A Unified User Profile Framework for Query Disambiguation and Personalization

Georgia Koutrika, Yannis Ioannidis

University of Athens, Greece
{koutrika, yannis}@di.uoa.gr

Abstract. Personalization of keyword searches has attracted interest in the research community as a means to decrease search ambiguity and return results that are more bound to be interesting to a particular user. We describe a term-based user profile that treats query disambiguation and personalization as a uniform term rewriting process. Its key feature is the representation of connections between terms based on possible rewritings between them on a per user basis. We present a query-rewriting algorithm for such query disambiguation based on the proposed profiles. Preliminary experimental results show the potential of the overall approach.

1 Introduction

Traditionally, search engines are deterministic in that they should return the same set of documents to all users with the same query at a certain time. Therefore, it is inherent that search engines are not designed to adapt to personal preferences. This deterministic behavior is desired in order to provide the users with the same view of information; however, in the context of the World Wide Web, it often hinders users from locating relevant information. There are several aspects to the problem. First is the problem of abundant information made available to a wide spectrum of users with possibly different information needs. Only a fragment of this information is useful to a single user. Typically, only top results of a search are browsed by a user. If interesting information is not found there, a new query may be submitted or the task may be abandoned. A second related problem is that users typically issue poorly defined queries of very few terms. For example, for the query "`java programming`", a user may be interested in tutorials, while another may be interested in source code. This ambiguity in the query is further amplified by the existence of synonyms and homonyms. Synonyms are two words that are spelt differently but have the same meaning. Homonyms are words that are spelt the same but have different meanings. For example, for the query "`apple`", some users may be interested in documents dealing with "`apple`" as "`fruit`", while other users may want documents related to Apple computers. Consequently, without prior knowledge, there is no way for the search engine to predict user interest from simple text based queries.

The above situation gave rise to the idea of personalized search. In particular, storing user preferences in user profiles gives a system the opportunity to return more

focused personalized (and hopefully smaller) answers. The general architecture of a personalized search system is depicted in Fig. 1.

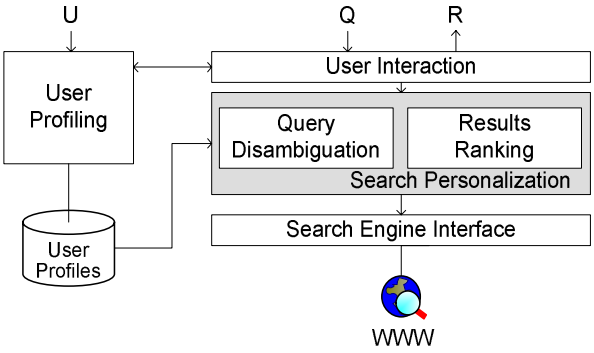


Fig. 1. Architecture of a Personalized Search System

The system keeps a repository of user information (*User Profiles*) that is either inserted explicitly by the user or collected implicitly by monitoring user interaction with the system (*User Profiling*). The user interacts with the digital library through a *User Interaction* component, issuing keyword queries (*Q*) and then browsing the content retrieved (*R*). A query may not represent a unique information need, resulting in generation of many irrelevant answers. For example, a user searching for information on the Java programming language may submit the query "`Java`". The search personalization module may be on top of a traditional search engine or may be integrated into it. The primary ways to personalize a search for an active searcher are *query disambiguation* and *results ranking*. Query disambiguation is typically performed by adding more terms in a query (query augmentation). For example, information that one usually asks about "`programming`" may be recorded. As a result, the query "`Java programming`", which is closer to the actual user information need, is produced. Results ranking comprises re-ordering results returned by the underlying search engine based on user preferences. For example, instead of modifying the initial user query, the information about "`programming`" may be used to place results regarding "`Java programming`" on top of the results returned.

Contributions. Our work concerns keyword searches over unstructured data. We provide a term-based user profile that treats personalization and query disambiguation as a unified term rewriting process (Section 3). Its unique feature is the representation of connections between terms expressing possible rewritings between them. Based on this kind of user profile, we describe a query rewriting algorithm for query disambiguation and personalization (Section 4). Furthermore, results may be ranked based on the proposed user profile. Our framework is independent of the underlying search engine and of the profiling method employed (of course, user profiling effectiveness is an important ingredient for the success of a personalized system). Section 5 provides an overview of a prototype system implementing the proposed framework and discusses user profiling and implementation of searching using Google as the underlying search engine. Experimental results show the potential of the proposed framework.

2 Related Work

Relationships of our work with previous research efforts are sketched below:

Information Retrieval and Filtering. Traditional Information Retrieval systems return the same results to all users issuing the same query [12]. Query disambiguation techniques are used in many of these systems. However, most of them aim to discover corpus-wide word relationships based on co-occurrence analysis of a whole collection (e.g., term clustering [15], similarity thesauri [11]) and they do not take into account how a person perceives word relationships. Information filtering systems employ content-based and/or collaborative filtering methods to return items interesting to a specific user according to a profile capturing long-term user interests [1, 2, 4, 7, 17].

Personalized Searches. Recently Personalized Search systems have emerged. Recall that the primary ways to personalize a search are query augmentation and results ranking. Most current approaches deal with results ranking. For example, Casper [14] ranks jobs returned to a user based on a user profile that specifies job cases previously ranked by the user. Inquirus [6] uses profiles that contain preferences about source selection and results ranking as well as terms from a predefined set that may be inserted to a query. METIORE [3] sorts an answer in the order of user preferences, giving the most interesting solution at the beginning. Their approach of personalization is based on the concept of objective. The user specifies a search objective for every new session. The result of the ranking algorithm is the degree of relevance of an object to the present objective of a user. Persona [16] re-ranks results returned by the underlying search engine based on adapted gradient ascent HITS. [9] presents results for each query under appropriate categories deduced from profiles stored. We differ from these approaches in that we employ user profiles to personalize a user search either at the level of results ranking or at the level of query augmentation. Outride [10] also performs query augmentation. Personalization based on query modification has been also proposed for structured queries over database systems [8].

User Modeling. User modeling refers to representation of user characteristics. In information filtering systems, common user profile representations are borrowed from Information Retrieval and include Boolean, vector-space, and inference models [4, 17]. Outride [10] employs user profiles based upon the ontology of the Open Directory Project (ODP), where each user has his own weighting across the top 1000 categories. In [9], a user profile consists of a set of ODP categories and for each category, a set of terms (keywords) with weights. The weight of a term in a category reflects the significance of the term in representing the user's interest in that category. In [3], for each objective all the documents evaluated are kept along with their evaluation. Each document has some representative features (keywords, author, year, etc…). These inherit the evaluation of the document that contains them. In Persona, the user profile is essentially a mapping of contexts to sets of ODP nodes [16]. A context is defined as a user query. We differ from the aforementioned approaches in that we provide a finer-grained user model which contains weighted terms and connections between them that represent term rewriting operations rather than semantic relations. Different users may have different profiles that do not adhere to some generally accepted con-

cept hierarchy. We believe that the main advantage of our approach is that we can model a user more precisely using terms and term associations that are useful to him. Moreover, this model permits the implementation of both query disambiguation and personalization as a unified process, based on a user profile that records what kind of operations should be performed in order to disambiguate and personalize a user query.

User Profiling. An important building block of a personalization approach is the collection of information about the user to generate a profile (based on the user model). User profiles for information filtering or personalized search are built either manually [6] or with the help of learning techniques [3, 9]. However, the latter typically build profiles consisting of one or more flat term vectors, while the user profile described in this paper is more complex. We describe a simple incremental algorithm for constructing such structured profiles, which we have used in order to evaluate the potential of the approach proposed.

3 User Profiles

We consider queries that are formulated as combinations of terms with the use of logical operators. A term may be a word, e.g., "`Java`", or a phrase, e.g., "`Artificial Intelligence`". We use t_1, t_2, … to denote terms. Logical operators include `AND`, `OR`, `NOT`[1] with their typical semantics. Whenever the operator among keywords is omitted, most search engines consider the logical-`AND` as the default operator. Examples of queries considered are the following:

"`Java AND programming`"

"`geological AND phenomenon`"

Parentheses are used to allow nesting of operators and formulation of complex queries. For example:

"`geological AND (phenomenon OR formation)` "

In practice, it has been observed that queries contain very few terms (one or two). That is why searches are often very ineffective. Given a user query, such as "`Java`", query terms may be combined with other terms through logical operators to produce a query, such as "`Java AND programming AND ZPress editions`", which is less ambiguous and represents user preferences. In this case, disambiguation and personalization of a query may be both viewed as two sides of the same coin, i.e. as a unified term-rewriting process. Hence, we propose a term-based user profile that represents connections between terms based on possible rewritings between them. Modification of a user query is dictated by the user profile. In particular, we model the profile of each user as a directed graph **G**(**V**, **E**) (**V** is the set of nodes and **E** is the set of edges) with the following characteristics:

- Nodes in **V** represent terms. The set of terms may be formed based on users' past interaction histories, ontologies, etc. A weight may be assigned to a term, indicating a user's interest in it. Weights are real numbers in the range [`0`, `1`], where a value of `1` indicates extreme interest, a value of `0` indicates lack of interest, and any inter-

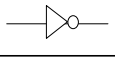
[1] Most search engines interpret this binary operator as equivalent to AND NOT.

mediate value indicates an intermediate level of interest for the term on the part of the user. Node weights may be used for ranking of results.

- Edges in **E** represent connections between terms. An edge from term t_i to term t_j may be associated with a specific logical operator and expresses a possible rewriting of t_i using t_j. More specifically, **E**=**C** ∪ **D** ∪ **N** ∪ **S**, where:
  - **C** is a set of *conjunction* edges. A conjunction edge $t_i \rightarrow t_j$ indicates that t_i is rewritten as t_i AND t_j.
  - **D** is a set of *disjunction* edges. A disjunction edge $t_i \rightarrow t_j$ indicates that t_i is rewritten as t_i OR t_j.
  - **N** is a set of *negation* edges. A negation edge $t_i \rightarrow t_j$ indicates that t_i is rewritten as t_i NOT t_j.
  - **S** is a set of *substitution* edges. A substitution edge $t_i \rightarrow t_j$ indicates that t_i is replaced by t_j. This operation is useful for dealing with cases of term misuse.

  Table 1 summarizes the edge types, their semantics, and graphical representation.

Table 1. Different edge types and their semantics

Edge Type	Semantics	Notation
conjunction	Given t_i, consider t_i AND t_j	
disjunction	Given t_i, consider t_i OR t_j	
negation	Given t_i, consider t_i NOT t_j	
substitution	Given t_i, consider t_i	

A weight may be assigned to each edge expressing the significance of the specific rewriting of the term for disambiguation/personalization of a query containing it. As with nodes, weights are real numbers in the range [0, 1] with the corresponding interpretation. Fig. 2 provides an example for each edge type. Edge weights are tagged to the edges. Nodes and edges with a weight equal to zero are not stored in a profile.

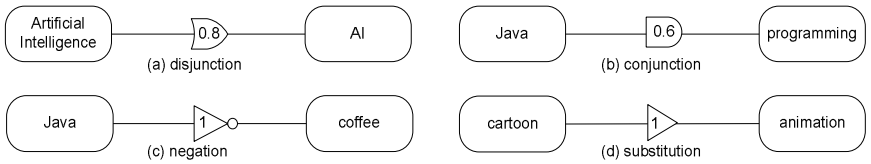


Fig. 2. Examples of edge types between terms

An example user profile is depicted in the graph of Fig. 3. Note that this may be a disconnected graph with components corresponding to unrelated user information needs. For example, in Fig. 3, user interests include background images and Java programming which are unrelated to each other. On the other hand, a connected graph component may capture more than one user need. Based on the profile of Fig. 3, user needs include Java programming as well as database systems, which are connected.

In general, for a pair of nodes t_i and t_j, there may be *at most* one edge $t_i \rightarrow t_j$ whose type would capture the rewriting that best reflects the typical conception of t_i with respect to t_j by the user. On the other hand, both $t_i \rightarrow t_j$ and $t_j \rightarrow t_i$ may exist.

For example, Fig. 3 depicts that "Java" is connected to "programming" and "programming" is connected to "Java". The weights of these rewritings, however, are not the same, since "programming" is also connected to "C".

In addition to immediate term rewriting expressed by an explicit directed edge $t_i \rightarrow t_j$, term rewriting may also be defined transitively by a set of adjacent edges connecting t_i to t_j through intermediary nodes in the profile graph. In that case, t_i may be rewritten using t_j by successively applying the rewritings expressed in the edges. The weight of a transitive rewriting is calculated as a function of the weights of the edges on the corresponding path. Specifically, if D_N is the set of weights along the path, then the transitive weight is expressed as a function $f_T(D_N)$. In principle, one may conceive of different functions that may play this role. Based on human intuition and cognitive evidence [13], these should satisfy the following basic condition:

$$f_T(D_N) \leq \min(D_N) \quad \textbf{(1)}$$

In our prototype, we used multiplication of weights for f_T. For example, for the profile shown in Fig. 3, the weight of the transitive rewriting of "Java" using "database systems" is $0.9*0.5=0.45$. Another possible function is the minimum of weights. For the same example, the corresponding transitive weight would be 0.5.

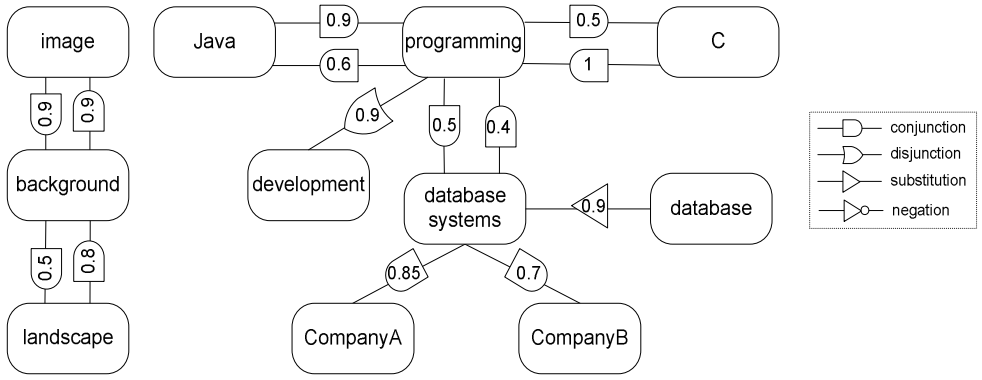


Fig. 3. Example user profile

It should be pointed out that not all graphs with the above characteristics map to valid profiles. For example, between two terms there cannot be substitution edges in both directions. In addition, when constructing a profile, it must be determined which terms are connected with edges and which terms are connected through other terms.

A separate profile as described above must be kept for each individual user of a system. Moreover, its overall design is quite general, so it can capture in exactly the same way profiles of groups of users as well but this is out of the scope of this paper.

4 Query Disambiguation

Based on the above, disambiguation and personalization of a user search are viewed as a unified query modification process, consisting of term-rewriting operations dictated by the user profile. An algorithm for this purpose is presented in Fig. 4, and is called *QDP* (*Query Disambiguation and Personalization*). The algorithm takes as

inputs a user query Q, a user profile U and a criterion CXT and produces a single modified query Q′. The criterion CXT establishes the *query context*, i.e. it specifies which terms of U, connected (directly or transitively) to terms included in Q, influence the original user query and, should therefore be considered in query modification. Examples of possible criteria are the following:

- The transitive weight of any path connecting a new term t_j to the query Q should be greater than a threshold T.
- The number of edges of any path connecting a new term t_j to the query Q should be less than a threshold T.

For example, for the query "database systems" and the profile of Fig. 3, consider the first criterion with a threshold T=0.6. Then, the terms "CompanyA" and "CompanyB" are the only ones considered within the query context. Criteria may be manually specified by an expert or automatically configured by the system.

The algorithm is based on a best-first traversal of the graph representing the user profile. The basic idea is to gradually modify the query by considering, in each round, terms from the profile that are connected to those already in the query. The algorithm stops when there are no terms in the context of the original query to be used for further query refinement.

QDP algorithm

Input: Query Q, User Profile U, Query-Context Criterion CXT
/*CXT(t_j,Q)=TRUE then t_j is in the context of Q */
Output: Modified query Q′

Begin
Q′=Q
Pick ($t_i \rightarrow t_j$)∈U s.t.:
t_i∈Q′, t_j∉Q′, $t_i \rightarrow t_j$ has max. weight among rewritings of Q′
While CXT(t_j,Q)=TRUE
Replace t_i∈Q′ with the result of rewriting ($t_i \rightarrow t_j$)
Pick ($t_i \rightarrow t_j$)∈U s.t.:
t_i∈Q′ and $t_i \rightarrow t_j$ has max. weight among rewritings of Q′
End

Fig. 4. Outline of *QDP* algorithm for personalization

More specifically, the original query is mapped to the user graph. Any query term not mapped to a node in the profile is not affected by the algorithm. In each round, a term t_i from the current version of Q′ is selected provided that there is an edge $t_i \rightarrow t_j$ stored in the profile with the greatest weight among all possible rewritings of any term in Q′ such that t_j is within the context of the original query according to criterion CXT and is not already included in Q′. Query modification stops when there are no terms in the context of the original query that can be integrated into it. Fig. 5 illustrates successive transformations of a query for the profile illustrated in Fig. 3 using this algorithm with a threshold T=0.8. A slightly different version of *QDP* is employed if there are more than one possible rewritings of a specific term, such as "Java" connecting to "programming" and "island". In this case, the algorithm may return a set of queries. Depending on the personalization philosophy adopted by the system, the user may be provided with this set in order to choose which one better

represents their current need, or, with the top-ranking results returned by each query. Details are omitted due to space considerations.

Fig. 5. Execution of QDP using a query "Java", the example profile and a threshold T=0.8.

5 Implementation

In this section, we provide an overview of the prototype system that we have built following the architecture of Fig. 1 and the proposed framework. We have built this system on top of an existing search engine (Google).

Search Interface. We have used the Google Web API service, a beta web program that enables developers to find and manipulate information on the web [18]. A personalized query, output by *QDP*, is translated into the appropriate syntax format expected by Google based on certain rules. For example, search for complete phrases is performed by enclosing them in quotation marks. A word is excluded from the search (due to a NOT operator dictated by the profile) by placing a minus sign immediately in front of it. Results by Google are displayed to the user. We are implementing a module that re-ranks results based on term weights stored in the user profile.

User Profiling. We implemented a simple incremental algorithm for construction of profiles based on the user model presented with the purpose of evaluating the potential of the user model and of the query modification process. We only sketch it here. It builds profiles with no negations based on user feedback and one-keyword queries.

When a user is presented with the results of a search, he may mark relevant documents. These are used as input to the profile construction algorithm which builds a sub-graph whose structure depends on the documents input as well as the number of them. The algorithm proceeds in three steps. First, document analysis is applied. Each document is mapped to a list of {word, number of word occurrences} pairs. Then, these lists and the initial query are used to construct a sub-graph. Initially, this sub-graph consists of one node corresponding to the query term. The algorithm gradually builds the sub-graph by adding nodes mapping (groups of) words encountered in the input document lists. Two or more words are treated as one group if the algorithm cannot decide on the number and type of edges to be used to connect them if these words are mapped to separate nodes. Rules for creation of edges include:

The query term and a group of words co-exist in a given list ⇒
A conjunction edge is created between the respective nodes

The query term does not exist in a given list ⇒
A disjunction edge is created between the query node and the node(s) mapping the words of this list.

The query term does not exist in any given list ⇒
A substitution edge is created between its respective node and the node(s)

mapping the words of the lists.

At the end of this step, nodes in the sub-graph represent terms and edges represent term rewritings. Finally, the algorithm merges the sub-graph produced in the previous step into the user's profile and assigns node and edge weights. If the user profile is empty, or has no similar nodes with the sub-graph, then the latter is simply added as a disconnected graph into the former. Similar nodes are those sharing common words, such as nodes $\{t_1\}$ and $\{t_1, t_2\}$. If the profile and the sub-graph have similar nodes, then they are merged. Merging preserves finer-grained structure which may already exist in the profile or dictated by the connections in the sub-graph. Each node of the profile that is new or has been affected by the merging is assigned the minimum of weights of the words mapped to it. The weight w_t of a word is given by the formula:

$$w_t = f_t * (N_t / N) \quad (2)$$

N_t: the number of past and current documents containing the word; N: the total number of documents of sets containing at least one document with this word; f_t: the number of the word's past and current occurrences to the total number of words of sets where the word exists. An edge is assigned a weight w_e with a value equal to:

$$w_e = (N_e / N') \quad (3)$$

N_e: the number of past and current documents containing associated words; N': the total number of documents of sets with at least one document including these words.

Experimental Results. We have conducted experiments with ten users. Since the underlying engine is Google, our dataset is the Web. Each user was assigned three search tasks, such as finding an appropriate car rental, identifying the latest in fashion, and so forth, as well as allowed to consider two tasks of their own. The goal of a search task was to find at least two interesting and relevant resources. Each user had two sessions with the system. During the first one, no modification of user queries took place. The user could resubmit a query until the goal of a task was satisfied. In addition, users submitted feedback by marking relevant documents. This information was used by the profiling method. During the second phase, which took place three weeks after the first one, personalization was activated using the profiles built during the first session. As the main measure of effectiveness of our approach we have used the task completion time. On the average, participants satisfied their needs significantly faster when personalization was in effect rather than when queries were executed in their original form. Specifically, the average gain in time was equal to 29%. In addition, there was an increase in the number of relevant documents found among the top 20 results returned by the search engine when personalization was applied. In some cases, when the initial user query had been extremely ambiguous, improvement was over 50% (i.e. over half of the top 20 results have been replaced by more relevant matches).

6 Conclusions and Future Work

In this paper, we have described a high-level architecture of a system for personalized searches. We have described a term-based graph-form user profile that allows thinking

of query disambiguation and personalization as a unified term rewriting process. It interprets connections between terms as possible rewritings between them. Based on this kind of profile, we have provided a query-rewriting algorithm for query disambiguation and personalization. We have described a prototype system and presented promising results regarding the potential of the proposed framework.

We are currently elaborating the prototype system. We are working on the ranking module, and on a more elaborate user profiling algorithm capable of handling negations and initial queries with more than one term. We are also interested in developing a mechanism for user profile validation and in more extensive experiments.

References

1 Belkin, N., Croft, W.B. (1992). Information Filtering and Information Retrieval: Two sides of the same coin?. *Communications of the ACM*, Vol.35, No.12. December 1992.

2 Bollacker, K., Lawrence, S., Giles, C. (1999). A System for Automatic Personalized Tracking of Scientific Literature on the Web. Int'l. Conf. on Digital Libraries, 105-113, 1999.

3 Bueno, D., David, A. (2001). METIORE: A Personalized Information Retrieval System. M. Bauer, P.J. Gmytrasiewicz, and J. Vassileva (Eds.): UM 2001, LNAI 2109, 168–177, 2001.

4 Callan, J. (1996). Document Filtering with Inference Networks. ACM SIGIR Conf., 262-269, 1996.

5 Cetintemel, U., Franklin, M., Giles, C. (2000). Self-Adaptive User Profiles for Large-Scale Data Delivery. Intl. Conf. on Data Engineering, 2000.

6 Glover, E. J., Lawrence, S., Birmingham, W. P., Giles, C. (1999). Architecture of a Metasearch Engine that Supports User Information Needs. Int'l. Conf. on Information and Knowledge Management, 1999.

7 Herlocker, J., Konstan, J., Borchers, A., Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. Intl. Conf. on Research and Development in IR, 230 - 237, 1999.

8 Koutrika, G., Ioannidis, Y. (2004). Personalization of Queries in Database systems. Intl. Conf. on Data Engineering, 597-608, 2004.

9 Liu, F., Yu, C., Meng, W. (2002). Personalized Web Search by Mapping User Queries to Categories. Intl. Conf. on Information and Knowledge Management, 2002.

10 Pitkow, J., Schutze, H., Cass, T., Cooley, R., TurnBull, D., Edmonds, A., Adar, E., Breuel, T. (2002). Personalized Search. *Communications of the ACM*, Vol. 45. No.9, Sep. 2002.

11 Qiu, Y., Frei, H.P. (1993). Concept-based Query Expansion. Intl. Conf. on Research and Development in IR, 160-169, 1993.

12 Salton, G., McGill, M.J. (1983). Introduction to Modern IR. McGraw-Hill, NY, 1983.

13 Collins, A., Quillian, M. (1969). Retrieval Time from Semantic Memory. *J. of Verbal Learning and Verbal Behaviour*, Vol 8, 240-247, 1969.

14 Smyth, B., Bradley, K., Rafter, R. (2002). Personalization Techniques for Online Recruitment Services. *Communications of the ACM*, Vol. 45, No. 5, 39-40, May 2002.

15 Sparck Jones, K., Jackson, D. M. (1976). The use of automatically-obtained keyword classifications for Information Retrieval. Information Processing and Management, 1976.

16 Tanudjaja, F., Mui, L. (2002). Persona: A Contextualized and Personalized Web Search. Proc. of the 35th Hawaii Int'l Conf. on System Sciences, 2002.

17 Yan, T., Garcia-Molina, H. (1995). SIFT: A Tool for Wide Area Information Dissemination. USENIX Technical Conference, 177-186, 1995.

18 Web API for Google. http://www.google.com/apis/api_faq.html