explanation. The subscription status of a user indicates an investment in the search process. If a user has paid money for extra site features, they may be more engaged in the system and willing to look for matches. On the other hand, when the site decides to feature a profile, this raises the credibility of the user, making them more attractive to other users. As a result, we expect these users to be more successful at finding matches.

The importance of the text similarity feature when in conjunction with the match features, suggests that there is information present in the text description that can be exploited for ranking. Text provides user an unconstrained field to discuss arbitrary interests. The ability to detect similarity in these interests using a very simple text similarity measure means that eliciting text from users and exploiting it for ranking is a good avenue for future work.

	labeled relevance						predicted relevance					
	$\overleftarrow{d}$	δ	$\overrightarrow{q}$	$\overleftrightarrow{q}$	$\delta\overleftrightarrow{q}$	$\overleftrightarrow{d}\overleftrightarrow{q}$	$\overleftarrow{d}$	δ	$\overrightarrow{q}$	$\overleftrightarrow{q}$	$\delta\overleftrightarrow{q}$	$\overleftrightarrow{d}\overleftrightarrow{q}$
AP	0.453	0.439	0.368	0.398	0.456	0.445	0.485	0.484	0.428	0.454	0.497	0.494
NDCG1	0.248	0.269	0.186	0.198	0.271	0.267	0.346	0.366	0.287	0.317	0.380	0.367
NDCG5	0.513	0.478	0.409	0.437	0.505	0.500	0.576	0.556	0.501	0.527	0.575	0.580
NDCG10	0.573	0.555	0.497	0.520	0.571	0.565	0.649	0.643	0.598	0.619	0.659	0.656
P@1	0.248	0.269	0.186	0.198	0.271	0.267	0.360	0.380	0.304	0.334	0.395	0.381
P@5	0.207	0.188	0.176	0.178	0.200	0.201	0.326	0.311	0.298	0.303	0.318	0.326
P@10	0.129	0.127	0.124	0.123	0.129	0.129	0.226	0.223	0.221	0.219	0.227	0.226
ERR	0.481	0.464	0.392	0.420	0.482	0.475	0.582	0.577	0.517	0.552	0.595	0.589

Table 3: **Results for match-making ranking using metrics defined with the high precision subset of labels as well as predicted relevance. Subsets of features include query-independent profile ranking ($\overleftarrow{d}$), profile similarity (δ), one-way match ($\overrightarrow{q}$), and two-way match ($\overleftrightarrow{q}$). Statistical significance between pairs of runs for a metric are described in the text.**

$\overleftarrow{d}$	δ	$\overrightarrow{q}$	$\overleftrightarrow{q}$	$\delta\overleftrightarrow{q}$	$\overleftrightarrow{d}\overleftrightarrow{q}$
featured	subscription status	distance	distance ($\overrightarrow{q}$)	text (δ)	distance ($\overrightarrow{q}$)
age	distance	max age	distance ($\overleftrightarrow{q}$)	distance ($\overleftrightarrow{q}$)	distance ($\overleftrightarrow{q}$)
height	height	max height*	max age ($\overleftrightarrow{q}$)	age (δ)	age ($\overleftarrow{d}$)
living arrangement	living arrangement	more kids	max age ($\overrightarrow{q}$)	distance ($\overrightarrow{q}$)	age ($\overleftarrow{d}$)
subscription status	featured	min height$^\circ$	max age$^\circ$ ($\overrightarrow{q}$)	subscription status (δ)	max age ($\overrightarrow{q}$)
religious activity	religious activity	max height$^\circ$	ethnicity ($\overleftrightarrow{q}$)	height (δ)	max age ($\overleftrightarrow{q}$)
employment	gender	ethnicity	max height* ($\overleftrightarrow{q}$)	featured (δ)	featured ($\overleftarrow{d}$)
num photos	eye color	ethnicity$^\circ$	min height ($\overrightarrow{q}$)	new user (δ)	featured ($\overrightarrow{d}$)
religion	humor	languages*	ethnicity* ($\overrightarrow{q}$)	gender (δ)	subscription status ($\overleftarrow{d}$)
interests	activity	has kids	body type ($\overrightarrow{q}$)	num photos (δ)	subscription status ($\overrightarrow{d}$)
new user	text	body type	max height$^\circ$ ($\overrightarrow{q}$)	eye color (δ)	height ($\overleftarrow{d}$)
smoking	new user	max age$^\circ$	max height$^\circ$ ($\overleftrightarrow{q}$)	max age ($\overrightarrow{q}$)	height ($\overrightarrow{d}$)
activity		education$^+$	max height* ($\overrightarrow{q}$)	religious activity (δ)	max height*
more kids		education$^\circ$	more kids ($\overrightarrow{q}$)	max age ($\overleftrightarrow{q}$)	new user ($\overleftarrow{d}$)
occupation		zip	body type$^+$ ($\overleftrightarrow{q}$)	featured (δ)	max height* ($\overleftrightarrow{q}$)

Table 4: **Top ten features for each of our feature combinations. Superscripts indicate query constraint features ($\circ$: 'any'; $+$: 'nice to have'; $\star$: 'must have'). Symbols in parentheses indicate the source of the feature. In all runs except δ, more than fifteen features were selected by the algorithm.**

8. CONCLUSIONS

We have presented the problem of ranking for match-making systems. We motivated this problem by its ubiquity as well as its compelling differences with many standard retrieval scenarios. Specifically, we believe that the notions of two-way relevance, its detection, and usefulness for ranking are all very different from many search problems.

We found that, for our problem setting, query-independent ranking provided a simple and effective method for ranking candidate matches. This greatly simplifies system design since ranking can be done globally, with some simple filtering for important match constraints (e.g. distance, age). At the same time, we believe that users' queries, as they are often revealed to all users in a system may provide a self-censoring which in turn results in a higher quality list of candidates. Furthermore, we believe that such systems may benefit from different query interfaces more appropriate for the task. This might include the option of users to issue free text queries, as text similarity appeared to be a strong predictor of relevance when combined with match features.

There is a wide range of research problems associated with retrieval in match-making systems. In the area of relevance, future work includes discovering additional implicit relevance signals. More interestingly, we can design system modules which, as a side effect, detect relevance with high accuracy. In the area of features, future work includes the refinement of features we propose and development of new attributes found to be highly correlated with relevance. For example, new structured attributes might be detected by inspecting text description term matches which are correlated with relevance. Finally, in the area of ranking, it may be possible to develop improved loss functions that take into account the unique notion of two-way relevance.

9. ACKNOWLEDGMENTS

10. REFERENCES

[1] K. Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, June 2008.

[2] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS*, 2007.

[3] K. Chakrabarti, K. Porkaew, and S. Mehrotra. Efficient query refinement in multimedia databases. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, page 196. IEEE Computer Society, 2000.

[4] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM 2009*, 2009.

[5] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW 2009*, pages 1–10. ACM, 2009.

[6] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 888–899. VLDB Endowment, 2004.

[7] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst.*, 31(3):1134–1168, 2006.

[8] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 87–94. ACM, 2008.

[9] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[11] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, January 1962.

[12] G. Hitsch, A. Hortaçsu, and D. Ariely. What makes you click? an empirical analysis of online dating. 2005 Meeting Papers 207, Society for Economic Dynamics, 2005.

[13] R. Irving. An efficient algorithm for the stable roommates problem. *Journal of Algorithms*, 6:577–595, 1985.

[14] M. Karimzadehgan and C. Zhai. Constrained multi-aspect expertise matching for committee review assignment. In *CIKM 2009*, 2009.

[15] M. Karimzadehgan, C. Zhai, and G. Belford. Multi-aspect expertise matching for review assignment. In *CIKM 2008*, pages 1113–1122. ACM, 2008.

[16] J. Kim, X. Xue, and W. B. Croft. A probabilistic retrieval model for semistructured data. In *ECIR 2009*, pages 228–239. Springer-Verlag, 2009.

[17] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[18] V. Lavrenko, X. Yi, and J. Allan. Information retrieval on empty fields. In *HLT 2007*, pages 89–96. Association for Computational Linguistics, April 2007.

[19] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.

[20] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Now Publishers, 2009.

[21] M. Ortega-Binderberger, K. Chakrabarti, and S. Mehrotra. An approach to integrating query refinement in sql. In *EDBT '02: Proceedings of the 8th International Conference on Extending Database Technology*, pages 15–33. Springer-Verlag, 2002.

[22] D. Petkova, W. B. Croft, and Y. Diao. Refining keyword queries for xml retrieval by combining content and structure. In *ECIR 2009*, pages 662–669. Springer-Verlag, 2009.

[23] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM 2008*, pages 43–52. ACM, 2008.

[24] R. Ranganath, D. Jurafsky, and D. McFarland. It's not you, it's me: Detecting flirting and its misperception in speed-dates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 334–342. Association for Computational Linguistics, August 2009.

[25] L. Wu, C. Faloutsos, K. P. Sycara, and T. R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 297–306. Morgan Kaufmann Publishers Inc., 2000.

[26] X. Yi, J. Allan, and W. B. Croft. Matching resumes and jobs based on relevance models. In *SIGIR 2007*, pages 809–810. ACM, 2007.

[27] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, 2008.