Training a Personal Alert System for Research Information Recommendation

Germán Hurtado Martín [1,2] Chris Cornelis[2] Helga Naessens[1]

1.Dept. of Applied Engineering Sciences, University College Ghent, Ghent, Belgium
2.Dept. of Applied Mathematics and Computer Science, Ghent University, Ghent, Belgium
Email: german.hurtadomartin@{hogent.be,ugent.be}, chris.cornelis@ugent.be, helga.naessens@hogent.be

Abstract— *Information Systems, and in particular Current Research Information Systems (CRISs), are usually quite difficult to query when looking for specific information, due to the huge amounts of data they contain. To solve this problem, we propose to use a personal search agent that uses fuzzy and rough sets to inform the user about newly available information. Additionally, in order to automate the operation of our solution and to provide it with sufficient information, a document classification module is developed and tested. This module also generates fuzzy relations between research domains that are used by the agent during the mapping process.*

Keywords— Automatic classification, Current Research Information Systems, Fuzzy-rough sets, Personal search agents

1 Introduction

Current Research Information Systems (CRISs) are information systems, operating at regional, national, or international level[1], that store and provide access to data on current research carried out by organizations or researchers. These data typically include information on people, projects, organizations, funding possibilities, facilities, etc. CRISs are usually not easy to query, due to several reasons. On the one hand, the data volume is huge. This makes them a very interesting source of information, but can also be overwhelming for the user when he tries to find some specific information in the system. On the other hand, this information is frequently updated, so users are often not aware of new information that could be relevant to them. As a consequence, they are often unable to express their information needs by means of a conventional query. In the last years, personal search agents seem to be the solution to those problems [1]: they gather potentially useful information for users to subsequently present it to them in the form of recommendations. This idea has been received well and can be found not only in research-related applications like CRISs[2] or article recommenders [2, 3], but also in a significant amount of systems from different domains that share the aforementioned problems [4, 5].

However, and despite the fact that these agents can indeed be helpful for the users, they are subject to limitations. Since what they generally do is just to look for exact matches between the user's interests (given as keywords) and the rest of the information in the system, the user will often miss out on useful information as the documents use different terms to refer to the same, or similar, concepts. Furthermore, most CRISs also face other information defects such as missing, ambiguous, or imprecise information [6].

Concepts from fuzzy [7] and rough [8] set theory, upon which the present approach is based, appear as a solution to the problems mentioned above by allowing for a more flexible matching process. Fuzzy sets allow to express partial relationships, which describe reality in a more faithful way than a binary classification, while the rough component provides mechanisms for query expansion: in this way, a user profile and a document may still be matched when they refer to different, but related, keywords, resulting in a higher recall (a higher number of retrieved relevant documents).

To put these ideas into practice, a Personal Alert System (PAS) is currently under development. This system contains profiles of researchers and their interests, activities, papers, etc., as well as documents with information regarding research projects, conference announcements, or funding possibilities. Both these information sources are mapped to a common ontology, which is currently the three-level IWETO[3] taxonomy of the Flemish government. The main goal of the system is to alert users whenever a document can be matched to their research interests by the search agent. A basic prototype has been implemented, using fuzzy-rough algorithms, and in which the user can also influence term relations through a simple feedback process.

While a conceptual version of PAS was described in [9], this work goes further and describes its concrete implementation, paying special attention to the development and evaluation of an automatic classification module which allows the system to classify new documents according to the IWETO taxonomy, as well as to acquire new information with a view to the matching process.

The remainder of this paper is organized as follows: in Section 2, PAS is presented, giving a brief overview of its architecture, and focusing on how the information is represented in the system. In Section 3, the automatic document classification mechanism is described in detail, while in Section 4 we show how the mapper works. Finally, the paper is concluded in Section 5, where we address some issues for future work as well.

2 General design of the system

As said in the introduction, PAS contains information about researchers, projects, funding possibilities, etc., and it will try

[1]Some examples can be found at http://cris.csrees.usda.gov/ (USA), http://www.ris.is/ (Iceland), or http://sicris.izum.si/ (Slovenia). More information can be found at http://www.eurocris.org/, the professional association of CRIS experts and developer of the standard for these systems.

[2]The possibility of using an agent to retrieve funding possibilities is given at EraCareers: http://ec.europa.eu/euraxess/

[3]IWETO, Inventaris Wetenschappelijk en Technologisch Onderzoek Vlaanderen (now FRIS); http://www.iweto.be

ISBN: 978-989-95079-6-8

to match researchers with research information by using the "intelligence" that fuzzy and rough set theories provide. This section first gives a brief, general view on the system, and then focuses on the representation of the information.

2.1 Architecture of the system

Fig. 1 below shows the different modules of the system and how they interact with each other. In brief, the system is organized as follows. Both users (researchers) and documents (information about projects, etc.) are represented by means of profiles. Those profiles, described in more detail in the next subsection, contain several fields, the field storing keywords being the most important one. These keywords represent the researchers' interests (in the cases of the users), or the fields the documents are related to (in the case of the projects), and correspond to the IWETO taxonomy, where they are distributed in three hierarchical levels: a first level with 5 main nodes, a second level with 29 nodes, and a third level containing 359 nodes (in this last level, a node may contain more than one research field, when the fields are closely related). The left-hand side of Fig. 2 shows a sample of the taxonomy. Once a document comes into the system, it is automatically classified (this process will be shown in detail in Section 3), to subsequently be sent to the mapper. The mapper is the central and, thanks to its fuzzy-rough algorithms, intelligent part of the system. It decides whether or not a given document could be interesting for a given user. After this process, and if the mapper has decided so, a notification (about that given document) is sent to the user. Finally, if he desires so, the notified user can give his feedback about his satisfaction degree with the received alert: the system will then use this information to adjust the relations described in Section 2.2.

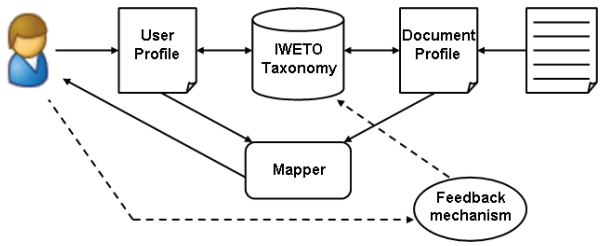


Figure 1: General architecture of PAS

For additional information about the architecture and a more detailed description of the different parts of the system, we refer to [9].

2.2 Representation of the information

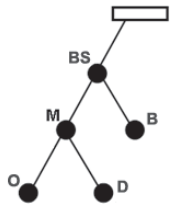
A key issue in PAS is the representation of the information: not only how profiles are represented but also how the IWETO taxonomy is stored.

Profiles are based on IWETO keywords, and their representation is common for both users and documents. Leaving aside the fields inherent to users or documents (i.e. "name" for users, "title" or "description" for documents), both user and document profiles contain a field dedicated to store the keywords that will represent them in the system. The structure used for that is a set where the pairs $<$*interest terms*, *degree of interest*$>$ are stored (*degree of relationship*, in the case of the documents). For example, the list $\{$*(AI,1)*, *(T,0.7)*, *(P,0.2)*$\}$ would correspond to a user very interested in *Artificial Intelligence*, quite interested in *Translation*, and slightly interested in *Physics*. The degree of interest/relation lies always between 0 (no interest/relation) and 1 (strong interest/relation). Every profile can therefore be seen as a fuzzy set in the whole collection of keywords X (since the keywords can come from any of the levels of the taxonomy, its hierarchical structure does not play any role at this point).

In the current implementation, the information contained in the profile keywords has different origins. While documents are automatically classified by using the techniques explained in Section 3, users must select their interests from a list of keywords. Users also select the different interest degrees from a list where these are linguistically represented (to subsequently map them onto a numerical value automatically).

Apart from user-keyword and document-keyword relations, we also consider relationships between keywords. A square matrix is used to represent the different relations between keywords, expressed as a degree ranging between 0 (no relation) and 1 (strong relationship). The right-hand side of Fig. 2 gives an example of this representation. A way of assigning degrees to term pairs is based on their co-occurrence during the training process; the more two terms co-occur in training documents, the higher their degree of relationship (see (10) in Section 4 for more detail).

These degrees make it possible to define a fuzzy relation R reflecting how closely two keywords k_1 and k_2 are related. It is important to remark that R is not a fuzzy tolerance relation, since it is not symmetric. It is reflexive, because a term is perfectly related to itself, so $R(k_1, k_1) = 1$, but $R(k_1, k_2)$ and $R(k_2, k_1)$ are not always equal. This is necessary to depict reality in a more faithful way. For instance, and referring to the example shown in Fig. 2, it is logical to think that there is a stronger relation from *Ophtalmology* towards *Medicine* than vice versa: a document related to *Ophtalmology* will always be related to *Medicine*, but a document related to *Medicine* will not be necessarily relevant to *Ophtalmology*.



	BS	M	B	O	D
BS	1	0.4	0.2	0.1	0.1
M	1	1	0.3	0.3	0.2
B	1	0.3	1	0.1	0
O	1	1	0.1	1	0.1
D	1	1	0	0.1	1

O: Ophtalmology *M: Medicine* *BS: Biomedical Sciences*
D: Dermatology *B: Biophysics*

Figure 2: Example of how the classification is stored

The information about the links between keywords (research fields) contained in the keyword relationship matrix can be used by the system, along with the interest research fields in the profiles, to compute a fuzzy-rough upper approximation for every profile. This process will be explained in more detail in Section 4.

3 Automatic document classification

Clearly, information is the core of any information system, CRISs included. Therefore, it is important for the system to

ISBN: 978-989-95079-6-8

be able to acquire new information easily. Since PAS aims to be as human-independent as possible, it is equipped with an automatic document classification module for this purpose. A representation of the process that documents undergo in this module is shown in Fig. 3 below. To build the module, some of the ideas in [10], such as the usage of keyword vectors for documents and classification, have been used.

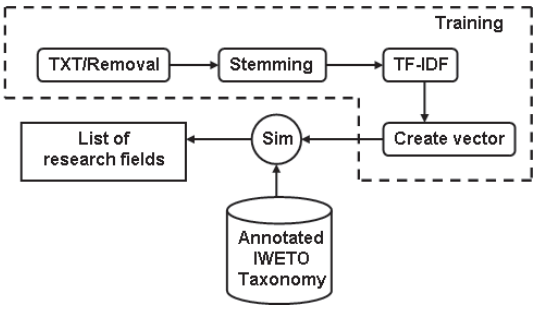


Figure 3: Automatic classification process

3.1 Training

First of all, the system needs to be trained, which is done by means of already classified documents. Currently, these documents come from the FRIS database[4]. Basically, each of them consists of an abstract about a given project, the keywords of that project, and the research fields under which the project was classified in the FRIS database. The documents must first be converted to a format readable by the system, namely txt. Afterwards, every document d is processed by a stemming algorithm. To this purpose, PAS currently uses a purpose-built stemmer, although the list of stopwords that this stemmer uses is an extended version of that which can be found at the site of the implementation of the Porter stemmer [11] in the Snowball language for stemming algorithms[5]. After that, the TF-IDF algorithm [12] is applied to calculate a weight w_i for each term in the document. Those terms with a weight w_i greater than a given threshold γ ($\gamma \geq 0$) are put into a vector $\vec{d}$ that represents the document. This vector $\vec{d}$ is further normalized so that it has unit length.

An important aspect at this point are the terms for which a TF-IDF value is computed. The first implementations of the classifier worked with two-word terms (bigrams), since such terms offer more information and are less ambiguous; for example "information system" is more meaningful than "information" and "system". However, they are also more difficult to match between documents, and some combinations are not so fortunate. So since early tests with one-word terms (unigrams) showed a sensible improvement, the classifier currently works with unigrams. Nevertheless, for some terms composed of more than 1 word, no splitting is applied. This happens for terms extracted via pattern recognition, for example those preceded by the string *"Keywords:"* in a document. In this cases, the terms are added, untouched, to the vector $\vec{d}$, with weight 1.0. Additionally, if the term is not a unigram, another copy of it is added to the rest of the text in the document to be split and processed as described above.

When the training documents have been reduced to their term vector form, the system is ready to start its learning process. It is here where the hierarchy of the IWETO thesaurus is used. First, and based on the information regarding the nodes (research fields) under which the documents have been classified, every vector $\vec{d}$ is assigned to its corresponding node or nodes n. A vector $\vec{n}$ for every node n is then computed. These vectors $\vec{n}$ are normalized vectors containing the terms that are most relevant to node n, or in other words, the terms (and respective weights) contained in the term vectors $\vec{d}$ assigned to n. More formally, let $S(n)$ be the set of subconcepts under concept n (children of node n). Also, let $\{d_1^n, d_2^n, \cdots, d_{k_n}^n\}$ be the individual training documents classified under concept n. $Docs(n)$, the set of all the documents classified under concept n augmented with the documents classified under all its children is defined as:

$$Docs(n) = \left[\bigcup_{n' \in S(n)} Docs(n') \right] \cup \{d_1^n, d_2^n, \cdots, d_{k_n}^n\} \quad (1)$$

The term vector $\vec{n}$ is then computed as:

$$\vec{n} = \left[\sum_{d \in Docs(n)} \vec{d} \right] / |Docs(n)| \quad (2)$$

This term vector $\vec{n}$ is finally normalized into a unit vector.

3.2 Automatic classification

The classification process starts in a similar way to that of the training process, including the application of the stemming and TF-IDF algorithms to create their representative vectors $\vec{d}$. Each test document is then compared to every node in the hierarchical classification. This is done by comparing their representative term vectors $\vec{d}$ and $\vec{n}$ by means of cosine similarity. Those nodes n for which the value of the comparison $sim(\vec{d}, \vec{n})$ is greater than zero are added to a priority queue along with their sim values, ordered with respect to these values. The cosine similarity measure sim for normalized vectors is defined as their inner product, i.e.,

$$sim(\vec{d}, \vec{n}) = \frac{\vec{d} \cdot \vec{n}}{|\vec{d}||\vec{n}|} = \vec{d} \cdot \vec{n} \quad (3)$$

Since the values $sim(\vec{d}, \vec{n})$ reflect how related the document d is to the class represented by node n, they are normalized so that d and n_1, the class with which d has the strongest relation, are related in a degree 1. Finally, the nodes in the N first places of the priority queue (i.e. the nodes with the N greatest values for sim, $N \geq 0$) are retrieved, along with their computed and normalized sim values, as long as they are greater or equal than a given threshold δ, $\delta \geq 0$. Thus, δ controls that only those classes to which d is sufficiently strongly related are taken into account.

The document is then classified under those research fields. This means that a profile is created for the document, with its representative keywords the research fields in which it has been classified, and with the respective sim values giving an idea of how related the document is to them.

[4]FRIS, Flanders Research Information Space; http://www.ewi-vlaanderen.be/fris/

[5]M. Porter, *Snowball: a language for stemming algorithms*; http://snowball.tartarus.org/index.php

ISBN: 978-989-95079-6-8

3.3 Classification results

The last part of this section is dedicated to show some results obtained with the proposed classifier. At the same time, some design decisions are discussed.

First of all, an evaluation measure is needed. Evaluating hierarchical classification is no trivial problem, a problem that gets yet more complicated by the fact of the classification being multi-label, i.e., documents can be labelled with different keywords. In this work we will use an adaptation to fuzzy sets of the measure hF_β proposed by Kiritchenko et al. in [13]. Specifically:

$$hF_\beta = \frac{(\beta^2+1)\cdot hP \cdot hR}{(\beta^2 \cdot hP + hR)}, \beta \in [0, +\infty) \quad (4)$$

where hP and hR are hierarchical precision and hierarchical recall, respectively. These measures are a variation of the traditional precision and recall evaluation measures which also take into account the hierarchical structure of the classes. In particular, rather than comparing the classes a document belongs to and the ones that were predicted for it, we also consider the ancestors of these classes. Formally:

$$\hat{C}_i = \{\cup_{c_k \in C_i} Anc(c_k)\} \qquad \hat{C}'_i = \{\cup_{c_l \in C'_i} Anc(c_l)\} \quad (5)$$

where C_i and C'_i are the actual and the predicted class sets, respectively, and $Anc(n)$ is the set of ancestors of n, n included. Therefore, $\hat{C}_i$ and $\hat{C}'_i$ are the actual and predicted class sets extended by adding them the ancestors of the classes they contain. Note that $\hat{C}'_i$ is treated as a fuzzy set, with its membership values equal to the predicted sim values, and that $\hat{C}_i$ is actually a crisp set, since a document d is related to all the classes under which it is actually classified in a degree of 1. Measures hP and hR are then defined as:

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} \qquad hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|} \quad (6)$$

Since we are dealing with fuzzy sets, corresponding operations are used, defining the intersection of $\hat{C}_i$ and $\hat{C}'_i$ by:

$$(\hat{C}_i \cap \hat{C}'_i)(x) = \min(\hat{C}_i(x), \hat{C}'_i(x)) \quad (7)$$

for a keyword x, and defining the cardinality of $\hat{C}'_i$ as:

$$|\hat{C}'_i| = \sum_{x \in X} \hat{C}'_i(x) \quad (8)$$

By this definition, wrongly predicted classes are penalized less strongly when they belong to the same branch as one of the actual classes. The value of β can be chosen; in our case we will use $\beta = 1$, so both hierarchical precision and hierarchical recall have the same weight in hF.

All the results shown throughout the rest of the section have been calculated with this evaluation measure. On the other hand, the tests to obtain those results were carried out with a dataset formed by 9438 IWETO project descriptions, on which 10-fold cross validation was performed.

The results are subject to a number of parameter settings. The parameters with more impact in the results are the ones related with the term vector $\vec{d}$ used to represent a given document d, and the way the system chooses the classes under which the document d will be classified. Those problems are related with the parameter γ mentioned in Section 3.1, and the parameters N and δ mentioned in Section 3.2, respectively. All these parameters are discussed here and some results about their tuning are presented in Table 1.

Parameter γ is necessary to restrict the number of terms in every term vector $\vec{d}$, mainly because of memory usage and system performance reasons, since terms with very low TF-IDF weights would be probably irrelevant anyway if they were included in the term vector. Due to the short length of the documents we used, parameter γ is not very essential for our purposes. Therefore, the value used for our tests was $\gamma = 0$. Nevertheless, as mentioned above, its use is recommended when dealing with longer documents (or documents with an unknown length).

More relevant in our tests was the number of classes in which the system classifies the document: too many or too few classes probably signify a big difference with respect to the actual classification (resulting in too low precision and/or recall), and will also have an important impact on the user's satisfaction (if a document is classified in too many categories, it will be considered for the matching process even when it is not strongly related to a given category; if it is classified in just one category, it probably will not reach its whole target group of users). Parameters N and δ control this.

As said in Section 3.2, N sets the number of research fields in which a document will be classified, i.e., the number of classes that will be retrieved from the first places of the priority queue. On the other hand, a threshold δ is also necessary, in order to avoid cases of weak relationships: N is the number of classes with the highest value for the similarity measure sim, but that does not necessarily mean that all N values are high. Threshold δ tries to solve this. Since it is applied to normalized results, δ must be a value between 0 and 1, and since strong relationships are desired, the value cannot be too low. The results of the different tests are shown in Table 1.

Table 1: Results for tests with different values for N and δ.

δ	N	6	5	4	3	2	1
0.3	hF_1	0.502	0.512	0.523	**0.53**	0.527	0.478
	hP	0.419	0.442	0.472	0.508	0.555	0.623
	hR	0.625	0.609	0.587	0.555	0.501	0.388
0.4	hF_1	0.505	0.514	0.524	**0.53**	0.526	0.478
	hP	0.426	0.447	0.475	0.509	0.556	0.623
	hR	0.62	0.605	0.584	0.553	0.5	0.388
0.5	hF_1	0.509	0.517	0.525	**0.53**	0.526	0.478
	hP	0.439	0.457	0.481	0.513	0.557	0.623
	hR	0.607	0.595	0.577	0.548	0.497	0.388
0.6	hF_1	0.514	0.519	0.525	0.528	0.524	0.478
	hP	0.458	0.473	0.493	0.52	0.56	0.623
	hR	0.584	0.575	0.562	0.538	0.493	0.388
0.7	hF_1	0.516	0.519	0.522	0.524	0.52	0.478
	hP	0.487	0.496	0.511	0.531	0.564	0.623
	hR	0.549	0.546	0.535	0.518	0.483	0.388
0.8	hF_1	0.514	0.514	0.515	0.516	0.514	0.478
	hP	0.528	0.531	0.537	0.55	0.574	0.623
	hR	0.5	0.499	0.495	0.486	0.465	0.388
0.9	hF_1	0.503	0.503	0.503	0.503	0.501	0.478
	hP	0.577	0.577	0.578	0.581	0.591	0.623
	hR	0.445	0.445	0.444	0.443	0.435	0.388

The average number of classes to which a given document from the dataset belongs is 3. Due to this fact, $N = 3$ offers the best values for the evaluation measure hF_1. It can also be

ISBN: 978-989-95079-6-8

seen that $N = 3$ offers the best equilibrium between hP and hR. Moreover, note that the precision hP is inversely proportional to N: the fewer classes d is classified in, the higher the precision. This is logical since a low value for N means that the document is classified in those classes which offered the highest values for the similarity measure sim, classes that are then probably among those under which the document was actually classified. But that small number of classes has a drawback: the recall is of course lower. In the same way, a higher value of N results in higher values of hR: the document is classified under a lot of concepts and therefore it is more probable that they cover all the classes actually linked with the document. Of course, this higher number of classes makes the classifier less precise.

Also δ is important, mainly for higher values of N: these high values imply taking more classes into account when classifying a document, but δ imposes a threshold that leaves out all those "highest values" that are not actually high enough to be considered for the classification. In that way, δ allows higher values of N to be used in order to avoid leaving out potentially adequate classes, but without the risk of considering classes that probably are not so related to the document.

Note that the combination of $N = 1$ and δ is useless, as reflected by the results: since the results of sim are normalized, the highest value in the priority queue (and therefore the only one that will be retrieved when $N = 1$) will be equal to 1.

Although a high precision is desirable, it is necessary to keep in mind that the system aims to alert the researchers about potentially interesting documents. Therefore, the documents are preferably classified under more than one research field (as long as those fields are relatively strongly related to the document). That means that a balance between precision and recall ($\beta = 1$) is recommended in this case. As the combination $\delta = 0.5$, $N = 3$ offers the best results ($\delta = 0.4$ and $\delta = 0.3$ obtain similar values for hF but a lower precision), they are the values currently used by the system.

There are nonetheless some remarks to be made. In this case, the hierarchical classification problem gets complicated by the fact that the documents in the dataset have been manually classified by different people. This means that in some cases the document has been indexed under all the possible research fields while in other cases it has been vaguely classified under a parent concept. As previously explained, this can cause big differences between the set containing the research fields under which the documents were actually classified and that predicted by the system.

4 Mapper

Since the mapper is the most important part of the system, this section is used to explain in detail how it works. The mapper determines whether a document is interesting enough to notify a user about it. In this process, as discussed in the introduction, the system should be able to identify interesting documents even when their keywords do not exactly match those in the user's profile, but are semantically related to them. To achieve this added intelligence, some ideas from fuzzy-rough set theory are used.

Specifically, fuzzy-rough query expansion is applied. In particular, the system currently uses an adaptation of the approach described in [14]: to assess how well a document profile D matches a user profile U (both represented as fuzzy sets in the set X of keywords) the algorithm first uses a fuzzy relation R to generate their respective upper approximations $R{\uparrow}D$ and $R{\uparrow}U$, where the upper approximation of a fuzzy set A in X under a fuzzy relation R is defined as:

$$(R{\uparrow}A)(y) = \sup_{x \in X} \min(R(x, y), A(x)), \forall y \in X \qquad (9)$$

A keyword belongs to this upper approximation to the extent that it is related, by means of the fuzzy relation R, to at least one of the keywords in A. In other words, (9) defines the set of objects possibly belonging to A to a certain degree. In the current system, the relatedness of two terms is based on their co-occurrence during the training process, as explained in Section 2.2. To be precise, the relation R between two terms x and y can be defined as:

$$R(x, y) = \frac{|Docs(x) \cap Docs(y)|}{|Docs(x)|} \qquad (10)$$

The similarity between D and U is then computed, using α-cuts[6], as

$$Sim_\alpha(U, D) = 1 - \frac{|B_{u\alpha}|}{|(R{\uparrow}D)_\alpha|} \qquad (11)$$

where

$$B_u = R{\uparrow}D \ominus [R{\uparrow}U \cap R{\uparrow}D] \qquad (12)$$

with $\ominus$ the difference of A and B, defined by $(A \ominus B)(x) = \max(0, A(x) - B(x))$.

If $|(R{\uparrow}U)_\alpha| = 0$, the similarity is defined to be 0. Currently, the system uses a fixed value $\alpha = 0.5$, which turned out to yield the best performance experimentally in preliminary tests. It is important to point out that Sim is asymmetric and that therefore the order of U and D is relevant. This keeps the focus of the comparison on the document (B_u is the set of terms in the upper approximation of D that are not shared with the upper approximation of U) and allows the document to reach more users, as long as it is of relevance for them.

The obtained similarity between user and document is then compared to a user-set notification threshold (entered linguistically and then mapped onto a numerical value): if the similarity is greater than or equal to the threshold, user U will be notified by the system about document D. In this way, the user can tune the sensitivity of the mapper: if he only wants to receive alerts about documents that are definitely relevant, he can impose a higher threshold.

5 Conclusion and future work

We have shown how fuzzy relations are used to represent gradual relationships between research fields, and how these relations can be generated from a given dataset. It has also been shown how to construct user and document profiles as fuzzy sets, and how these sets can be matched by applying ideas from fuzzy-rough set theory. In addition, we have also proposed and explained a method to automatically classify documents, in order to keep the system easily updated. However, and despite the promising performance offered by the current

[6] The α-cut of A is defined as $A_\alpha = \{x \in X | A(x) \geq \alpha\}$, where $\alpha \in [0, 1]$

ISBN: 978-989-95079-6-8

implementation of the system, there is still a lot of work to be done.

Since the main goal of the project is to match researchers with documents that can be potentially useful for them, the main effort will be put into the mapping algorithms. New algorithms will be developed and tested to replace the one currently used. This current algorithm, though very interesting, is quite basic, so we think that there is a lot of room for improvement. For example, different similarity measures to compare the upper approximation can be investigated, and we plan to set up an experiment with a group of real test users.

The performance of the fuzzy-rough algorithms is of course also linked to the availability of sufficient document profiles and the quality of the classifier. However, the current version of this module of the system is quite simple, and further improvements could be very interesting and fruitful. For instance, more complex techniques from language technology could be used to refine the keyword extraction. Also, the fuzzy weights of the relationships between concepts can play a more important role, in the classification process as well as in the evaluation of results. The possibility of semi-automatically enriching the IWETO thesaurus with a fourth level, for the sake of precision, will be studied too. Besides, an alternative representation of the information, with the profiles as copies of the classification (which would allow the user to assign his own weights to the relations), will be considered.

Other modules will be added as well. In order to achieve a greater autonomy of the system, the feedback mechanism will be automated, and different techniques will be used to monitor the user's behavior while using the system, so that it can automatically update his list of interests and preferences, adding information that can be useful for the mapper. Also, the classifier could be used to extract additional information from the researcher's publications, when available. All this information can also be used, along with that provided by the feedback mechanism, to refine and adjust the relationships between concepts from the thesaurus. The user will also be free to explore the system on his own: a browsing utility and a search engine will be added with that finality. This engine can also use the same query expansion techniques that are used for personalized notification.

Acknowledgment

The authors thank the Flanders Research Information Space program for its cooperation. Also, Chris Cornelis thanks the Research Foundation-Flanders for supporting his research.

References

[1] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, 1994.

[2] T. Bogers and A. van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08)*, pages 287–290, 2008.

[3] S.M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S.K. Lam, A.M. Rashid, J.A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work (CSCW'02)*, pages 116–125, 2002.

[4] M. Montaner, B. López, and J.L. de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4):285–330, 2003.

[5] G. Adomavicius and A. Tuzhilin. Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[6] A. Motro and P. Smets. *Uncertainty management in information systems: from needs to solutions*. Springer, New York, NY, 1997.

[7] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[8] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[9] G. Hurtado Martín, C. Cornelis, and H. Naessens. Personalizing information retrieval in CRISs with fuzzy sets and rough sets. In *Proceedings of the 9th International Conference on Current Research Information Systems (CRIS2008)*, pages 51–59, 2008.

[10] A. Sieg, B. Mobasher, and R. Burke. Learning ontology-based user profiles: a semantic approach to personalized web search. *IEEE Intelligent Informatics Bulletin*, 8(1), 2007.

[11] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[12] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.

[13] S. Kiritchenko, S. Matwin, R. Nock, and A. Fazel Famili. Learning and evaluation in the presence of class hierarchies: application to text categorization. In *Canadian Conference on AI*, pages 395–406, 2006.

[14] P. Srinivasan, M.E. Ruiz, D.H. Kraft, and J. Chen. Vocabulary mining for information retrieval: rough sets and fuzzy sets. *Information Processing and Management*, 37(1):15–38, 2001.

ISBN: 978-989-95079-6-8