Social search with missing data: which ranking algorithm?

ZHU, J., EISENSTADT, M., GONÇALVES, A., DENHAM, C., UREN, V. and SONG, D.

2007

Social Search With Missing Data: Which Ranking Algorithm?

Jianhan Zhu[a], Marc Eisenstadt[a], Alexandre Gonçalves[b], Chris Denham[a], Victoria Uren[a], Dawei Song[a]

[a]Knowledge Media Institute and Computing Research Consortium, The Open University, United Kingdom

{j.zhu, m.eisenstadt, c.m.denham, v.s.uren, d.song}@open.ac.uk

[b]Stela Institute, Brazil

a.l.goncalves@stela.org.br

ABSTRACT

Online social networking tools are extremely popular, but can miss potential discoveries latent in the social 'fabric'. Matchmaking services which can do naive profile matching with old database technology are too brittle in the absence of key data, and even modern ontological markup, though powerful, can be onerous at data-input time. In this paper, we present a system called BuddyFinder which can automatically identify buddies who can best match a user's search requirements specified in a term-based query, even in the absence of stored user-profiles. We deploy and compare five statistical measures, namely, our own CORDER, mutual information (MI), phi-squared, improved MI and Z score, and two TF/IDF based baseline methods to find online users who best match the search requirements based on 'inferred profiles' of these users in the form of scavenged web pages. These measures identify statistically significant relationships between online users and a term-based query. Our user evaluation on two groups of users shows that BuddyFinder can find users highly relevant to search queries, and that CORDER achieved the best average ranking correlations among all seven algorithms and improved the performance of both baseline methods.

Keywords

Social software, Ranking algorithms, Relation discovery, Instant messaging.

1. Introduction

Online social networking tools and services, in the form of friendship networks, instant messaging and chatting tools, dating services and business partner tools are extremely popular on the Web today – an extensive and evolving survey/taxonomy of Social Networking Services is provided online in [Meskill]. Such tools and services have evolved from early-adopter 'leisure' tools to mission-critical business collaboration tools, and have attracted increasing attention from the research community [Nardi *et al.* 2002; Alani *et al.* 2003; Isaacs *et al.* 2002; Setlock *et al.* 2004]. Today's online social networking tools typically involve a large number of users who coalesce into a community of mixed backgrounds and preferences, exhibiting varying degrees of overlapping interests and social cohesion. An online user typically has a number of contacts or buddies in his/her interest groups, with the amount of overlap (shared interests) dropping off dramatically as degrees of separation increase: groups may include work colleagues, family members, friends, conference acquaintances, friends-of-friends, recommended contacts, deliberately sought-out contacts, and of course random or even unwanted contacts. Current social networking tools allow users to manage additions to and deletions from their buddy lists manually, although most allow import from standard productivity-tool address books and preferences to block unwanted contacts.

In this era of powerful search engines, consider the problem of seeking mission-critical or immediate advice characterized by the question 'Who is available who can help me deal with an urgent problem *now*?' The Knowledge Management mantra of 'the right knowledge in the right place at the right time' rings somewhat hollow if you need to find key people quickly but cannot. Mixed hi-tech and low-tech solutions (e.g. using Google or a social networking service as a first pass, then emailing or phoning around) may miss out on key availability information, which is one of the great strengths of instant messaging (IM) tools. Indeed, in this context it is not the messaging and chat capabilities of IM that pay the biggest dividends, but rather the *presence* information that (when correct at least) shows who is available [Chakraborty 2001]. Modern social networking services such as Tribe (http://www.tribe.net), and indeed many discussion forums, typically include a 'presence indicator icon' so users can see at a glance who is available. Other tools, such as our own BuddySpace[1] [Vogiazou *et al.* 2005], also include the option of overlaying presence indicators on top of custom maps to deal with those cases where location is either important or simply 'feels good' to the user.

How, then, can we address the problem of finding the right person *now*? This problem falls at the overlap of two seemingly contradictory niches:

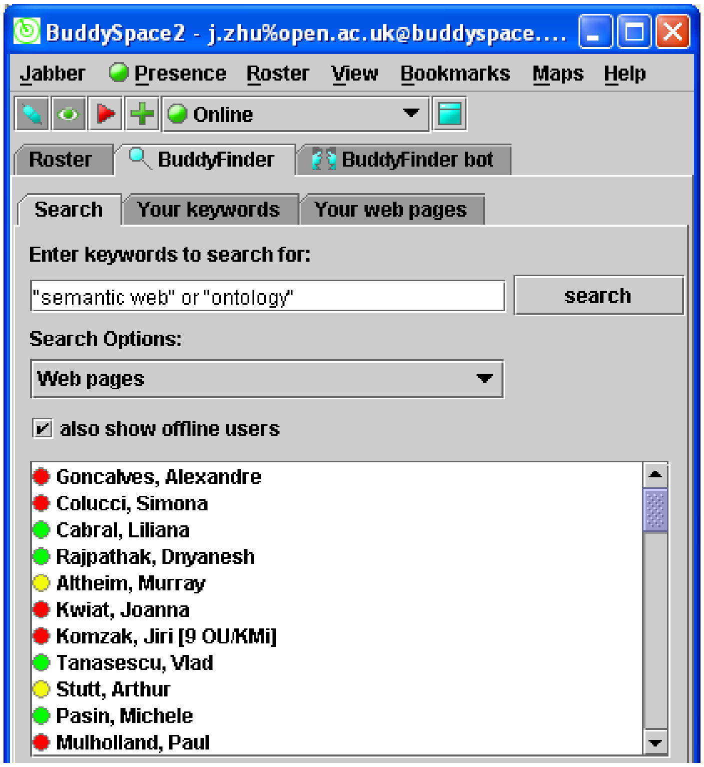
- *Good availability, but no need for search:* this is the 'IM niche' which is great for seeing who is available now, but since a user's IM buddy list is populated by people they already know, it may seem rather artificial to have to search through such people for matches
- *Web-centric search, but poor availability info*: the generic problem of searching the web for 'the right person' is well known, but generally we cannot count on availability information being provided.

We address this contradiction by noting a key exception to the assumption that 'a user's IM buddy list is populated by people they already know'. In particular, enterprise-wide IM and any service that provides automatic buddy-list population ('automatic roster or contact list generation'), has the potential for auto-enrolling users into large groups, such as student cohort groups and large just-in-time project teams, where hierarchically-grouped buddy lists can grow just beyond a 'familiarity horizon': in other words, a user does *not* know a lot about everyone on the buddy list, yet it contains, within acceptable privacy limits, availability information for those who may have the knowledge to help solve a user's problem. A typical case is our own ELeGI project (http://www.elegi.org/) [Allison *et al.* 2003], a 23-

[1] The BuddySpace and BuddyFinder project and downloading homepage is at: http://buddyspace.sourceforge.net/

partner European consortium with several hundred individuals involved. All are automatically subscribed to the IM roster, yet not everyone generally knows much about everyone else.

We believe that such usage scenarios will be increasingly typical of future knowledge workers and e-learners. In these cases, it is clear that simply listing online users in alphabetical order (http://www.msn.com, http://messenger.yahoo.com) and performing a naïve match (http://www.icq.com) cannot help users find those with the right knowledge accurately and efficiently.



Enter keywords to search for:

"semantic web" or "ontology"

Search Options:

Web pages

☑ also show offline users

🔴 Goncalves, Alexandre
🔴 Colucci, Simona
🟢 Cabral, Liliana
🟢 Rajpathak, Dnyanesh
🟡 Altheim, Murray
🔴 Kwiat, Joanna
🔴 Komzak, Jiri [9 OU/KMi]
🟢 Tanasescu, Vlad
🟡 Stutt, Arthur
🟢 Pasin, Michele
🔴 Mulholland, Paul

Fig. 1. BuddyFinder output for a search on "semantic web" OR "ontology". Different colours (green, red, and yellow) depict different presence states such as online, offline, away.

In this paper, we propose a novel text mining solution to the "who knows what" problem. By taking into account the online users' homepages and other pertinent web pages, our approach is aimed at identifying those who are highly relevant to a searcher's information need specified as a term-based query. Fig. 1 illustrates a system called BuddyFinder, which implements the proposed CORDER (COmmunity Relation Discovery by named Entity Recognition) algorithm [Zhu et al. 2005], as well as a number of other existing statistical approaches, in the framework of BuddySpace [Eisenstadt *et al.*

2003]. In order to capture a richer semantics involved in the user's information need, we propose to use Boolean queries[2]. The choice of using Boolean model does not exclude us from using other relevance models such as BM25 etc, which can be adapted for integration in our approach. Our initial experiments show that Boolean model results in better buddy search results than BM25. In Fig. 1, a BuddyFinder user has input a query, "semantic web" OR "ontology", and obtained a list of related users ranked by CORDER. He/she can then choose to interact with some or all of these users by chatting, emails, or viewing their profiles, etc. Our task-based evaluation indicates that the recommendations made by CORDER correlate with human judgments and outperform existing approaches.

The rest of the paper is organized as follows. Section 2 presents related work in database and semantic web based approaches for buddy search, followed by a review of the existing information retrieval and statistical approaches for deriving term relationships. Section 3 details the CORDER algorithm, which, along with four existing approaches, is employed in our BuddyFinder system. A method for integrating Boolean operators in the proposed statistical similarity measures is also introduced in the same section. The Buddyfinder has been evaluated by two sets of user experiments. The detailed evaluation methodology and experimental results are reported in Section 4. Finally, in Section 5, we conclude the paper and highlight the future work.

2. Approaches to Buddy Search

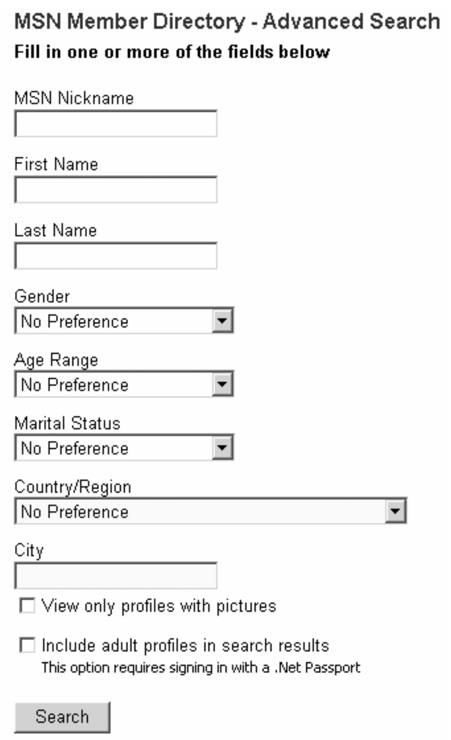
There are two main streams of research on identifying people who match a set of preferences, what we call buddy search. These are approaches which use formal knowledge about individuals stored in

[2] Although Boolean search has not been popular in the general web search, it is widely used in many domain specific search engines, such as the PubMed (http://www.pubmedcentral.nih.gov/), and the Intranet search, the reasoning being that users in specific domains tend to know better about what they are going to search and therefore they tend to accept the use of Boolean queries which allow them to better specify deeper semantics of their information needs.

databases, ontologies etc. and explicit linkage patterns, and approaches which exploit other resources such as text documents using information retrieval (IR) and statistical methods.

2.1 Database and Semantic Web Approaches

Much of the existing work for buddy search uses database profile matching tools. These store profiles of users gathered in a registration process. They can return accurate search results provided that the profiles are accurate and complete. This is the classic approach of dating services, and can be very effective in a live instant messaging context too. In Fig. 2, we can see how MSN member search (http://members.msn.com) enables users to search for other users by their MSN nicknames, last names, first names, genders etc.



MSN Member Directory - Advanced Search
Fill in one or more of the fields below

MSN Nickname

First Name

Last Name

Gender
No Preference

Age Range
No Preference

Marital Status
No Preference

Country/Region
No Preference

City

☐ View only profiles with pictures

☐ Include adult profiles in search results
This option requires signing in with a .Net Passport

Fig. 2. MSN advanced search.

A more recent semantic web inspired approach is to ask online users to write FOAF (Friend of a friend) files to describe themselves (http://www.foaf-project.org). The structured information in these FOAF files can then be searched. However, both profile data and FOAF files generally need to be explicitly and voluntarily provided by the users. Overall, we note four shortcomings of the database and FOAF-profile approaches:

- 1. The credibility of the data may be questionable, i.e., users may not have provided true or complete information about themselves, sometimes in a deliberate attempt to abuse the system.
- 2. Asking users to provide the data creates unacceptable overheads.
- 3. It is hard to maintain the currency of the data, i.e., users may not update the data to reflect their ongoing activities.
- 4. The search is limited to the information specified in the profile or FOAF file, thus it is impossible to discover something new or unknown.

Our work is also related to Communities of Practice (CoP). A community of practice refers to a group of professionals formally or informally sharing knowledge and information for performing some common tasks or exploring common problems of their professional activities and interests [Lave and Wenger 1991]. Existing research in deriving CoPs has been focused on inferring and visualizing community structure based on explicit linkage patterns, e.g., hyperlinks, contact lists, sender-receiver of emails, and domain-specific ontologies. As an example and one of the more exciting tools to come out of the semantic web community in recent years is Flink [Mika], which deploys a plethora of sources and services, including Google's API, FOAF profiles, archived email messages and publications, to 'triangulate' in order to infer relations and ranking in a powerful visual browsing paradigm. Even so, it does not yet offer the ability to answer the problem we posed in Section 1, namely finding the right person with the right knowledge at the right time, although it provides very valuable background information to assist a power user in navigating the space of possibilities.

Another semantic web approach exploits labeled relations between entities (such as people, projects, and organizations) in organizational ontologies. ONTOCOPI (Ontology-based Community of Practice

Identifier) [Alani *et al.* 2003] attempts to uncover a latent community of practice (COP) by applying a set of ontology-based network analysis techniques that examine the connectivity of instances in the knowledge base with respect to the type, density, and weight of these connections. These COPs can be used for buddy search within an organization. Although effective, this approach also has two shortcomings of profile based approaches.

- 1. It is generally hard to maintain the currency of the ontology.
- 2. The search is limited to knowledge in the ontology.

2.2 IR and Statistical Approaches

The motivation for IR and statistical approaches is the significant amount of information about Web users in their personal homepages and blogs, their friends or colleagues' homepages, and the websites of their work place – all potentially relevant to their personal and/or working life. The information can provide fertile ground for a social networking tool to search for buddies who match a user's preferences, *even in the absence of a person's specific profile information,* and *even in the absence of a person's own web pages*. Our belief is that we can leverage social networks ('whom you know') for opportunistic discovery that can potentially deliver the right knowledge in the right place at the right time, by deploying a mixture of buddy list, presence, and text mining methods. Such a hybrid approach can overcome the shortcomings of the current approaches.

- 1. Since web pages about users are from various sources, the search is less susceptible to abuse. The influence of false information contained in a small number of "bad" pages to the final result can be largely alleviated by the statistical measures applied to the whole dataset. Furthermore, we can judge the credibility of the data source when deciding whether to include pages in the search process. This can be done either automatically or manually. First, we can train some spam filtering algorithm to filter web pages on a continual basis. Second, we can remove web pages from certain websites which are not trustworthy, which involves very little manual judgments.

- 2. There are fewer overheads in collecting the data. Users do not need to fill in a form or maintain a FOAF file. At most they may be asked for a list URLs of pages about themselves. However even this can be avoided by having the system make an intelligent guess of some URLs based on a web search (see Section 3.1).
- 3. We can get a more complete profile of a user and are not limited to information in a registration form or a FOAF file. Therefore a greater variety of searches can be successfully answered by the system.

In our buddy search scenario, assume for simplicity that each user has a profile consisting of a list of web pages. An online user can specify a query specifying his/her preferences that consists of a list of terms and Boolean relations between the terms. The query is matched against the web pages in the profiles of users. Buddies who are returned are seen as matching the preferences of the user. For example, a query *"Java AND C++"* will return users whose profiles contain web pages matching both the term *"Java"* and the term *"C++"*. These users are assumed to have interests in Java and C++.

In contrast to the above approach of returning people relevant to a search query, information retrieval (IR) systems are mainly *document-oriented*. Given a user's term-based query, current search engines such as Google present the user with a list of web pages in the order of their relevance to the query. From the user's perspective, however, they are generally more interested in the information contained in the pages than the pages themselves.

For a typical query such as *"Java AND C++"*, a buddy search system may return a number of matching buddies, who are of different levels of relevance to the query. By giving a relevance score to each of these buddies, we can present a user with a list of buddies ranked by their relevance scores. In the ranked list of buddies, buddies more relevant to the query are on the top and buddies who are probably false positives are at the bottom and can be ignored. We assume that a buddy is more relevant to a term in a query when the buddy's name has more co-occurrences with the term in web pages. Therefore, we can simply rank buddies by taking into account co-occurrences between each buddy and terms in the query. Boolean model based query processing can be easily integrated with the co-occurrence model. However, other relevance models such BM25 can also be adapted for integration with the co-occurrence model. Naturally, false positives will occur, but as long as the high-ranking results are 'good enough', the payoffs can still be high.

Conrad and Utt [1994] presented a concept-oriented rather than document-oriented Association System to find associations between well-defined target information, e.g., people and organizations from documents. Their Association System is used to answer queries related to the target information and to support ways of accessing information impossible with standard IR systems. They used a named entity recognition system to extract features, e.g., people and organizations' names, from text. They argued that a direct association occurs between two features when they occur 'close' to each other in the text. Closeness is measured by either the distance between the two features or their linguistic context, e.g., in the same sentence, paragraph, or document. Experiments in [Croft *et al.* 1991] show that word distance is the strongest evidence for phrasal relationships. Conrad and Utt [1994] used a *window* to define direct associations between features, and two statistical measures, namely, mutual information (MI) and phi-squared, to identify significant associations between features.

In computational linguistics, MI and phi-squared [Church and Hanks 1989] [Church 1991] have been used to measure the level of associations between words using their co-occurrence information in a corpus. Vechtomova et al. [2003] studied long-span collocates for query expansion. Given a target word, its collocates are other words significantly co-occurring with it in same sentences, paragraph, or documents. They improved two statistical measures, namely, MI and Z score, by taking into account window size, and used them to measure the degree of associations between collocates.

In this paper, we present a novel approach, called CORDER, which is used in our BuddyFinder application to derive the associations between people and query terms based on co-occurring information. The aforementioned existing statistical methods are also implemented in BuddyFinder for ease of comparison with the CORDER algorithm.

Related work also includes the area of discovering social networks in P2P (Peer to Peer) systems. Peers sharing the same interests, as defined by their document collections, are clustered and can be ranked by their importance to a search query. Jin *et al.* [2006] used Latent Semantic Indexing (LSI) to reveal semantic subspaces of feature spaces from documents stored on peers, and support vector machine (SVM) to classify the peers into different categories based on the vectors extracted using LSI. We can apply CORDER as one of the algorithms to P2P search and compare with the others' approaches.

3. BuddyFinder

In buddy search, a query consists of terms and Boolean relations between them. Given the query as the target, we find buddies who have strong associations with the query, deploying the overall architecture described in Section 3.1. We calculate the association between a buddy and the query in three steps. First, we use a hierarchical clustering algorithm to align variants of the same buddy's name (Section 3.2). Second, given each term in the query as the target, we calculate the association between the buddy and the term (Section 3.3). Third, we combine associations between the buddy and terms in a search query by taking into account the Boolean relations (Section 3.4).

3.1 Overall System Architecture

BuddyFinder relies on an instant messaging platform Jabber [Adams 2001] for exchanging information. In Fig. 3, Jabber [Adams 2001] uses an XML based protocol called XMPP for all transactions between Jabber users and the Jabber server in **B**. When a Jabber user in **A** issues a term-based query to search for buddies, the query is sent from the user's Jabber client using chat messages following the XMPP protocol. The BuddyFinder query is routed by the server to the BuddyFinder Chatbot in **C**, which interacts with various modules to get a ranked list of buddies. The BuddyFinder Chatbot is a standard Jabber client, an architecture which allows a lot of flexibility in terms of where it is hosted and allows BuddyFinder users to interact with it using any Jabber based instant messaging client software. BuddyFinder Chatbot will reply with the results of the query following the XMPP protocol.

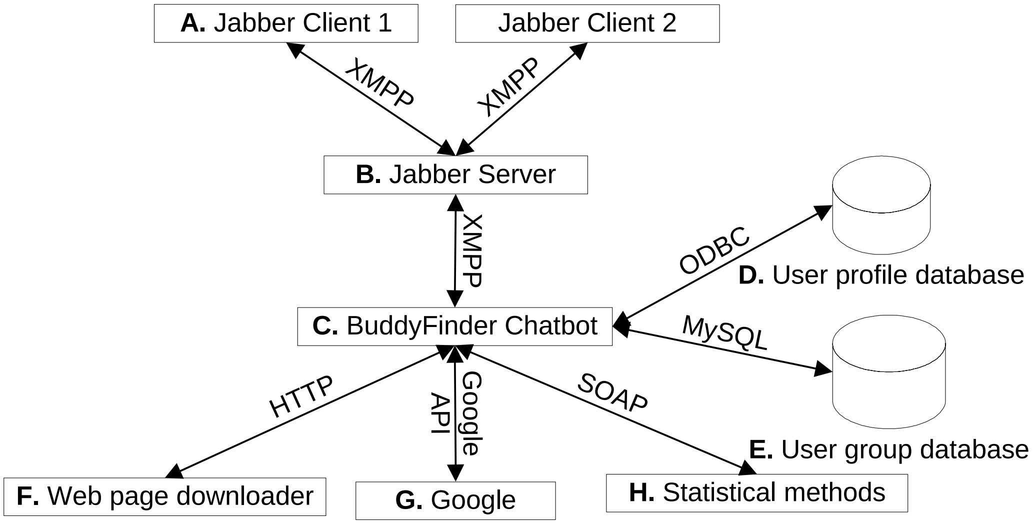


Fig. 3. BuddyFinder Architecture.

The user profile database in **D** stores each user's profile consisting of a list of web page URLs, which can either be supplied by the user or guessed by BuddyFinder as described below. The BuddyFinder Chatbot accesses user profile information via an ODBC connection. The initial list of URLs for a user is generated automatically using a Web search component via the Google API. To give an easy start for a user to specify these URLs, the Web search component makes an "intelligent guess" in finding web pages relevant to him/her. The intelligent guess is based on the domain shown in his/her email address and his/her name, which are stored in the user group database in **E**. For example, if one user's email address is m.eisenstadt@open.ac.uk and his name is "Marc Eisenstadt", we guess the domain of the user is open.ac.uk. We send the query "Marc Eisenstadt site:open.ac.uk" to Google and use URLs of the top 10 web page as the default profile for the user.

To improve search performance, web pages in users' profiles are downloaded in **F** and cached for search. Each user is subscribed to various IM user groups, e.g., KMi user group and Open University user group. Given a user, a list of buddies who belong to his/her user groups is generated by issuing SQL queries on the profile database in **E**. When a user sends a query to search, BuddyFinder performs the search on the profiles of the list of buddies providing valuable 'triangulation' results to compensate for missing data. Thus different users can get different search results even if they had submitted the same query.

The buddy search process in Fig. 3 is as follows. A user in **A** sends a search query to Jabber server in **B**, which forwards the query to BuddyFinder Chatbot in **C**. The Chatbot interprets the query for a list of terms and Boolean relations between them. The Chatbot queries the user group database in **E** for a list of buddies who belong to the same groups as the current user. BuddyFinder Chatbot queries the user profile database in **D** for web pages in the profiles of these buddies.

Given a buddy, we use a set of web pages for ranking him/her against the query. Both pages from the buddy's profile and pages from profiles of other buddies who are in the same groups as his/hers are included in ranking him/her. We remove HTML markup and scripts from these pages to get plain texts, and plain texts are segmented into words. Given the buddy, for each term in the query, we process the set of pages for word offsets of the buddy's name and the term. In the statistical methods module, **H**, which is implemented as a web service, we choose an algorithm and a window size to calculate the association between the buddy and the query based on word offsets of the buddy's name and terms in the query, and Boolean relations between these terms. Buddies with strong associations with the query are sent back to the Chatbot to rank them. For example, suppose we choose the phi-squared measure (described subsequently) and window size 100 to calculate the association between people in KMi and a query “BuddySpace” AND “Messaging”: in that case, “Marc Eisenstadt” and “Jiri Komzak” are found to have strong associations with the query and sent back as search results.

But there are many ways to compute the strength of association, and this is the ‘inner component’ of box H that we explore in depth, comparing our own algorithm against other contenders, in the rest of the paper.

3.2 Buddy Name Alignment

Variants of a buddy’s name, e.g., full name, first name, full name with the title, may exist on pages of the dataset. We use a hierarchical clustering algorithm proposed by us [Zhu et al. 2005] to align the variants. Two names, *N1* and *N2*, judged similar by their string similarity *StrSim(N1,N2)* are more likely to be variants of the same name if they occur on the same page or two pages which link to each other (we use the Levenshtein edit distance but other metrics are also suitable). The two names may appear on two different pages, and we define the contextual distance *ConDis(N1,N2)* as the minimum number of links, regardless of link direction, between the two pages where these two names occur. The contextual distance is zero if the two names occur on the same page. We define the similarity between the two names as *Sim(N1,N2)*= $\frac{StrSim(N1,N2)}{1+a\times ConDis(N1,N2)}$, where *a* is weight to adjust the influence of contextual distance. As we move away from a page, traversing several out-links, we may introduce some noise, so if we decrease the value of *a*, the effect of such noise can be alleviated. On the other hand, if we increase *a*, we can find variants of a buddy’s name on more remotely linked pages.

3.3 Computation of Associations

We extend current statistical approaches to buddy search. In current approaches, relations between features as words or named entities are considered, while in our buddy search, we consider associations between a named entity, i.e., a buddy's name, and a search query. We propose to take into account Boolean relations between terms in the query in associations. We need to find out whether these statistical measures can be extended to identify buddies truly relevant to the query. We have used five statistical measures in our buddy search, namely, MI, phi-squared, Vechtomova et al's MI, Vechtomova et al's Z score, and our own CORDER algorithm [Zhu *et al.* 2005]. Next, we present MI and phi-squared in Section 3.3.1, Vechtomova et al's MI and Z Score in Section 3.3.2, and the CORDER algorithm in Section 3.3.3.

3.3.1 Mutual Information and Phi-Squared

Given a buddy's name and a term in a query, a direct association occurs when they occur 'close' to each other in the text. We use a window to define their direct association. A window refers to the number of words on either side of a target feature. The window size is chosen to roughly approximate average paragraph-level or document-level associations. The strength of association between the two features depends on the number of co-occurrences in a window and how common they are in the whole dataset. For example, a single co-occurrence of "semantic web" and "Jianhan Zhu" in a window is not likely to be significant, if the two features occurred frequently in the whole dataset.

Given the target as a term, *T*, the MI between *T* and a buddy, *B*, is:

$$MI(T,B) = \log_2 \frac{P(T,B)}{P(T)P(B)}$$

where *P(T,B)* is the probability that *T* and *B* occur together; *P(T)* and *P(B)* are the probabilities that *T* and *B* occur individually.

The MI measure compares the probability of *T* and *B* occurring together to the probability of them occurring independently. When there is a strong relationship between the two features, the joint probability *P(T,B)* will be greater than chance and *MI(T,B)* will be greater than 0. If their co-occurrences are mainly due to chance, their MI score will be close to zero. If they occur mainly individually, their MI score will be a negative number.

We often use a contingency table to calculate both MI and phi-squared, ϕ^2 . The table is represented as follows.

	B	$\overline{B}$
T	a	b
$\overline{T}$	c	d

Where cell [a] records the number of co-occurrences of *T* and *B* in a window, cell [b] records the number of times *T* occurs but *B* does not, cell [c] records the number of times *B* occurs but *T* does not, and cell [d] records the number of times neither of them occurs.

Given the target as *T*, the MI between *T* and *B* is:

$$MI(T,B) = \log_2 \frac{a(a+b+c+d)}{(a+b)(a+c)}$$

The ϕ^2 measure between *T* and *B* is:

$$\phi^2 = \frac{(ad - bc)^2}{(a+b)(a+c)(b+d)(b+c)}, \text{ where } 0 \le \phi^2 \le 1.$$

MI typically favors low-frequency features. Conrad and Utt [1994] suggested to use ϕ^2 instead of MI, since ϕ^2 tends to favor high-frequency features more. Their experiments with different window sizes for the two methods show that ϕ^2 generated better associations than MI.

3.3.2 Vechtomova et al.'s Mutual Information and Z Score

Vechtomova *et al.* [2003] has improved MI by taking into account windows of various sizes. Ideally left and right sides of a window have equal lengths, *W*. However, the actual window sizes can be smaller for two reasons. First, a window around the target feature *T* may be truncated if it hits the boundary of the current document. Second, the window may be truncated when another occurrence of feature *T*, *T1*, is inside the window around *T*. We need to avoid duplicates of the same co-occurred feature, *B*, of *T* and *T1* when their distance is less than *W*. If *T1* occurs after the target *T*, the window around *T* is truncated at the position of *T1*; if *T1* occurs before the target *T*, the left half of the window around *T* is ignored altogether.

Vechtomova et al.'s improved MI (IMI)'s most important difference from the standard MI is the asymmetry of their method due to their use of average window sizes. Standard MI is symmetrical, i.e., *MI*(*T*,*B*) = *MI*(*B*,*T*), since the joint probabilities of *T* and *B* are symmetrical, i.e., *P*(*T*,*B*) = *P*(*B*,*T*). In calculating IMI, the window sizes around instances of a target *T* may be smaller than the ideal window size of (*W* + *W*). Instead, the average of all window sizes around feature *T* as v_T is used to estimate the probability of occurrences of *B* in these windows around *T* as $P_T(T,B)$. However, if we had used *B* as the target and considered the occurrences of *T* in the windows around *B*, we would have a different average window size (v_B instead of v_T). Generally v_B and v_T are different. Given the target *T*, the IMI measure between two features *T* and *B* is:

$$IMI(T,B) = \log_2 \frac{P_v(T,B)}{P(T)P(B)} = \log_2 \frac{\dfrac{f(T,B)}{Sv_T}}{\dfrac{f(T)f(B)}{S^2}}$$

where *f(T,B)* is the joint frequency of *T* and *B* in the dataset, *f(T)* and *f(B)* are frequencies of independent occurrences of *T* and *B* in the dataset respectively, v_T is the average window size around *T* in the dataset, and *S* is the dataset size.

Due to the limitation of MI which favors low-frequency features, we use *Z* score [Vechtomova *et al.* 2003] to measure the distance in standard deviations between the actual frequency of co-occurrences of *B* and *T*, and the expected frequency of their co-occurrences. We take as the null hypothesis that the occurrence of *T* does not predict the presence or absence of *B* in these windows, i.e., the likelihood that *B* occurs by chance in these windows is the same as the likelihood that it occurs by chance in any other location in the whole dataset. For a pair of low-frequency features which co-occur by chance, we may misleadingly get a high MI score, while their *Z* score will not be high due to large variances in their probabilities.

Given the target *T*, the total number of locations where *B* and *T* may co-occur is $v_T f(T)$. Under the null hypothesis, the probability that each of these locations contains *B* is *f*(*B*)/*S*. Thus the expected number of co-occurrences of *B* and *T*, *E,* is the mean of a binomial distribution, v_T *f*(*T*) *f*(*B*)/*S*. The actual observed co-occurrences of *B* and *T*, *O,* is *f*(*T*,*B*). Therefore we define the *Z* score as:

$$Z(T,B)=\frac{O-E}{\sqrt{E}}=\frac{f(T,B)-\frac{v_T f(T) f(B)}{S}}{\sqrt{\frac{v_T f(T) f(B)}{S}}}$$

Vechtomova *et al.* [2003] set the threshold of significance of association between two features measured by *Z* score as no less than 1.65 standard deviations.

3.3.3 CORDER Algorithm

Our own CORDER algorithm [Zhu *et al.* 2005] discovers relations by identifying co-occurrences of named entities (NEs) in text. The algorithm is based on the intuition that if an individual has expertise in an area his/her name will be associated with key terms about that area in many documents. Similarly if two individuals often work together we expect to see their names associated. In general, we assume that NEs that are closely related to each other tend to appear together more often and closer. The CORDER measure takes into account three aspects as follows.

Co-occurrence. Two features are considered to co-occur if they appear in the same text fragment, which can be a document or a text window. Generally, if a feature is closely related to the target, they tend to co-occur more often. For *T* and *B*, we compute a relative frequency [Resnik 1999] of co-occurrences of them as

$$\hat{p}(T,B) = Num(T,B)/N$$

where *Num*(*T*,*B*) is the number of text fragments in which *T* and *B* co-occur, and *N* is the total number of text fragments.

Distance. Two features closely related to each other tend to occur close to each other. If *T* and *B*, both occur only once in a text fragment, the distance between them is the number of words between them. If *T* occurs once and *B* occurs multiple times in the text fragment, the distance between *T* and *B* is the number of words between *T* and its nearest occurrence of *B*. When both *T* and *B* occur multiple times, we average the distance from each occurrence of *T* to *B* and define the logarithm distance between *T* and *B* in the *i*th text fragment as

$$\overline{d}_i(T,B) = \frac{\sum_j (1+\log_2(\min(T_j,B)))}{Freq_i(T)}$$

where $Freq_i(T)$ is the number of occurrences of *T* in the *i*th text fragment and $min(T_j,B)$ is the distance between the *j*th occurrence of *T*, T_j, and *B*.

Frequency: A feature is considered to be more important if it has more occurrences in a text fragment. Consequently, a numerous feature tends to have strong relations with other features which also occur in that text fragment.

Given a target, *T*, we calculate the relation strength between *T* and another feature, *B*, by taking into account their co-occurrences, distance and frequency in co-occurred text fragments. The relation strength, *R(T, B)*, between *T* and *B* is defined as follows.

$$R(T,B) = \hat{p}(T,B) \times \sum_{i} \left(\frac{f(Freq_i(T)) \times f(Freq_i(B))}{\bar{d}_i(T,B)} \right)$$

where $f(Freq_i(T)=1+log_2(Freq_i(T))$, $f(Freq_i(B)=1+log_2(Freq_i(B))$, and $Freq_i(T)$ and $Freq_i(B)$ are the numbers of occurrences of *T* and *B* in the *i*th text fragment respectively.

In our previous work, we used the CORDER algorithm to exploit named entity recognition and co-occurrence data to associate individuals in an organization with their expertise and associates. We found that CORDER can find other NEs which are strongly related to a target. CORDER's rankings of related NEs were close to users' own rankings and better than rankings produced by a Google based co-occurrence method [Zhu *et al.* 2005].

3.4 Boolean Query Processing

It appears that there is a user population consisting of *N* users, $U = \{u_1, u_2, \ldots, u_N\}$ and users may form groups $g_j = \{u_{j1}, u_{j2}, \ldots, u_{jk}\}$ with each g_j being a subset of *U*, i.e. $g_j \subseteq U$. *G* is a collection of such user groups, $G = \{g_1, g_2, \ldots, g_n\}$. In g_j, the web pages in the profile of one user, *u*, often mention not only him/herself but also other buddies as his/her friends or colleagues in the same group, e.g., co-authors of papers and members of the same projects. Similarly, web pages from profiles of these other users also contain information about *u*. Furthermore, *u* may belong to more than one group in *G*. Given *u*, web pages from both his/her profile and profiles of other buddies in the same groups as his/hers are all taken into account in associating *u* with a term-based query. This is precisely how we 'triangulate' on relations, and pick up information even when *u* fails to mention anything about him/herself.

A searcher *s* can specify his/her preferences in a term-based query, *Q*, to search other users in the same groups as *s*. The query *Q* consists of *L* terms, $\{T_k\}$, $1 \le k \le L$, and Boolean relations between them. These terms can be either single words or phrases. Given the query *Q* as the target, we use one of the five measures, namely, MI, phi-squared, improved MI and Z score, and CORDER, to calculate the association between *Q* and each of the users, *u*.

We consider three Boolean relations, namely, AND, OR, and NOT. For two terms *T1* and *T2*, we consider AND and OR relations between them. The AND operator is used to specify that both *T1* and *T2* must evaluate to TRUE for "*T1 AND T2*" to evaluate to TRUE. Given a user, *u*, in order for both *T1* and *T2* to evaluate to TRUE, *u* needs to co-occur with *T1* and *T2* in a window or a document. If *u* co-occurs with both *T1* and *T2* in a window, "*T1 AND T2*" evaluates to TRUE. If *u* co-occurs with *T1* in one document /window and co-occurs with *T2* in another document /window, "*T1* AND *T2*" still evaluates to TRUE. We use one of the five measures to get two association measures between *u* and *T1* and *T2* as *A(T1,u)* and *A(T2,u)* respectively. We define the association between "*T1* AND *T2*" and the user as *A(T1 AND T2, u)=A(T1,u)*×*A(T2,u)*.

The OR operator is used to specify that either *T1* or *T2* must evaluate to TRUE for "*T1* OR *T2*" to evaluate to TRUE. In order for either *T1* or *T2* to evaluate to TRUE, *u* needs to co-occur with either *T1* or *T2*. We define the association between "*T1* OR *T2*" and the user as *A(T1 OR T2, u)=A(T1,u)*+ *A(T2,u)*.

For one term *T1*, the NOT operator is used to specify that *T1* must evaluate to FALSE for "NOT *T1*" to evaluate to TRUE. We define the association between "NOT *T1*" and the user as *A(NOT T1, u)* =0 if *A(T1, u)*> 0 and =1 if *A(T1, u)*= 0.

Given a query *Q*, we get the association, *A(Q,u)*, between *u* and *Q*, by combining *A(Tk, u)* using the Boolean relations between them. Example 1, for a query *Q1* = (*T1 AND T2) AND (NOT T3), A(Q1,u) = A(T1,u)×A(T2,u) × A(NOT T3,u)*. Example 2, for a query *Q2* = (*T1 OR T2) AND (NOT T3), A(Q2,u) = A(T1,u)× A(NOT T3,u)+ A(T2,u) × A(NOT T3,u).*

Finally, we rank these users according to their associations with *Q* and send the results back to the searcher.

4. Evaluation

During our evaluation, we found that there is very little existing useful information in ontologies or other forms of structured data such as FOAF files or expertise databases to answer users' search queries. The major disadvantage of structured data in buddy search is their low coverage (ontologies and databases only cover some aspects of a domain and are often not up to date, and only a few users have FOAF files and these FOAF files are also not up to date). Due to the lack of available structured data which can be used in our experiments, a comparison with structured data based approaches is not applicable. Therefore, our empirical evaluation is focused on statistical approaches. The aim of the experiments is to test the effects of finer-grained statistical association discovery approaches proposed in the paper.

We introduce two TF/IDF (term frequency/inverse document frequency) based baseline algorithms and compare our five ranking algorithms with them. In our first baseline algorithm (B1), a buddy's association to a term is measured by aggregating the TF/IDF values of the term for all the documents in the buddy's profile where the query term appears. The association strength $A_{B1}(T,B)$ is defined as follows.

$$A_{B1}(T,B) = \sum_{D \in P_B} tf_{T,D} \cdot \log_2 \frac{N}{n}$$

Where P_B is the buddy's profile, $tf_{T,D}$ is the frequency of term T in document D, N is the total number of documents in the dataset, and n is the number of documents where T appears.

In our second baseline algorithm (B2), a buddy's association to a term is measured by the relation strength obtained in B1 plus the aggregated value of the TF/IDF scores of the term for all the documents in other buddies' profiles where the term and the buddy's name co-occur in these documents. The association strength $A_{B2}(T,B)$ is defined as:

$$A_{B2}(T,B) = \sum_{D \in P_B} tf_{T,D} \cdot \log_2 \frac{N}{n} + \sum_{D \notin P_B, CoOccur_D(B,T)} tf_{T,D} \cdot \log_2 \frac{N}{n}$$

We have used the query processing method described in Section 3.4 in applying the two baseline algorithms for buddy search.

To establish whether the seven algorithms were fit for purpose, we designed a strategy to get a group of users' collective opinions on who were most relevant to a given topic within their group. In a large organization, everybody knows about the expertise and associations of some of his/her colleagues, and is not sure of, or does not know about the others, but their collective knowledge may tell us almost all the experts pertinent to a topic. In our evaluation, we also told the evaluators that they can spend time getting to know more about an 'expert' by reading documents about him/her, chatting with him/her, or emailing him/her, before they can judge his/her relevance to a query. Given a topic, e.g., "semantic web", a group of people may have various opinions about who is strongly associated with the topic in their organization. We assume that, just like a democratic voting or referendum system, we can identify the 'experts' from the collective opinions expressed by the ratings given by a group of people. Therefore, we use their collective opinions as a *standard* to evaluate the five ranking algorithms and the two baseline algorithms.

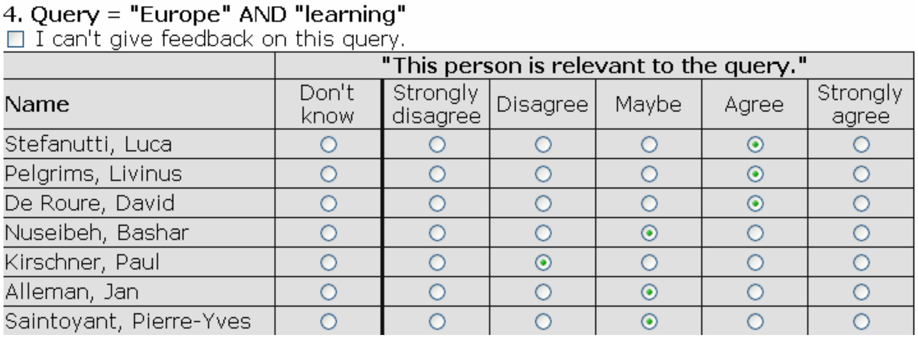
4.1 Experimental Set-up

We have evaluated our system on two groups of users, namely, 70 people in the Knowledge Media Institute (http://kmi.open.ac.uk), and 142 people in the ELeGI project from 23 organizations and 9 European countries. There are 5 people who are in both groups, i.e., work in KMi and the ELeGI project. For the KMi and ELeGI groups, there are in total 1,011 and 1,978 web pages from people's profiles, respectively. After removing tags and scripts, and word segmentation, for the KMi and ELeGI datasets, there are in total 1,217,689 and 2,394,704 words respectively.

Two members of the KMi and ELeGI groups have selected 17 and 19 queries[3] for user evaluation on the two groups, respectively, as listed in the left most columns of Table 1 and Table 2. For each query, we have applied seven algorithms, namely, CORDER, MI, phi-squared, improved MI, Z score, baseline one, and baseline 2 to the dataset respectively. The window size is set as 150 (the effect of different window sizes in a wide range of 100 to 300 does not have much impact on these algorithms' performances, and the details will be discussed in Section 4.3). The threshold for MI and phi-squared is set as greater than 0.0, and the threshold for Z score is set as above 1.65 standard deviations same as the one used by Vechtomova et al. [2003]. These are all statistically significant values and will remain the same in any experimental environment. A ranking algorithm can generate a long list of relevant buddies, however, users generally only look at the top ranked buddies to find their answers. We selected up to top 10 ranked buddies (can be less than 10 when there are less than 10 buddies in the list) returned by each algorithm for each query as the relevant buddy set of the algorithm. For a given query, the seven algorithms may not only rank the same buddies differently but also have different buddies in the relevant buddy set. We union the seven relevant buddy sets by the seven algorithms to get the overall relevant buddy set of the query. The total numbers of relevant buddies generated by the seven

[3] Normally, the user does not have to specify the relations between query terms, because we assume that there is an AND relations implicitly between them. Boolean connectives OR and NOT are not commonly used, so they have not appeared in our evaluated queries.

algorithms for the 17 queries in the KMi group and 19 queries in the ELeGI group are 335 and 230, respectively, so they give the average numbers of relevant buddies generated for each query in the KMi and ELeGI groups as 19.71 and 12.11, respectively.



4. Query = "Europe" AND "learning"

□ I can't give feedback on this query.

	"This person is relevant to the query."					
Name	Don't know	Strongly disagree	Disagree	Maybe	Agree	Strongly agree
Stefanutti, Luca	○	○	○	○	⊙	○
Pelgrims, Livinus	○	○	○	○	⊙	○
De Roure, David	○	○	○	○	⊙	○
Nuseibeh, Bashar	○	○	○	⊙	○	○
Kirschner, Paul	○	○	⊙	○	○	○
Alleman, Jan	○	○	○	⊙	○	○
Saintoyant, Pierre-Yves	○	○	○	⊙	○	○

Fig. 4. A query with a relevant buddy list generated by seven ranking algorithms in randomized order for user evaluation.

We created a web based form (Fig. 4) which allows human evaluators to assess the query results generated by seven algorithms. In order to minimize the influence of the order of the queries and buddies in the relevant buddy set of each query, every time an evaluator visits the form, the order of both the queries and buddies in the relevant buddy set of each query are randomly generated. The order in which each algorithm has ranked the list of buddies is hidden from the evaluators. The evaluators were asked to spend as much time as they need to determine a person’s expertise on a topic. They can choose to read documents about these experts found by searching them in Google, chat with these experts, or email them in order to know their level of expertise on a topic. Since each evaluator may not be able to evaluate all the buddies for all the queries, we give them the option to skip a query, or select “don’t know” for a buddy to a query. To evaluate the statement that a buddy is relevant to a query, they can choose from “strongly disagree”, “disagree”, “maybe”, “agree”, and “strongly agree”. The KMi form has been evaluated by 23 people from KMi group, representing a range of experience from PhD students, secretaries, project managers, lecturers, research fellows, and professors, and the ELeGI form has been evaluated by 17 people from ELeGI consortium, representing a range of experience from PhD students, project members, project leaders, lecturers, research fellows, and professors. Each relevant buddy in each query has been evaluated by at least three people. These gave us a *post hoc* standard against which to measure the seven algorithms' performance.

Evaluators may have different opinions on the association between each buddy in the relevant buddy set of each query and the query. In order to get a collective opinion from a group of evaluators, we assign following rating values to their opinions: -2: “strongly disagree”, -1: “disagree”, 0: “maybe”, 1: “agree”, 2: “strongly agree”, and “don't know” is ignored. We calculate the mean of the opinion values from a group of users as the group's rating value for a buddy. In order to remove dubious user ratings, we calculate the standard deviation of the rating values and remove any user rating value that is two times the standard deviation value away from the mean. After removing dubious ratings, we calculate the mean of a group of users' ratings again as the group's rating value for the buddy. In our evaluation, standard deviations for most of the evaluated experts are quite small confirming that majority of the users can reach a consensus of these experts' expertise on topics.

Given a query, we can use the group of users' rating value for each buddy to rank these buddies. Thus the new ranked list of buddies reflects the group of users' collective opinion in how the list of buddies should be ranked. We use the group's ranked list to evaluate how well each algorithm has ranked the buddies to reflect the group's views. We have used the Spearman's rank correlation [Gibbons 1976] (value between -1 and 1) to measure how well two ranked lists match each other.

$$RC_{Algorithm,Group} = 1 - \frac{6\sum_i (r_{i,Group} - r_{i,Algorithm})^2}{N^3 - N}$$

where $r_{i,Group}$ and $r_{i,Algorithm}$ ($1 \leq i, r_{i,Group}, r_{i,Algorithm} \leq N$) are the two rankings provided by the group of users and an algorithm respectively for the *i*th buddy in the list. *RC*=1 when the two sets of rankings are in perfect agreement and *RC*=-1 when they are in perfect disagreement. Seven algorithms may have found different buddies to compose the overall relevant buddy list and some buddies in the overall relevant buddy list may be missing from the top relevant buddy list of a particular algorithm. To make the ranked lists of equal length for Spearman’s comparison, if there are “missing” buddies from the top relevant buddy list of an algorithm, these buddies were appended to the algorithm’s list in the same relative order that they appeared in the full ranked relevant buddy list for the algorithm.

4.2 Results

The ranking correlation for each algorithm on each query in the KMi and ELeGI groups is shown in Table 1 and 2, respectively. “AKT”, “BuddySpace”, “Magpie”, and “ScholOnto” are project names. Generally, people would search for people closely related to research areas and projects. We have also experimented with a more variety of types of queries to emulate real users’ various needs in their searches, and show that BuddyFinder can give satisfying answers to various types of queries. In Table 2, people’s names, e.g., “Marc Eisenstadt”, and academic institution’s names, “University of Graz”, are used. In this case, we are looking for people who are closely related to a given person, e.g., colleagues, collaborators, and closely related to an institution, e.g., employees, collaborators.

4.3 Analysis

We can see from Table 1 that CORDER, MI, phi-squared, improved MI, Z score, B1, and B2 performed the best in 6, 1, 2, 0, 0, 4, and 4 queries respectively. We can see from Table 2 that CORDER, MI, phi-squared, improved MI, Z score, B1, and B2 performed the best in 4, 0, 3, 1, 3, 7, and 3 queries respectively. In Table 1, CORDER has the highest average ranking correlation, closely followed by B1, B2, and phi-squared. In Table 2, CORDER has the highest average ranking correlation, closely followed by B1. The two MI algorithms performed the worst. B2 does not improve on B1, and B2 has the largest ranking correlation variations among all seven algorithms. We think the reason is that B2 does not use a text window in judging the relevance between a term and a buddy, thus it is sometimes wrong to say that a buddy is relevant to a term simply because the buddy and the term co-occur in other buddies' profiles. While CORDER has shown improvements over B1 by taking into account text windows and distance between entities. Although B1 performed very well even comparable to CORDER, we have to remember that in our experiments almost all the buddies' profiles were well specified, while in real world situations, many users may not have their profiles well specified and search on them will be largely based on their friends' profiles.

Table 1. Spearman's rank correlation for KMi group, highest rank correlation values are in bold and underlined

Query/Rank Correlation (RC)	CORDE	MI	Phi-squared	Improved MI	Z Score	B1	B2
1. AKT	0.6842	0.2474	0.6910	-0.0008	0.5286	**0.7910**	0.6699
2. Artificial Intelligence	**0.9140**	-0.0351	0.8061	-0.1044	0.7781	0.4921	0.8965
3. Buddyspace AND Messaging	0.3654	0.2115	0.3324	0.1181	0.4066	0.4643	**0.7692**
4. Hypertext	0.6838	0.3320	0.6093	0.1993	0.7007	0.7064	**0.7674**
5. Information Extraction	0.7047	0.0735	**0.8137**	-0.1397	0.4730	0.6581	0.6973
6. Knowledge Management	**0.9200**	0.1863	0.8229	0.1244	0.7101	0.8937	0.9012
7. Knowledge Modelling	**0.9596**	-0.0114	0.7667	-0.2298	0.7702	0.7904	0.9491
8. Machine Learning	**0.5839**	0.0559	0.2378	0.0909	0.1329	0.3776	0.1853
9. Magpie	0.9045	0.1286	0.8271	-0.0594	0.9045	0.7602	**0.9474**
10. Natural Language Processing	-0.1545	**0.5636**	0.3182	0.5000	0.3455	-0.1500	-0.6500
11. Ontologies	**0.9466**	0.4628	0.9313	0.1086	0.7337	0.9280	0.9165
12. Planning AND Scheduling	0.7657	0.5000	0.6538	0.6189	0.6818	**0.9371**	0.3776
13. Question Answering	0.6171	0.0939	0.6089	0.1228	0.6264	**0.6429**	0.6316
14. ScholOnto	0.5723	0.6887	0.5037	0.3909	0.7230	**0.9069**	0.7770
15. Semantic Web	0.9688	0.1719	**0.9846**	0.3169	0.9285	0.8777	0.9735
16. Web Services	0.6054	0.4632	0.6569	0.3333	0.6397	0.6397	**0.7586**
17. Social Software	**0.7335**	0.2487	0.5552	0.2591	0.3361	0.7283	0.7035
RC variance	0.07682	0.04493	0.04582	0.05267	0.04587	0.07266	0.15728
RC average	**0.6927**	0.2577	0.6541	0.1558	0.6129	0.6732	0.6630

Table 2. Spearman's rank correlation for ELeGI group, highest rank correlation values are in bold and underlined

Query/Rank Correlation (RC)	CORDE	MI	Phi-squared	Improved MI	Z Score	B1	B2
1. Artificial Intelligence	**0.7821**	-0.3107	0.1607	-0.2286	0.1661	0.6071	0.7750
2. BuddySpace	0.6000	0.5429	0.5429	0.7143	0.7143	**0.7714**	0.4000
3.Europe AND Learning	0.5797	0.2967	0.4863	0.2637	0.5027	0.5330	**0.6813**
4. Hugh Davis	0.5000	0.1000	**0.8000**	-0.8000	0.4000	0.7000	0.2500

5. Human Learning	0.0909	0.2091	0.2818	**0.3727**	**0.3273**	0.0773	-0.2273
6. Hypermedia	0.5265	0.3000	0.4147	0.3368	0.5265	0.3618	**0.6221**
7. Hypertext	0.5385	-0.0412	0.3159	0.0742	0.4231	**0.5769**	0.5742
8. Instant Messaging	0.4643	0.3929	**0.5714**	0.3929	**0.5714**	-0.1607	-0.4821
9. Java AND C++	0.5515	0.1273	0.2485	0.1273	0.1636	**0.5818**	0.4727
10.Knowledge Representation	0.6905	0.3095	0.3095	0.3095	0.3095	**0.7381**	-0.5833
11.Learning AND Grid	**0.6835**	0.4923	0.6091	0.6099	0.6808	0.6214	0.6679
12. Machine Learning	**0.4286**	0.1429	0.1429	0.1786	0.1786	-0.1250	-0.1250
13. Marc Eisenstadt	0.7212	-0.1273	0.4303	-0.0061	0.5515	**0.7939**	0.4182
14. Semantic Web	0.2445	-0.1291	0.3104	-0.2335	0.2060	0.1374	**0.4533**
15. Social Software	**1.0000**	-1.0000	0.5000	-1.0000	0.5000	0.5000	0.7500
16.Software AND Grid	0.2451	0.1483	0.3358	0.3370	**0.3909**	0.0368	0.2500
17.University of Graz	0.9007	-0.1091	0.5784	-0.0686	0.5833	**0.9669**	0.8848
18. University of Southampton	0.8187	-0.5027	**0.8626**	-0.5687	0.7225	0.8297	0.6126
19. WSRF	-0.6000	-0.2571	0.0286	-0.6000	-0.5429	**0.4857**	-0.5714
RC variance	0.1246	0.1401	0.0465	0.2252	0.0834	0.1102	0.3065
RC average	**0.5140**	0.0308	0.4174	0.0111	0.3882	0.4754	0.3065

We have used a t-test to find out whether CORDER is significantly better than the other algorithms. Using the rank correlations for MI as a benchmark, we do a t-test to see the improvements by CORDER, phi-squared, improved MI, Z score, and the two baseline algorithms. We make the null hypothesis, H_0, that the tested algorithm is not significantly better than the benchmark, and the alternative hypothesis, H_1, that the tested algorithm is significantly better than the benchmark. We use the settings of Two-Sample Assuming Unequal Variances.

For Table 1, the one-tail critical values for significance levels α =0.05 and 0.01 (degree of freedom =32) are 1.6973 and 2.4487, respectively. T stat values for CORDER, phi-squared, improved MI, Z score, B1, and B2 are 5.1386, 5.4242, -1.3448, 4.8594, 4.9950, and 3.7161 respectively. There it turns out that CORDER, phi-squared, Z score, B1 and B2 have significant improvements over MI, while the improved MI does not. Using the rank correlations for phi-squared, B1, and B2 as a benchmark, respectively, we do a t-test to see whether CORDER is better than them. T stat value for CORDER is 0.4539, 0.2076, and 0.2524, respectively, so we cannot say CORDER is significantly better than any of them.

For Table 2, the one-tail critical values for significance levels α =0.05 and 0.01 (degree of freedom =36) are 1.6883 and 2.4378, respectively. Using the rank correlations for MI as a benchmark, T stat values for CORDER, phi-squared, improved MI, Z score, B1, and B2 are 4.0947, 3.9007, -0.1416, 3.2953, 3.8746, and 1.9936 respectively. There it turns out that CORDER, phi-squared, Z score, B1, and B2 have significant improvements over MI, while the improved MI does not. Using the rank correlations for phi-squared, B1, and B2 as a benchmark, respectively, T stat value for CORDER is 1.0186, 0.347, and 1.5339, respectively, so we cannot say CORDER is significantly better than any of them.

We noticed that all the seven algorithms performed better on the KMi than the ELeGI datasets in terms of average rank correlation due to the reason that the profiles of KMi users are of better quality than those of ELeGI users.[4] The percentages of decrease in average rank correlation of CORDER, MI, phi-squared, improved MI, Z score, B1, and B2 between the two datasets are 25.80%, 88.05%, 36.19%, 92.88%, 36.66%, 29.38%, and 53.77%, respectively. We can see that CORDER is the most robust despite the lower quality of user profiles, and clearly much more consistent than B2 and phi-squared etc.

Table 3. Total average ratings and variances for No.1 buddy for all queries, highest ratings and lowest variances are highlighted

Group/Ratings for No.1 for each query		Overall	CORDER	MI	Phi-squared	IMI	Z Score	B1	B2
KMi	**Average**	1.7422	**1.3488**	0.3041	1.2441	0.1422	1.0980	1.2610	1.2871
	Variance	0.7729	0.4295	1.5724	0.4694	0.9279	0.7302	**0.3225**	0.6795
ELeGI	**Average**	1.608	1.191	0.517	**1.271**	0.738	1.148	0.8079	1.2638
	Variance	0.1334	0.3258	0.6731	**0.3095**	0.5314	0.5087	0.9105	0.3609

We have examined the results to find out whether the top buddies recommended by the algorithms are consistently good. In Tables 1 and 2, we averaged the user ratings for the buddies ranked as No.1 for

[4] All KMi users have their official email addresses and profiles in the same organization easier for our automatic user profile population, and ELeGI users use different email addresses with some even non-official email addresses and their profile documents are distributed making it difficult for user profile population.

all 17 and 19 queries (KMi and ELeGI respectively) by each of the seven algorithms to get their average ratings (the range of the average ratings is -2 to 2) and variance of average ratings for the No.1 buddy. The results are shown in Table 3. We can see that CORDER achieved the highest average ratings for No.1 on the KMi queries, and Phi-squared achieved the highest on the ELeGI queries. The No.1 buddies recommended by CORDER, phi-squared, Z score, and B2 have all got the average evaluators' ratings above 1 ("agree"), suggesting that these top buddies are a safe bet to have strong associations with the query. CORDER and phi-squared consistently give high average ratings to No.1 buddy for each query, given that they have the lowest variances among the seven algorithms. B1 has high variance on the ELeGI dataset, while B2 has high variance on the KMi dataset.

Our experiments show that Vechtomova *et al.* [2003]'s improved MI and Z score for query expansion do not make significant improvements over original MI and phi-squared, respectively, instead, performed slightly worse than them on these tasks. Considering human subjectivity in our evaluation, the improved MI and Z score performed comparatively similar to MI and phi-squared, respectively.

Our experiments show that CORDER, phi-squared, Z score, B1, and B2 favor high frequency associations, while two MI methods favor low frequency associations. Since high frequency associations often correspond to the intuition about associations, CORDER, phi-squared, Z score, B1, and B2 are the better choices for finding relevant buddies. To study the influence of different window sizes, we have experimented with window sizes 20, 50, 100, 200, and 300 on the five algorithms which use window sizes and found that the average ranking correlations of these five algorithms increase when the window size increases from 20 to 50, and 50 to 100, respectively, while their average ranking correlations keep roughly the same for window sizes 100, 200, and 300.

CORDER as the best performing algorithm can find truly relevant buddies and rank them in the correct order in most of the queries. However, when the ranking accuracy of CORDER for a query is low, it is often the result that there are no clear relevant buddies for the query topic. For example, consider the query “natural language processing” (NLP) in Table 1. “NLP” is not a major research area in KMi (hence there are proportionally fewer pages mentioning it) and most evaluators do not have a clear idea about who are associated to it, this is reflected in the standard deviations of users’ ratings for each relevant buddy on “NLP”, which are all between 0.683 and 1.328, given that for most of the other queries the standard deviations of users’ ratings are between 0.0 and 0.7. Associations for “NLP” are also under-represented in pages. We found that there are only two pages matching both “NLP” query and the buddies ranked as No. 1 and 2 in the collective opinion of evaluators, respectively. MI and improved MI achieved good ranking correlation for “NLP” query by chance since they favor low frequency associations.

5. Conclusions and Future Work

We began by posing the problem of just-in-time discovery of relevant knowledge, in the form of queries based on the ‘who can help me now?’ dilemma. This problem has many facets, including the need to ‘triangulate’ to deal with missing evidence (caused by ‘loopholes’ such as people not keeping their user profiles or their own web pages up to date). We briefly described an architecture that leverages the latent information available from people’s own social networks in the form of their online buddy lists, particularly for very large groups that are automatically created, such as those of enterprise-wide messaging tools and our own BuddySpace, used on very large European projects. We then ‘drilled down’ to explore one particular facet of the problem, namely how to rank the results that are found as a result of pooling the various sources of knowledge.

We have shown that the BuddyFinder system can help users’ online collaboration by enabling them to search for buddies matching their interests specified in term-based queries. Since current instant messaging tools mostly rely on registration information for buddy search, they are susceptible to fraud, limited information, and out-of-date information. BuddyFinder can enable more trustworthy, more versatile, and more up-to-date buddy search. This can be particularly potent in the hands of a ‘power user’, who (by analogy with a Google ‘power user’) can drive BuddyFinder with carefully targeted searches to find a source of expertise to help solve a critical problem quickly.

Our task-oriented user evaluation on two groups of users shows that BuddyFinder can find buddies highly relevant to search queries. CORDER is the best performing ranking algorithm out of the seven ranking algorithms, but very closely followed by two baseline methods and phi-squared. The good performance of the two baseline algorithms tell us that even very simple co-occurrence models can generate accurate results in buddy search. Mutual information based methods achieved the worse results among the seven algorithms indicating that they are not suitable for buddy search. Since Baseline 1 method is only based a buddy’s profile without considering his/her friends’ profiles, its good performance in our two experiments cannot guarantee that it will perform well in a real world scenario where lots of buddies’ profiles are missing and search on them will be largely based on their friends’ profiles. Taking into account window sizes and entity distances lead to better performance evidenced by the better performance of CORDER than Baseline 2 method.

Our rankings are derived from data mined from a collection of documents. In this way it gives a wider view of the “world” of a domain than data from a single source, such as registration information provided by users. We are continuing to test potential improvements to our buddy search approach.

First, we are working on improving our ranking algorithm. We have experimented with combining CORDER and phi-squared measures in our evaluation on two groups of users in Section 4, and the combined measure did not perform significantly better than either of them. In associating a query with a buddy, we are considering giving higher weight to documents in the buddy’s profile than documents in his/her friends’ profiles, and the challenge is how to set the weight. We will integrate CORDER in a two-stage language model consisting of a document relevance model and a co-occurrence model in expert search. Document internal structures will be considered in our approach. For example, in many scientific papers, the author of these papers is commonly mentioned once at the top in the author section and not mentioned again in the result of the paper. By identifying sections such as author sections of these of documents, we can associate experts mentioned in one section of a document with query terms in other sections of the same document. We will also investigate using coreference information such as pronouns (he, she etc.) to recognize the buddy's identity in documents.

Second, we are working on extracting and maintaining structured information, e.g., in the form of FOAF files about users, from mainly unstructured information in web pages using BuddyFinder's text mining approach for relation discovery between online users. For organizational use, we are considering using our text mining approach to keep an organizational ontology up to date with knowledge embedded in documents produced by the organization. The ontology could be used by systems such as ONTOCOPI [Alani *et al.* 2003] and Flink [Mika] for community of practice discovery and buddy search.

Third, we aim to take part in the TREC (Text REtrieval Conference, http://trec.nist.gov/) Enterprise Search Track, where a crawl of the W3C (www.w3.org) site is used as test collection. CORDER will be further evaluated in terms of its effectiveness in discovering significant associations between entities, e.g., Person-Topic associations, to complete the Expert Search task.

Fourth, Burgess *et al.* [1998] proposed the Hyperspace Analogue to Language (HAL) model to capture the meaning of a word by examining its co-occurrence patterns with other words in the language use, e.g., a corpus of text. HAL represents words as vector spaces of other words, which occur with the target words within a certain distance, e.g., a text window. The associations between concepts can be computed via different means of comparing their underlying vector representations [Bruza *et al.* 2004;

Song and Bruza 2003; Song *et al.* 2004]. As HAL model is based on the co-occurrences as well as distances between concepts, the CORDER system can be naturally integrated as one of the association derivation mechanism. We have started integrating HAL model with CORDER, e.g., to tackle users' vague search queries [Zhu *et al.* 2006]. Since HAL model provides more informative view of NEs in terms of a multi-dimensional vector and multiple ways for both explicit and implicit entity relation inferencing, we can further improve our BuddyFinder system.

6. Acknowledgements

Work on BuddySpace and BuddyFinder is supported by the European Community under the Information Society Technologies (IST) programme of the 6th Framework Programme for RTD – project ELeGI, contract IST-002205. This document `does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein. Work on CORDER was partially supported by the Designing Adaptive Information Extraction from Text for Knowledge Management (Dot.Kom) project, Framework V, under grant IST-2001-34038 and the Advanced Knowledge Technologies (AKT) project. AKT is an Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. This research was also partially supported by the Brazilian National Research Council (CNPq) with a doctoral scholarship held by Alexandre Gonçalves. We are grateful to Jiri Komzak for the implementation of BuddySpace and significant input into BuddyFinder, Martin Dzbor for significant input into the concepts, design, and implementations underlying BuddySpace, and Enrico Motta and Marta Sabou for helpful discussion.

7. REFERENCES

Adams, D. J. (2001) *Programming Jabber Extending XML Messaging*. O'Reilly.

Alani, H., Dasmahapatra, S., O'Hara, K., Shadbolt, N. (2003) Identifying Communities of Practice through Ontology Network Analysis. *IEEE Intelligent Systems* 18(2): 18-25.

Allison, C., Cerri, S.A., Gaeta, M., Ritrovato, P., Salerno, S. (2003) Human Learning as a Global Challenge: European Learning Grid Infrastructure. In *Varis, T., Utsumi, T. and Klemm, W. (eds.), Global Peace Through the Global University System*, RCVE, Tampere, pp. 465-488, http://www.terena.nl/conferences/tnc2004/core_getfile.php?file_id=576

Bruza, P.D., Song, D., McArthur, R. (2004) Abduction in semantic space: Towards a logic of discovery. *Logic Journal of IGPL*. Volume 12, Number 2, March, pp. 97-109.

Burgess, C., Livesay, K., Lund, K. (1998) Explorations in context space: words, sentences, discourse. *Discourse Processes*, 25(2&3), 211-257

Chakraborty, R. (2001) Presence: A Disruptive Technology. Presentation at *JabberConf*, Denver.

Church, K., and Hanks, P. (1989) Word Association Norms, Mutual Information and Lexicography. *Association for Computational Linguistics*, Vancouver, Canada.

Church, K. (1991) Concordances for Parallel Text. *Seventh Annual Conference of the UW Centre for the New OED and Text Research*, Oxford, England.

Croft, W.B., Turtle, H.R., Lewis, D.D (1991) The Use of Phrases and Structured Queries in Information Retrieval. In Proc. of *SIGIR 1991*, ACM Press, New York, pp. 32-45.

Conrad, J. G., Utt, M. H. (1994) A System for Discovering Relationships by Feature Extraction from Text Databases. In Proc. of *SIGIR 1994*. pp. 260-270.

Eisenstadt, M., Komzak, J., Dzbor, M. (2003) Instant Messaging + Maps = Powerful Collaboration Tools for Distance Learning. In Proc. of *TelEduc03*, Havana, Cuba.

Gibbons, J.D. (1976) *Nonparametric Methods for Quantative Analysis*, Holt, Rinehart and Winston.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D. J., Kamm, C. A. (2002) The Character, Functions, and Styles of Instant Messaging in the Workplace. In Proc. of *ACM conference on Computer Supported Cooperative Work (CSCW 2002)*, pp. 11-20.

Jin, H., Ning, X., Chen, H., and Yin, Z. (2006) Efficient Query Routing for Information Retrieval in Semantic Overlays. In Proc. of *the 21st Annual ACM Symposium on Applied Computing (SAC'06).*

Lave, J. and Wenger, E. (1991) *Situated Learning. Legitimate Peripheral Participation*. Cambridge University Press.

Meskill, J. A Social Networking Services Meta-List: http://socialsoftware.weblogsinc.com/entry/9817137581524458/

Mika, P. Flink: The WHo is Who of the Semantaic Web http://flink.semanticweb.org

Nardi, B.A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., Hainsworth, J. (2002) Integrating Communication and Information through ContactMap. *Communications of the ACM*, 45:4, pp. 89-95.

Resnik, P. (1999) Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*. 11:95-130.

Setlock, L. D., Fussell, S. R., Neuwirth, C. (2004) Taking It Out of Context: Collaborating within and across Cultures in Face-to-Face Settings and via Instant Messaging. In Proc. of *ACM conference on Computer supported cooperative work (CSCW 2004)*, pp. 604-613.

Song, D., and Bruza, P.D. (2003) Towards context-sensitive information inference. *Journal of the American Society for Information Science and Technology (JASIST)*, Vol. 54, No. 4, pp. 321-334.

Song, D., Bruza, P.D., and Cole, R.J. (2004) Concept learning and information inferencing on a high-dimensional semantic space. *ACM SIGIR 2004 Workshop on Mathematical/Formal Methods in Information Retrieval (MF/IR'2004)*.

Vechtomova, O., Robertson, S., Jones, S. (2003) Query Expansion with Long-Span Collocates. *Information Retrieval*, 6(2), pp. 251-273.

Vogiazou, I.T., Eisenstadt, M., Dzbor, M., Komzak, J. (2005) From Buddyspace to CitiTag: Large-scale Symbolic Presence for Community Building and Spontaneous Play. In Proc. of *the ACM Symposium on Applied Computing (SAC’05)*, Santa Fe, New Mexico, March 13 -17, pp. 1600-1606.

Zhu, J., Gonçalves, A. L., Uren, V. S., Motta, E., Pacheco, R. (2005) Mining Web Data for Competency Management. In Proc. of *2005 IEEE/WIC/ACM International Conference on Web Intelligence 2005 (WI'2005)*, France, pp. 94-100.

Zhu, J., Eisenstadt, M., Song, D., and Denham, C. (2006) Exploiting Semantic Association To Answer Vague Queries. In *Proc. of the Fourth International Conference on Active Media Technology (AMT 2006)*, Brisbane, Australia.