Personalization through query explanation and document adaptation

Anthony Ventresque, Sylvie Cazalens, Thomas Cerqueus, Philippe Lamarre, Gabriella Pasi

Personalization through Query Explanation and Document Adaptation

Anthony Ventresque[1], Sylvie Cazalens[2], Thomas Cerqueus[2], Philippe Lamarre[2] and Gabriella Pasi[3]
[1]SCE, Nanyang Technological University, Singapore
[2]LINA, Université de Nantes, France
[3]DiSCo, Università di Milano Bicocca, Italy
[1]aventresque@ntu.edu.sg, [2]FirstName.LastName@univ-nantes.fr, [3]pasi@disco.unimib.it

ABSTRACT

We present a new formal approach to retrieval personalization which emcompasses a query personalization process at the user's side with a light document adaptation at the information server's side. Our solution relies on the use of a domain ontology: queries and documents are in fact indexed by sets of concepts. For each concept of the query, the query personalization process allows to express the importance of linked concepts, which may vary according to the search context. Each query concept can be "clarified" by this process; although the proposed method clarifies only central query concepts. The initial query as well as its defined clarifications are sent to the server. Then, the server reconsiders its document representations based on both the query and the concepts clarifications it received. The proposed solution does not require that the information server maintains any user profile, and can be useful when, for privacy concerns, it is committed not to profiling the users.

Keywords

Query Explanation, Document Adaptation, Similarity and Propagation, Semantic Vector Space

1. INTRODUCTION

Personalization is nowadays an important issue for many data and information retrieval applications, aimed at enhancing the user experience and business profits. Coping with a huge and increasing amount of data accessible from the Web, retrieval systems need to display not only relevant information to a specific query but also information that match specific users needs, interests, preferences. And this is seen as an important marketing tool and a requirement for many electronic businesses.

Most of the personalization models are based on two important and complementary aspects: (i) implicit or explicit collection and consequent representation of user's behavior, preferences, interests; (ii) leveraging that knowledge during the retrieval process. This is mainly done by expanding queries [10], by re-ranking search results or by re-indexing documents [2]. Some of these tasks, like expanding queries, can be achieved either at the user's side or at the information providers side (i.e. server's side). This paper presents a personalized information retrieval approach which does not assume any user profiling by the information server. Our model favors (semi-)automatic query explanation at the user's side. Then the server reconsiders its document representations in the light of both the query and the explanation it received, thus enhancing the query evaluation process by specifically adapting it to the user. This approach can be useful when the server privacy policy commits it to not profiling the user.

In parallel to an ever increasing amount of information, these last ten years have witnessed a huge work concerning semantics, in particular with the definition of many domain ontologies (like in medecine, biology, almost any domain) and linked technologies. While a specific language, namely SPARQL, has been designed to query triple stores, other approaches consider using the concepts of the ontology to represent both data (let it be semi-structured or unstructured) and queries, thus considering search at a more semantic level. Our approach relies on the usage of a domain ontology, with queries and documents indexed by its set of concepts. We represent both queries and documents by semantic vectors [1]. Each concept is weighted according to its representativeness in the document (respectively the query). As in the classical vector space model, relevance is then modeled as a distance between the two vectors.

This paper does not focus on the indexing, which can be achieved (semi-)automatically by several indexers. Then, given a user query (user's side) and document vectors (server's side), the objectives of our approach are to define (i) a query explanation process and (ii) a light weight adaptation process which adapts the document representations to the query (but does not require re-indexing) and thus conceptually enhancing the query evaluation process. Both objectives are linked, as the adaptation process uses the query explanation. The proposed solution relies on several assumptions and choices.

First, each weighted concept of the query is explained separately. This seems more precise to us than a classical query expansion, which may lead to an unbalance of the original components of the query [10].

VLDB '10, September 13-17, 2010, Singapore

Second, the explanation of a given concept considers two notions : given the user's domain ontology, we assume that a similarity function on the set of concepts specifies to which extent a given concept is similar to another one. This is part of the user's modeling of the domain. However, to our view, similarity is not enough to express the importance of a linked concept in a query. Indeed, two different search contexts may require to give more or less importance (interest) to a same similar concept. We formalize this intuition by introducing the concept of propagation function. For each concept, it expresses the importance in the search of concepts with given similarity values. Both the similarity and importance values are computed automatically, but the user can keep control on the process. Given that importance propagation may vary depending on the search context, the user should have a toolbox with several propagation functions which she can choose or which are automatically proposed depending on the context (there may be a profiling module at the user's side which helps).

Finally, both the initial query and the concept explanations (which are vectors) are sent to the information provider, which has to evaluate the relevance of its documents. As explained before, our choice is to avoid re-indexing. Then, given the document representations (i.e. vectors), there are two options : (i) comparing each concept explanation with each document, and then aggregate the results to get a global relevance measure or (ii) computing an adapted document representation (without changing the initial one) which characterizes better each document with respect to the search needs expressed by the concept explanations. We have chosen the second option, in which, in the end, the relevance computation considers the initial query and the document adaptations.

Our contributions are: (i) a new formal approach to personalization which encompasses a query personalization process from users together with a light document adaptation; (ii) our solution does not require that the document provider maintains user profile; (iii) our solution is non-intrusive for existing systems, i.e. it can be plugged in systems without need of document reindexing, query reformulation nor special relevance function.

In the remaining of this paper, we first present a motivating example which shows that questions "how concepts are similar?" and "how much of them are interesting and to what extent?" are crucial for personalized retrieval; this is the core of our solution, and hence we present its architecture (Section 2). Then we formally define query personalization (Section 3) and document adaptation (Section 4). After a discussion on main assumptions of our work (namely ontological heterogeneity and similarity and propagation functions) in Section 6, we conclude (Section 7).

2. MOTIVATING EXAMPLE AND SYSTEM ARCHITECTURE

While selecting query terms that are fully compatible to documents' providers terms (index terms) is in itself a difficult problem, a same term could also have slightly different meanings to different users (term ambiguity).For instance let us assume three users Alice, Bob and Chikako, willing to adopt a dog. Their request, e.g. "I would like a dog", is very straightforward, and they can send it to an animal welfare organization nearby their living place. Unfortunately, there are a lot of sheltered dogs in these organizations, and scanning their data base could be tedious. Hence, it should be useful to specify which kind of dog each people intend to adopt. On the other hand, if they want a pedigree dog, they could be disappointed when using the keyword "dog". Indeed, dog breeders are canine specialists, and their animals shall not be deemed to be "dog", but "labrador", "akita", or whatever. In both cases, a more accurate description of the user's preferences, i.e. the intended objects should be useful to specify or to expand queries.

Let's Alice and Bob more likely consider as a dog prototype the labrador, while Chikako's dog prototype is represented by akitas (see Figure 1). Even if the concept *dog* has the same meaning in their mind, the descendant (more specific) concepts of dogs are not all similar, being some are more relevant than others. The consequence is that when querying the animal welfare organization data base, they don't look for the exactly same items. Then, a solution to improve the results expected from the evaluation of their queries should take into account the users' prototypal concepts and their similar concepts. This means to consider a central concept (the prototype) and to formalise a *concept similarity function*; the user will finally decide which items in the ranked list will be relevant to her/him.Thus, while labradors are the prototype "dogs" to Alice and Bob, it may happen that dalmatians and akita are still acceptable to Alice while not to Bob. Likewise, Chikako considers that dalmatians and labradors, even if they are less relevant than akita, are still ok. We call this combination of proximity to the central concept and interest values a *propagation function*. In Figure 1 the interest values that Alice, Bob and Chikako give to the concepts according to their similarity to *dog*.

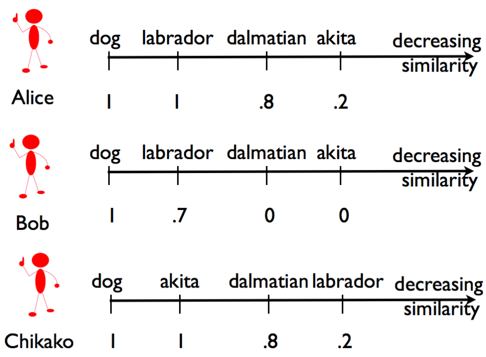


Figure 1: Alice, Bob and Chikako's similarity ranking of concepts and the propagation of their interest, both w.r.t. concept *dog*.

The propagation function aims at describing a dimension of a query, i.e. one of its main concept. As each user manages its own propagation (own similarity and own interest values), we call such a description a *personalized dimension* of the query. Once the query is personalized, our solution is to keep the query unchanged, but to transform the document representations according to the personalized dimensions. It brings us to *adapted documents*. Finally, the adapted documents are compared to the initial query. Architecture of our system is composed of five modules (Figure 2), over both user and document provider. Actually, the basic modules (white) are already provided by current systems. You can see two *semantic indexing* modules on both user's and document provider's side; these modules represent the queries and documents based on the representation model of the IR system (in our case: semantic vectors). Every system has also a *relevance computation* module (matching module), which ranks documents according to their relevance to the query (cosine). We add to this classical architecture three new modules (grey). On the user's side, the *query personalization module* explains the central concepts of the query, according to user's similarity and propagation functions. We see below in this paper how a user could obtain these functions (see Section 6.2). The descriptions are then given to the *data adaptation* module, which transform the document representation w.r.t. them.

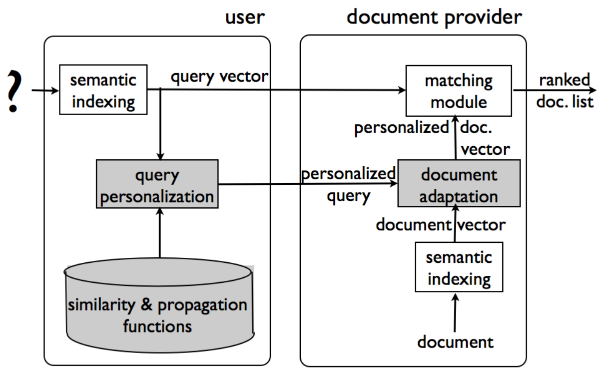


Figure 2: System architecture. All new grey modules are included in a classical semantic retrieval system.

3. QUERY PERSONALIZATION

In this section, after a description of the semantic vectors, we present a formalization of the propagation of users' interests, which constitutes the main process of query personalization. We provide some inputs on similarity and propagation functions later in this paper (see Section 6.2).

3.1 Semantic Vectors

In the vector space model [1], both queries and documents are represented as vectors of keywords (terms). If there are n keywords, each query or document is represented by a vector in the $n-$dimensional space. Relevance of a document to a query can then be calculated by measuring the proximity of the two vectors. An approach based on *semantic vectors* [16] uses the same kind of multi-dimensional linear space except that it no longer considers as dimensions the keywords, but *concepts* belonging to a considered ontology: the content of each query (respectively document) is represented by a semantic vector according to each concept.

We consider a very general definition of ontology [6]: it is a set of concepts together with a set of relations between those concepts. The only assumption we make is to be able to compute a similarity between concepts of an ontology, whatever the relations used. In the rest of the paper, we assume the existence of an ontology Ω, $\mathcal{C}_\Omega$ being its set of concepts. Then, a simple formal definition of a semantic vector can be the following:

Definition 1 (Semantic Vector). *A semantic vector $\overrightarrow{v_\Omega}$ is an application defined on the set of concepts $\mathcal{C}_\Omega$ of the ontology:*

$$\forall c \in \mathcal{C}_\Omega, \overrightarrow{v_\Omega} : c \rightarrow [0,1]$$

Reference to the ontology will be omitted whenever there is no ambiguity.

3.2 Propagation of Interest

Conceptual similarity is a function centered on a concept: it gives a value to every concept according to its similarity to the central concept.

Definition 2 (Similarity Function). *Let c be a concept of $\mathcal{C}_\Omega$. sim_c: $\mathcal{C}_\Omega \rightarrow [0,1]$, is a similarity function iff $sim_c(c) = 1$ and $0 \leq sim_c(c_j) \leq 1$ for all $c_j \neq c$ in $\mathcal{C}_\Omega$.*

Given a similarity function and a central concept c, we define a *propagation function* as a function which describes the importance of every concept according to c. We assume this function is monotonically decreasing.

Definition 3 (Propagation function). *Let c be a concept of $\mathcal{C}_\Omega$; and let sim_c be a similarity function. A function $\mathcal{P}f_c$:*
$$\begin{array}{ccc} [0,1] & \mapsto & [0,1] \\ sim_c(c') & \rightarrow & \mathcal{P}f_c(sim_c(c')) \end{array}$$
is a propagation function from c *iff $\mathcal{P}f_c(sim_c(c)) = 1$, and $\forall c_k, c_l \in \mathcal{C}_\Omega$ $sim_c(c_k) \leq sim_c(c_l) \Rightarrow$*
$$\mathcal{P}f_c(sim_c(c_k)) \leq \mathcal{P}f_c(sim_c(c_l))$$

We have suggested some propagation functions in [14]. They are inspired by membership functions used in fuzzy logic, i.e. the most similar concepts are given the value 1, the most dissimilar are weighted with 0, and in between, concepts receive a value according to their similarity:

$$\mathcal{P}f_c(x) = f_{l_1,l_2}(x) = \left| \begin{array}{ll} 1 & if\ x \geq l_1 \\ \frac{1}{l_1-l_2}x + \frac{l_2}{l_1-l_2} & if\ l_1 > x > l_2 \\ 0 & if\ l_2 \geq x \end{array} \right.$$

3.3 Semantic Personalized Query

As we said in the introduction we do not expand (by propagation) in a single new vector the weights of the central concepts of the query. Moreover, each central concept c of a query $\overrightarrow{q}$ is personalized in a separate vector. Thus a personalized dimension $\overrightarrow{persD}_c$ is a semantic vector which records the propagation of one concept only. We use $\mathcal{C}_{\overrightarrow{q}} = \{c : c\ is\ a\ central\ concept\}$; a central concept is an important concept, e.g. any weighted concept, a concept weighted with a greater value than a threshold, etc.

Definition 4 (personalized dimension). *Let $\overrightarrow{q}$ be a query vector and let c be a concept in $\mathcal{C}_{\overrightarrow{q}}$. A semantic vector $\overrightarrow{persD}_c$ is a* personalized dimension (persD), *iff $\exists \mathcal{P}f_c \forall c' \in \mathcal{C}_\Omega, \overrightarrow{persD}_c[c'] = \mathcal{P}f_c(sim_c(c'))$ and $\overrightarrow{persD}_c[c] =$ 1.*

The mathematical expression ending the definition means that no matter how the personalized dimension is obtained the only restriction is that no concept can have a greater weight than c. And most importantly, the value of the central concept c in a $\overrightarrow{persD}_c$ is *always* 1, and not the original value, because a personalized dimension is an explanation of a dimension of the query. The original query, and then the original values of the central concepts, are kept for the matching process.

A personalized query is the set of personalized dimensions, one for each central concept of a query: $persQ_{\overrightarrow{q}} = \{\overrightarrow{persD}_c : c \in \mathcal{C}_{\overrightarrow{q}}\}$. Figure 3 shows the personalization of a query $\overrightarrow{q}$ with two weighted concepts c_4 and c_7.

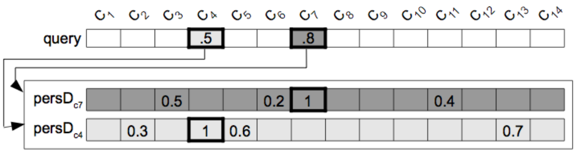


Figure 3: A personalized query composed of 2 personalized dimensions.

4. DOCUMENT ADAPTATION

Once user has explained central concepts of his/her query $\overrightarrow{q}$, s/he sends the query personalized to the document provider, who adapts his/her documents to the $persQ_{\overrightarrow{q}}$. Adaptation is not a reindexing of the documents, like with Bordogna and Pasi [2] for instance. It is a lightweight filter of documents through what the query explains as important in its personalized dimensions. If a concept c_i is relevant for $\overrightarrow{persD}_{c_j}$, then every document with this concept c_i should give that information in its adaptation. In fact, for any $\overrightarrow{persD}_{c_j}$, documents retain a unique value in the adaptation vector for concept c_j, which is the best correlation between personalization value of $c_i \in \overrightarrow{persD}_{c_j}$ and value $\overrightarrow{d}[c_i]$. While all concepts involved in some personalized dimension are already captured by this process, their values are then null in the adaptation. Other concepts, not relevant for any personalized dimension, keep their original value, as they show some dimensions of the document not relevant for the personalized query. Indeed, the norm of the vector gets higher (and consequently, its relevance lower). For example, this is the case for concepts c_1 and c_9 in Figure 4.

Algorithm 1 details the computation of the personalization of document representation $\overrightarrow{d}$.This algorithm ensures that all the central concepts of the initial query vector are also weighted in the personalized document representation as far as it is related to them. With respect to the query, the personalization of the document representation is more accurate because it somewhat enforces the characterization of the document over each dimension of the query.

The example of Figure 4 illustrates the computation of a persR. Each persD of the personalized query is combined with the semantic vector of the document. Let us consider $\overrightarrow{persD}_{c_4}$. In the document, the weight of c_4 is null. However, the personalized dimension related to c_4 weights other concepts. In particular, we have $\overrightarrow{persD}_{c_4}[c_2] = 0.3$. As $\overrightarrow{d}[c_2]= 1$, the resulting product is 0.3. This value improves $\overrightarrow{d}[c_4]$ (which is null), so we keep it in the adaptation of the document representation. Hence, in the $persR$, we can express that the document is related to concept c_4 of the query, even if it wasn't the case initially. Likewise, three concepts of the document (c_6, c_7 and c_{11}) are important to $\overrightarrow{persD}_{c_7}$, and the adaptation retains only one value for

Algorithm 1: Adaptation of document representation wrt. a query.

input : a semantic vector $\overrightarrow{d}$ and a personalized query $persQ_{\overrightarrow{q}}$ on an ontology Ω
output: a semantic vector $persR_{\overrightarrow{d}}^{\overrightarrow{q}}$.
begin
  forall $c \in \mathcal{C}\overrightarrow{q}$ **do**
    forall $c' : \overrightarrow{persD}_c[c'] \neq 0$ **do**
      $persR_{\overrightarrow{d}}^{\overrightarrow{q}}[c] \leftarrow \max(\overrightarrow{d}[c'] \times \overrightarrow{persD}_c[c'], persR_{\overrightarrow{d}}^{\overrightarrow{q}}[c])$;
  forall $c \notin \mathcal{C}_{\overrightarrow{q}}$ **do**
    if $\exists c' \in \mathcal{C}_{\overrightarrow{q}} : \overrightarrow{persD}_{c'}[c] \neq 0$ **then**
      $persR_{\overrightarrow{d}}^{\overrightarrow{q}}[c] \leftarrow 0$
    else
      $persR_{\overrightarrow{d}}^{\overrightarrow{q}}[c] \leftarrow \overrightarrow{d}[c]$;
  return $\overrightarrow{i}_d$;
end

$\overrightarrow{persD}_{c_7}[c_7] = 0.6$. Note that while a classical expanded query would have given one single value from central concept c_4 (but possibly on c_2 instead of c_4), expansion should have weighted 3 concepts from c_7 (c_6, c_7 and c_{11}). Our solution does not add as much noise as it could with classical expansion. Concepts c_1 and c_9 eventually keep their original value in persR of document $\vec{d}$ because they are not involved in any persD.

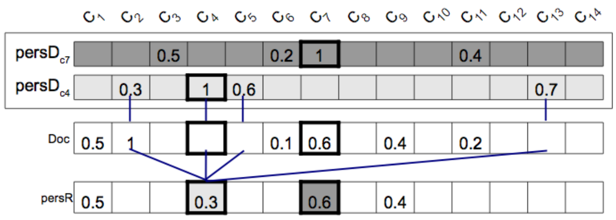


Figure 4: Obtaining the adapted document representation.

5. EXPERIMENTS

Our goal is to validate our approach in several steps: first step is cost analysis which enables to quantify the additional costs induced by the method. Second one is just to verify that given a query and different search contexts, the users get different results; this is can be viewed as a minimum requirement to get personnalized results. Finally, the method should be faced with a significant number of users who would estimate whether (or to which extent) they get personalized results. As we currently judge our number of users not significant enough, the paper focuses on the first two steps.

Complexity of our solution relies on the complexity of similarity and propagation functions, which together define query personalization, and document adaptation. There exist a lot of similarity measures (see Section 6.2) but they always consist on two nested loops. Let n be the number of concepts in the ontology, then similarity computation is in $O(n^2)$. Propagation gives a weight to every concept; and there are as many propagation functions as there are central concepts. Assume m the number of central concepts, then we need $O(m \times n)$ to compute propagation. As m is generally very small compared to n, query personalization has a complexity of $O(n^2 + n)$. However, these two steps can be computed before and cached; so there is not always needed to compute them whenever user queries the system. Adaptation mainly consists of a loop on every concept of the ontology for every document and persD. Then, for every central concept and every document, values are inserted at indices of central concepts. Finally, a loop is processed on concepts to put the values of concepts not involved in any persD. If d is the number of documents, we have an adaptation computation in $O(d(2(m \times n) + m \times log(n) + n))$. But this can be strongly reduced by using proper data model: big vectors (as many indices as there are concepts in the ontology) are useless while documents and queries are not expressed on all the concepts but a very small subset, etc. The worst case is not realistic and we assume a fast computation, by replacing at least n by n' which is drastically smaller.

Our experiments use the Cranfield corpus and WordNet (considered as a "lightweight" ontology) to index the documents. We developped a prototype software called Mysins [15] with a service oriented architecture. In Mysins search can be personalized by choosing the similarity and propagation functions. Later versions will have a more friendly interface, with automatic or semi-automatic personalized explanations (which can be obtained through profiling for example). The server side of Mysins runs the document adaptation module. We ran the 225 queries of the corpus. The number of retrieved documents is 50 (among the 1400 of the corpus). This later assumption seems reasonable as several user behavior analysis show that users generally consult the first result pages only. We use two different similarity measures: Wu and Palmer [17] noted sim^1, and a modified version of sim^1 (which permutes values of three closest groups of concept) as sim^2. Likewise we use two different propagation functions: $prop^1 = f_{0.95,0.9}$ and $prop^2 = f_{0.8,0.6}$.

In order to compare two sets of retrieved documents with their relevance value, we consider two measures. First one (Jaccard coefficient) measures the similarity of the two sets of documents under consideration (without considering their relevance value). It is defined as the number of documents in the intersection divided by the number of documents in the union the two sets. Second one takes into account the order in the ranking of retrieved documents. We have chosen a modified version of Rank Distance (RD) [4]: each document is given a value according to its position in the top-50, 50 for the 1st, 49 for the 2nd, etc. and 0 for the 51st onwards. Value of each document in first list is then compared to its value in second list. This measure gives of course more importance to permutations on top of the list of retrieved documents. You can see in Figure 5 (a), (b) and (c) the results for $\langle(sim^1, prop^1), (sim^1, prop^2)\rangle$, $\langle(sim^1, prop^1), (sim^2, prop^1)\rangle$ and $\langle(sim^1, prop^1), (sim^2, prop^2)\rangle$ respectively. Every dot corresponds to the comparison of answer lists of a query, using the given parameters. X-axis shows Jaccard measure and y-axis the home-made RD.

It is first worth noticing that dots are not close to (0,0), which means that there are differences between the two results sets. Most dots have Jaccard values between 0.05 and 0.4, while their RD values are between 0.1 and 0.6. Results sets are different (not the same collection of documents) and their ranking are even more different. Propagation eventually seems more important than similarity, because Figures 5 (a) and (c) have more scattered dots, with higher dissimilarity and/or disorder values in average.

This section has proven that: (i) additional cost of our solution is limited and (ii) in different contexts the results sets are different and show a personalization of the retrieval. Future work intends to validate the approach with "real" users.

6. DISCUSSION AND RELATED WORK

In this section, we first position our assumptions and propositions w.r.t. related work. We then discuss how collection of similarity and propagation functions can be thought.

6.1 IR, Personalization and Ontologies

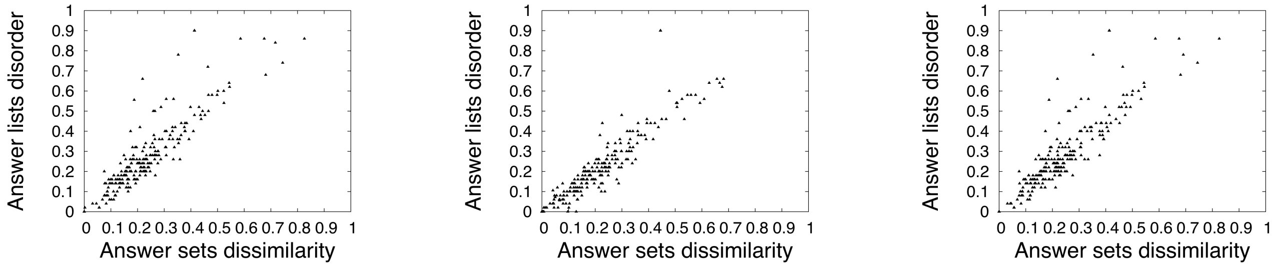
Context formalization for IR has focused a lot of attention in past few years [9]. Many work address this problem through the construction of a contextual space, collecting information on past queries, users clicks, etc. While most of them use terms to characterize the context, Mylonas et alii [9] propose to use an ontology. Their work is very interesting and can be compared to ours. But it does not use semantic vectors and our solution is more lightweight.

Query expansion has been seen promising to enhance small-size queries in order to help IR engines [3]. But while query expansion is a worthwhile contribution to IR, offering more relevant results, it often adds noise in the retrieval. So IR systems need to know when to use it [12]. Our solution do not use a query expansion, but a description of central concepts of the query through a propagation function on the concepts of the ontology. We have proven in [14] that our solution performs better than expansion in general case. And it is specifically more resistant to the use of many concepts.

Assuming a total agreement on ontologies on both sides is not realistic: an ontology is a conceptualization of knowledge upon the world, and we can hardly imagine a unique model of the world for every users. Alignment of ontologies, i.e. mappings between parts of the ontologies [5], are mostly used to address these problems. However these alignments are often incomplete: either because the process is time or resource consuming, or because users do not want to share all their conceptualizations, or because it is not always possible. While query or document indexing could be done on unshared parts of ontologies, it is useless. Indeed, every unshared element could not be understand, and hence no document would be relevant (cosine works only on common parts of queries and documents vectors). However, these unshared parts are meaningful for users and document providers and they are worthy of being used. We propose in [14] an *interpretation* process which let users and providers free to use their own ontologies during the information retrieval process. We are still working on an extension of the system described in this paper to heterogeneous context.

6.2 Similarity and Propagation

Similarity functions have been studied for a very long time [11, 13], etc. There exist a lot of different similarity functions, depending on the application and some desired properties. While most of them are context-independant, some



(a) (b) (c)

Figure 5: Comparisons of three pairs of parameters: same similarity and different propagation functions (a), different similarity and same propagation functions (b) and different similarity and different propagation functions, using Jaccard (x-axis) and home-made RD (y-axis).

takes it into account [7, 8]. Even if the problem of finding a personalized similarity function is not exactly addressed in these studies, we assume it could be done quite easily. For instance, we could imagine to collect the relative use frequency of sister concepts (e.g. *labrador* and *akita*) to give them different similarity values.

We do not address either the issue of propagation function personalization. It is a topic in itself and we focus here on the general process of personalization. However, we can imagine to first use a basic propagation, like we used in the worked mentioned before; then the system could collect feeedback from the user and change this basic propagation, according or not to some context. We would like to focus later on this issue.

7. CONCLUSION

Personalization of answering, content filtering, recommendation systems, etc. have been a topic of immense interest in recent times. While some solutions use a collection of user's behavior at providers' side or may substantially modify the retrieval system, our solution does not require that the information server maintains any user profile and is non-intrusive for retrieval systems. Moreover, we focus on a description of the query in order to watch documents in the light of its needs, and do not invent a new query formulation paradigm, or a reindexing of documents. Once users provide similarity and propagation functions, our system is lightweight and can be integrated in most information systems, assuming the system use semantic vector representations for queries and documents; then our solution can be used with documents, comments in blog, etc.

8. REFERENCES

[1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, 41(2):335–362, 1999.

[2] G. Bordogna and G. Pasi. Personalised indexing and retrieval of heterogeneous structured documents. *Information Retrieval*, 8(2):301–318, 2005.

[3] P.-A. Chirita, C. S. Firan, and W. Nedjl. Personalized query expansion for the web. In *SIGIR*, pages 7–14, New-York, 2007.

[4] L. P. Dinu. On the classification and aggregation of hierarchies with different constitutive elements. *Fundam. Inf.*, pages 39–50, 2002.

[5] J. Euzenat and P. Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg (DE), 2007.

[6] A. Gómez-Pérez, M. Fernández, and O. Corcho. *Ontological Engineering.* Springer-Verlag, London, 2004.

[7] V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.

[8] C. Keßler, M. Raubal, and K. Janowicz. The effect of context on semantic similarity measurement. In *OTM Workshops (2)*, pages 1274–1284, 2007.

[9] P. Mylonas, D. Vallet, P. Castells, M. Fernandez, and Y. Avrithis. Personalized informtion retrieval based on context and ontological knowledge. *The Knowledge Engineering Review*, 23(1):73–100, 2008.

[10] J.-Y. Nie and F. Jin. Integrating logical operators in query expansion invector space model. In *SIGIR workshop on Mathematical and Formal methods in Information Retrieval*, 2002.

[11] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[12] J. Teeman, S. T. Dumais, and D. J. Liebing. To personalize or not to personalize: Modeling queries with variations in user intent. In *SIGIR*, pages 163–170, Singapore, 2008.

[13] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.

[14] A. Ventresque, S. Cazalens, P. Lamarre, and P. Valduriez. Improving interoperability using query interpretation in semantic vector spaces. In *ESWC*, pages 539–553, 2008.

[15] A. Ventresque, T. Cerqueus, L.-A. Celton, G. Hervouet, D. Levin, P. Lamarre, and S. Cazalens. Mysins : make your semantic information system. In *EGC*, pages 629–630, 2010.

[16] W. Woods. Conceptual indexing: A better way to organize knowledge. Technical report, Sun Microsystems Laboratories, April 1997.

[17] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *ACL*, pages 133–138, Las Cruces, New Mexico, 1994.