DYNAMIC INTEREST PROFILES
Tracking User Interests Using Personal Information

Justin Lessler, Stefan Edlund, Joann Ruvolo, Vikas Krishna
IBM Almaden Research Center, 650 Harry Road, San Jose California, USA

Keywords: Personal Information Management, E-Mail

Abstract: When building applications it is usually the case that developers are forced to focus on "one size fits all" solutions. Customization is often burdensome for the user, or so complex that it would be unrealistic to ask an end user to undertake this task. In the areas of personal information management and collaboration there is no reason to accept this limitation, as there is a body of information about the user that reflects their interests: namely their personal documents. The Dynamic Interest Profile (DIP) is a system intended to track user interests, allowing for the creation of more intelligent applications. In this paper we discuss our approach to implementing the DIP, challenges that this implementation presents, as well as the security and privacy concerns that the existence of such an application raises.

1 INTRODUCTION

One of the biggest challenges in developing software is that all users are different. For example when retrieving email some users make meticulous use of folder structures to file and find email, while others keep all email in one location and use search or sorting to find needed documents (Ducheneaut, 2001). Because of these differences, attempts to add intelligence to applications often fail. For instance due to its tendency to give advice to users who neither want nor need it, the Microsoft Office Assistant (a.k.a. Clippy the paperclip) was hated by many.

Personal information management (PIM) applications are not free from the problem of dealing with user difference. Lacking any better course of action these applications tend to cater to the lowest common denominator (e.g. address completion applications suggest names in alphabetical order and search returns results based on lexical similarity). But in the case of personal information management there is a ready store of information that can be used to tailor applications to individual users, namely their personal documents. The Dynamic Interest Profile (DIP) is our implementation of a system that tracks user interests and provides an interface for applications to easily access this information.

The DIP recognizes four different types of entities: people, documents, terms, and collections. For each of these the DIP assigns both an all time importance and detects entities of emerging importance. The DIP is able to determine these values based on a user's personal documents.

Tracking user information for the purpose of making these determinations carries with it a set of challenges. The algorithms for determining importance involve a classic computation vs. storage trade-off. Depending on the PIM system, maintaining the information necessary to keep the DIP up to date may require additional storage. This is an especially important challenge as the infrastructure requirements for enterprise email systems are usually quite large already.

In addition to the practical implementation challenges, privacy and security concerns are raised by the very existence of a process that tracks user interests. Many users may balk at the idea of their interests being tracked, and mistrust assurances that the information is not being shared. Even if users trust that their information will not be misused, it is necessary to provide the same level of security for this summary information to that which is provided for the documents characterized.

It is our belief that despite these challenges the functionality the Dynamic Interest Profile provides is well worth the effort. Applications such as search and address completion, when effectively tailored to a user's interests, can greatly ease their lives. Collaborative applications can be automated to

Lessler J., Edlund S., Ruvolo J. and Krishna V. (2004).
DYNAMIC INTEREST PROFILES - Tracking User Interests Using Personal Information.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 281-288
DOI: 10.5220/0002650702810288

indicate when a user's interests provide an opportunity to leverage the skills of others. An inbox can be sorted not just by date or sender, but by importance as well. In an era when many of us spend much of our day dealing with email and other documents, seemingly small improvements such as these can significantly improve both productivity and job enjoyment.

2 ALGORITHMS

User profiling has been an area of interest for many years. A variety of techniques have been developed, based on different data sources (shopping habits, browsing habits, user feedback (Widyantoro et. al., 1999), etc.) and utilizing different algorithmic techniques (machine learning, keyword analysis, social filtering, etc. (Soltysiak and Crabtree, 1998)). In our approach we use user data and actions to determine the overall and emerging importance of four different entity types: people, terms, documents, and collections. For each instance of one of these types (e.g. a person, a term) we use a variety of statistical techniques specific to the type in order to determine the instance's importance to the user.

2.1 People Importance

The importance of a person to a user is determined by analyzing the relationship between the user and the person as indicated by the user's email. Those people who are most important are the ones with whom a user has the strongest relationships. Relationship strength is based on the relationship's (Whittaker et. al., 2002):

- Longevity
- Currency
- Reciprocity
- Exclusivity
- Frequency

This information is captured using a linear combination of functions where each function represents a particular aspect of the relationship. The equation for calculating a person *p*'s importance is:

$$score(p) = \lambda_1 f_{longevity} + \lambda_2 f_{currency} + \lambda_3 f_{reciprocity} \\ \lambda_4 f_{exclusivity} + \lambda_5 f_{frequency}$$

The component functions of this equation can be described as follows:

$f_{longevity}(p_1) > f_{longevity}(p_2) \rightarrow p_1$ has older interactions with the user than p_2.

$f_{currency}(p_1) > f_{currency}(p_2) \rightarrow p_1$ has more recent interactions with the user than p_2.

$f_{reciprocity}(S_{p_1}, R_{p_1}) > f_{reciprocity}(S_{p_2}, R_{p_2}) \rightarrow p_1$ has more bidirecitonal interactions with the user than p_2.

$$f_{exclusivity}(p) = \frac{f_{1:1}(p)}{|S_p| + |R_p|}$$

$$f_{frequency}(p) = |S_p| + |R_p|$$

Where:

S_p = the set of communications sent to *p*.

R_p = the set of communications received from *p*.

$f_{1:1}(p)$ = the number of 1 on 1 interactions between the user and *p*.

2.2 Term Importance

Terms are words and phrases that carry meaning in a user's documents. Parsing documents to find terms falls outside the scope of this paper, but there is a large body of literature on this topic, (Silva & Lopes, 1999), (Reyle & Saric, 2001), (Thanopolous et. al., 2003), and (Manning and Shutze, 2000), just to name a few. Once terms have been identified, we define a term as important to the user when it is:

- Associated with the user
- Descriptive
- Discriminative
- Relevant

As with people importance, this information is captured by using a linear combination of functions where each function represents one of the above qualities. The equation for calculating a term *t*'s importance is:

$$score(t) = \lambda_1 f_{user} + \lambda_2 f_{subject} + \lambda_3 f_{folders} + \lambda_4 f_{time}$$

The component functions of this equation have the following properties:

$f_{user}(t_1) > f_{user}(t_2) \rightarrow t_1$ is more associated with the user than t_2.

$f_{subject}(t_1) > f_{subject}(t_2) \rightarrow t_1$ is more associated with document subjects than t_2 and hence considered to be more descriptive.

$f_{folder}(t_1) > f_{folder}(t_2) \rightarrow t_1$ is more useful in determining what folder a document containing the term resides in than t_2 and hence considered to be more discriminative.

$f_{time}(t_1) > f_{time}(t_2) \rightarrow t_1$ is more associated with the current time period than t_2 and hence considered to be more relevant.

2.3 Document Importance

Documents are the basic unit in a PIM system. A document may be a calendar entry, an email, a word processing document, etc. Document importance is determined by considering document content, meta-data, and usage. Unlike the calculation of people and term importance, document importance is dependant upon other DIP calculations, namely people and term importance. The exact equation varies based on document type, in the case of an email, *d*, it is:

$$score(d) = \lambda_1 f_{usage} + \lambda_2 score(a) + \frac{\lambda_3}{|R|} \sum_{p \in R} score(p) + \lambda_4 \max_{t \in S} score(t) + \frac{\lambda_5}{|B|} \sum_{t \in B} score(t)$$

where a is the document author, R is the document recipient list, S is the set of terms in the subject and B is the set of terms in the body.

The function $\boldsymbol{f_{usage}}$ is a function with the property that:

$f_{usage}(d_1) > f_{usage}(d_2) \rightarrow$ d_1 has been accessed more than d_2.

The astute reader may ask why we choose to average the importance of the body terms and take the max of subject term importance when calculating document importance. This is a result of our observation of the differing content of the document subject and the document body. The document subject or title tends to contain one or two key terms that describe the document while the other terms serve as window dressing. Because of this averaging over the importance of all of the terms in the title leads to inaccurate comparisons. For instance, if "golf" is my most important term a document titled "Improving your Golf" should not be considered less important than one titled "Golf". Document bodies on the other hand tend to contain similar numbers of filler words as they are generally written in prose. Because of this, averaging over the importance of terms in the bodies still leads to comparable results.

2.4 Collection Importance

Collections may be of varying types, ranging from email folders to public discussion databases. What constitutes a collection is dependant on the specific PIM system being used, but for all collections the broad attributes that indicate importance are the same: collection content and usage. For some collections meta-information about the collection may also be important. As with documents, the calculations for discovering collection importance are dependant on other DIP calculations. The general form of the calculation of the importance of a collection *c* is:

$$score(c) = \lambda_1 f_{access} + \lambda_2 f_{change} + \frac{\lambda_3}{|D_c|} \sum_{d \in D_c} score(d)$$

where D_c is the set of documents in collection *c*.

The component functions of this equation have the following properties:

$f_{access}(c_1) > f_{access}(c_2) \rightarrow c_1$ is accessed by the user for reads more often than c_2.

$f_{change}(c_1) > f_{change}(c_2) \rightarrow c_1$ is modified by the user more often than c_2.

Note that the document score *score(d)* incorporates term and people importance, hence it reflects the frequency of important people and terms in the collection.

2.5 Emerging Items

It is not always the case that we are interested in items that are important over all time, sometimes we

are interested in items that are of emerging importance. In order to detect emerging items we calculate how associated the item is with the recent time period using point wise mutual information, Chi-Squared, or some equivalent statistical test (Yang and Pedersen, 1997) (Manning and Shutze, 2000) (Duda et. al., 2001). In the case of terms and people the number of references to the person or term is used for this calculation. In the case of collections the frequency of access and modification by the user is used. The general equation for detecting the extent to which a term *x* is of emerging importance is:

$emerge(x) = \mathrm{A}(x,t)C_x$ where t is the current time period, $\mathrm{A}(x,t)$ is the strength of association between x and t, and C_x is the frequency of term x.

2.6 Efficient Calculation and Storage

In order for the DIP to be deployable in a real world setting it is necessary for the importance and emergence calculations to be performed efficiently. In order to completely recalculate many of the functions involved the entire corpus of documents would have to be examined, a very time consuming operation. To allow for efficient recalculation, intermediate results are stored. These allow recalculation based only on the delta in the document corpus. These intermediate results take the form of feature vectors. A feature vector is a set of keys (name, terms, etc.) and associated counts. These feature vectors represent those intermediate parts of the calculation that can be incrementally changed when documents are created, deleted, or modified.

Once DIP scores have been recalculated the results are stored as score vectors representing the importance of the DIP entities to the user. Score vectors are sets of keys associated with a floating point score. Details of the storage framework are discussed in section 3.

2.7 Limited DIP Implementation

In order to validate the underlying concepts of the DIP and provide a platform for further experimentation we have developed a limited DIP implementation. This implementation performs the calculations for term importance, as well as limited versions of the document importance and people importance calculations, based upon user emails.

When calculating term importance we treat the subject and body as sets of terms. We do not use any information as to the order of the words or the frequency of their occurrence within the document. The implementations of the sub functions making up term importance are:

$$f_{user} = \max[0, \mathrm{I}(t, user)\log(C_{t,user})]$$

$$f_{subject} = \max[0, \mathrm{I}(t, subject)\log(C_{t,subject})]$$

$$f_{folders} = \max_{f \in Folders}[\mathrm{I}(t, f)\log(C_{t,f})]$$

$$f_{time} = \max[0, \mathrm{I}(t, year)\log(C_{t,year})]$$

where $C_{t,x}$ is the number of occurrences of term t in set x and $\mathrm{I}(x,y)$ is the point wise mutual information between x and y. See (Yang and Pedersen, 1997) and (Manning and Shutze, 2000) for a description of point wise mutual information.

In order to calculate $\boldsymbol{f_{user}}$ it is necessary to have some notion of the general frequency of use of a term. In order to determine this we maintain an organizational "background vector" that contains the number of times each individual term has appeared overall in an organization's email. In environments where security is a concern, it is possible to replace this organizational background vector with one based on the general frequency of words within the language being used.

People importance in the draft implementation is based upon a limited notion of frequency and reciprocity characterized by the equation:

$score(p) = \lambda_1 |S_p| + \lambda_2 |R_p|$ where S_p is the set of emails sent to p, R_p is the set of emails received from p and $\lambda_1 >> \lambda_2$.

Document importance in the draft implementation is the same as the full equation described in section 2.3 minus the usage term:

$$score(d) = \lambda_1 score(a) + \frac{\lambda_2}{|R|}\sum_{p \in R} score(p) + \lambda_3 \max_{t \in S} score(t) + \frac{\lambda_4}{|B|}\sum_{t \in B} score(t)$$

Both term importance and people importance are stored in the prototype implementation, while document importance is calculated on the fly.

3 IMPLEMENTATION

As we set about implementing the DIP component into a piece of enterprise collaboration software, we initially compiled a list of requirements:

Single- and multi-user support. We had to ensure that we were able to maintain interest profiles for a potentially large number of users. In addition, maintaining a DIP for a single user on a single workstation working off-line was also required.

Abstract document model. The DIP is required to handle information from a diverse set of domains, including: email documents, calendar & scheduling data, instant messaging, and personal documents. A document model was needed for transforming data from such domains into an abstract format, capturing the information needed for compilation of interest profiles.

DIP object model The object model captures summary interest profiles for entities. As discussed in Section 2, summary interest profiles contain importance scores associated with terms, people, and collections.

Incremental updates. The DIP is compiled from a large body of documents generated over potentially long periods of time. The DIP infrastructure had to be able to incrementally update the object model to gradually refine results over time.

DIP persistence model. Incremental updates require a persistence model to store interest profiles constructed over long periods of time. Also, since maintaining the DIP for multiple users consumes space, persistence allows us to bring a smaller "active" working set of the interest profiles into server memory on demand.

Privacy and Security. As mentioned earlier, documents can contain sensitive information that people aren't always willing to disclose. Mechanisms for preventing access to sensitive personal information were needed.

We now proceed to discuss the details of the implementation.

3.1 DIP Document Model

The document model adopted in the implementation summarizes the original document with regard to people and terms. As discussed in Section 2, information on people is extracted from fields within the original document, preserving the relationships that exist between documents and people. This allows us to give a higher importance to (for instance) a person on the "to:" list of an email versus a person on the "cc:" list.

For email documents, people are represented by their email addresses. However, since multiple addresses can map to a single email account, ambiguity must be resolved. When translating between original documents and abstract DIP documents, we attempt to normalize email addresses to their unique id maintained by a profile management system. In our target enterprise collaboration software, WebSphere Member Manager™ (WMM) is the global solution for managing member profiles.

Terms encountered in header fields and the body of documents are compressed into a set of unique terms as discussed in Section 2. Bags of unique terms are maintained for each field deemed relevant for compilation of interest profiles. Also, any source collection information included in the original document is kept.

3.2 DIP Object Model and Persistence

We adopted the vector model discussed in Section 2 for the DIP object model, where a collection of feature and score vectors comprise the interest profile for a user. This allows us to extend the model by adding additional features as needed. The two main features captured are term and people occurrence, but the model also contains 24 smaller vectors representing terms and people for the last 12 months. Counts computed from documents processed during a more limited timeframe can be more informative than the global vectors since they incorporate a recency factor into the calculations, and they allow applications to track how interests change over time.

Feature vectors are either personal, such as the vector describing people importance, or shared. The vectors contain a user identifier (or an identifier for the global user), a vector id, and a set of features, where each feature has a feature name – for instance a term – and a count/score. The feature vectors are persisted in a relational database (DB2 UDB™) in a table having these four columns. To optimize for quick retrieval of vector features, a B-tree index on user id and vector id is maintained. The table is a read-only table most of the time, minimizing lock contention and maximizing concurrency. While the DIP is being updated, applications are forced to wait until the update is complete.

3.3 Offline Operation

We have also implemented an offline version of the DIP, running on a workstation for a single user. When in offline mode, a client runs the DIP algorithms locally on a smaller set of personal documents.

As mentioned in Section 2, shared information is used, though minimally, to compute the important terms for a user. Shared information in the form of a background vector is stored on a common server, and it is replicated to the client whenever the client goes online. The replica of the background vector on the client improves the accuracy of the term importance calculation. The client keeps track of local changes to the background vector, and it updates the shared representation when going back online.

3.4 Security and Privacy

A DIP server falls under a category of systems described as "user modeling servers" (Fink and Kobsa, 2000), that is servers that support applications requiring user models for personalization features. As such, it needs mechanisms to protect personal information from unauthorized access and potential eavesdropping. Many factors are taken into consideration when determining the level of security and privacy to provide, for instance the privacy preferences of the user, self-regulatory privacy principles within an organization, or the privacy laws of the country or state.

A combination of anonymization and encryption techniques can be deployed to ensure the privacy of the individual. In the DIP system, personal information is encrypted at the database level to ensure privacy. This protects from both eavesdropping and from individuals, such as database administrators, having high levels of authorization. Methods for anonymization (i.e. concealment of the relationship between a particular user and data about him) are currently being investigated for future implementation.

3.5 Implementation Overview

Figure 1 shows an overview of the DIP implemented as a processing agent in a pipeline of agents

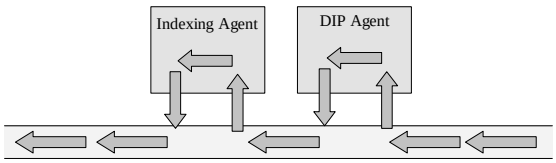

Figure 1: Agents operating on a stream of documents

operating on a document stream. Processing agents are designed to carry out specific tasks, for instance indexing documents for search, flagging documents that contain viruses or – in the case of the DIP – updating interest profiles. Some agents modify documents as they travel down the stream, for instance an agent expanding group names into a list of members. While the DIP agent does not modify documents, it is easy to imagine a system where the agent tags documents with information such as sender importance, text importance, or an aggregated overall importance score. Observe that such tagging provides a snapshot of the importance calculated when the document was first received, and that the importance will change when the underlying interest profile is updated. Sometimes it is useful to recall the importance of a particular document at the time it was received, but as a person's interest profile changes, it is often desirable to bring old documents to the forefront. For this reason, we do not put scores into the documents.

Documents on the stream originate from a variety of applications, including email, calendaring & scheduling, and instant messaging. A flag may be associated with documents indicating whether they have been updated or deleted from their corresponding data repositories. This allows us to update the DIP object model correctly. For instance, spam ignored by spam filters is typically deleted immediately from an inbox. Often we can detect that an email has been deleted early enough to remove it from the queue before it has been incorporated into an interest profile. Efficient spam filters also help keep DIP profiles more accurate.

In real-world mail systems, message transfer agents (MTAs) ensure that incoming messages are delivered locally into the mail store, and outgoing messages are sent using SMTP to their destinations. Mail transfer agents are operating on queues of incoming/outgoing email, and the DIP processor is simply another agent for such queues. Similar mechanisms exist for other messaging systems (e.g. instant messaging).

Figure 2 shows the agent passing documents to the DIP translator, which handles the conversion from the stream document format to the DIP document format. A translated document is pushed onto internal queues, where they are stored until a task is triggered to update the interest profiles. Translation, name normalization, and queue operations are handled quickly to prevent the DIP agent from becoming a bottleneck in the system. DIP documents on the queues are periodically popped and fed to the processor, which incrementally updates the interest profiles. There is a many-to-one mapping between documents and profiles, enabling the processor to update one profile at a time by batching documents together. Taking email as an example, the batch contains all emails sent by or received by an author during the timeframe since the last DIP update was carried out. As a consequence, documents are tagged to indicate which person (or group) they are associated with. For instance, an email sent by person A to person B with a carbon copy to person C generates three new documents (but not necessarily on the same mail system). To assist in the task of batching documents, we use multiple queues internally, one for each unique person.

A batch of documents is processed in memory building a corresponding set of delta vectors used to incrementally update the underlying interest profile. In our current implementation, a persisted interest profile is stored in a relational database, and the implementation of the delta vectors allow us to track the SQL inserts, deletes, and updates needed to incorporate the new set of documents. To ensure database consistency, all updates to an interest profile are treated as a single transaction.

Applications access interest profile information through a public application programming interface (API) in a read-only mode. The API has methods for retrieving people and terms ranked by importance, and also methods for computing aggregated importance scores for documents. Search applications can use this functionality to refine search results according to personal interest profiles.

The DIP is implemented as an enterprise business component based on the J2EE recommended multi-tiered architecture. The four tiers in our implementation are, bottom up, the Resource tier housing the data store and data accessor classes, the Service tier containing Enterprise Java Beans, the Workspace tier containing the client side representation of the server classes, and the User tier in which DIP clients operate.

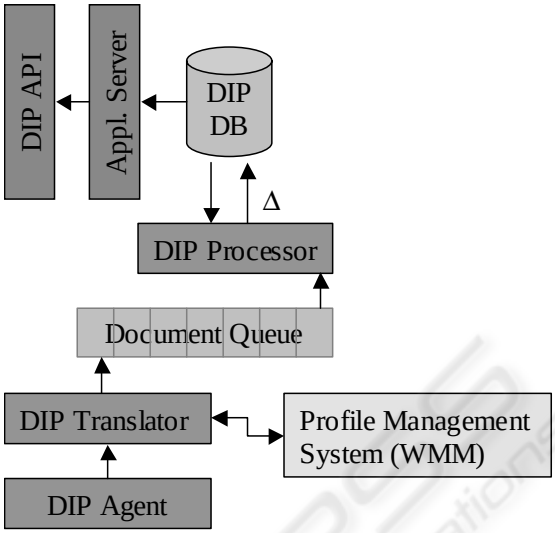

Figure 2: High level architecture and data flow in the DIP

4 DISCUSSION

Our early experiences with testing the DIP have been quite encouraging. Even the relatively simple versions of our algorithms in the described implementation yield quite good results, and developing useful applications based on the DIP is quite easy. Early tests of DIP technology have led to some interesting observations on the use of DIP technology.

It is often the simplest DIP based applications that prove the most compelling. For instance: inbox sorting and address completion are always a hit at demos, while the more complex displays of the people associated with important terms receives a more muted response.

Displaying raw DIP results, such as a user's *n* most important people, is not recommended. Terms and people that may be computationally important in determining importance may seem "wrong" when viewed by the user. For instance, a rare abbreviation commonly used by my manager may be valuable in discovering document importance, but I would not perceive it as a term that is important. We have begun research on filtering based on controlled vocabularies and ontological information to help in the production of DIP results that are more human friendly.

Two of the most compelling DIP based applications are the dynamic address book and inbox sorting. Our target enterprise collaboration software uses the DIP to maintain dynamic address books, i.e.

address books that are automatically compiled from the most important people as determined by the DIP. In addition, the collaboration software allows users to search documents and refine the results using personal interest profiles. Documents deemed important by the DIP are ranked higher, increasing the likelihood that users find what they are looking for.

One of the favorite demo applications we developed was sorting the user's inbox by importance. The DIP proved to do a wonderful job of giving spam and mailing list articles low importance and important work communications from coworkers and managers high importance. By allowing for some adjustment of the weights involved the sort could be further refined, sending either documents with important topics or by important people to the top of the list. We hope to perform formal validation of this functionality and make it available in products in the future.

This paper has discussed the concept of Dynamic Interest Profiles and the integration of such profiles into enterprise collaboration software. As important next steps our team plans to: formalize the concepts described, perform more validation, and see what role standards can play in this area.

We would like to acknowledge the following people for their contributions and support in this work: Carl "Pooter" Kraenzel, Mike O'Brien, Kate Glickman, Sesha Baratham, Igor Belakovskiy, Niklas Heidloff, Gurushyam Hariharan, and Bill Cody.

REFERENCES

Ducheneaut, N., & Bellotti, V. 2001a. Email as Habitat: An Exploration of Embedded Personal Information Management. *Interactions*, 8(5), 30-38.

Duda, R. O., Hart, P.E., Stork, D.G., 2001. *Pattern Classification.* John Wiley & Sons, Inc. 2nd Edition.

Fink, J. and Kobsa, A., 2000. A review and analysis of commercial user modelling servers for personalization on the world wide web. *User Mod. User-Adapted Interact.10, 3-4, 209-249*

Manning, C. D., & Schutze, H. 2000. *Foundations of Statistical Natural Language Processing*, The MIT Press. Cambridge Mass., 3rd printing.

Nardi, B., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., Hainsworth, J., 2002. Integrating communication and information through ContactMap. *Communications of the Association for Computing Machinery*

Reyle, U. & Saric, J. 2001. Ontology Driven Information Extraction. In *Proceedings of the 19th Twente Workshop on Language Technology,* University of Twente.

Schiaffino, S.N., Amandi, A. 2000. User Profiling with Case Based Reasoning and Bayesian Networks. *IBERAMIA-SBIA 2000 Open Discussion Track.*

Silva, J., & Lopes, G., 1999. Extracting Multiword Terms from Document Collections. In *Proceedings of the VExTAL, Venezia per il Trattamento delle Lingu, Universitá cá Foscari, Venezia November 22-24.*

Soltysiak, S., & Crabtree, B. 1998. Knowing Me Knowing You: Practical Issues in the Personalisation of Agent Technology. In *Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98).*

Thanopolous, A., Fakotakis, F., Kokkinakis, G. 2003. Text Tokenization for Knowledge-free Automatic Extraction of Lexical Similarities. *TALN 2003. Traitement Automatique des Langues Naturelles.*

Whittaker, S., Jones, Q., Terveen, L., 2002. Contact Management: Identifying Contacts to Support Long-Term Communication. *CSCW'02*

Widyantoro, D.H., Ioerger, T.R., Yen, J., 1999. An Adaptive Algorithm for Learning Changes in User Interests. *Proceedings of the Eighth International Conference on Information and Knowledge Management*, 405-412.

Yang, Y., Pedersen, J., 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning.*